Pertemuan 10

DESAIN ARSITEKTUR

1. PENDAHULUAN

- Perancangan arsitektur merupakan tahap pertama dalam proses perancangan PL, yang dimulai dengan perancangan data kemudian berlanjut pada penurunan satu atau lebih struktur arsitektural sistem.
- Arsitektur sistem/PL adalah struktur sistem/PL yang menggabungkan komponen PL, menggabungkan properti yang tampak dari komponen tersebut, dan mendeskripsikan hubungan antar komponen.
- *Output* dari perancangan arsitektur berupa model arsitektur yang menggambarkan bagaimana sistem diatur sebagai satu set komponen yang saling berkomunikasi.

2. ARSITEKTUR PL

Arsitektur mencakup:

- Komponen bangunan yang berbeda dapat diintegrasikan menjadi suatu bentuk keseluruhan yang bersifat kohesif
- Bangunan yang dibuat sesuai dengan lingkungannya
- Bangunan yang dibangun sesuai dengan kegunaannya
- Tekstur, warna dan material pembentuknya dikombinasikan untuk membuat tampilan yang bagus
- Perancangan pencahayaan, template, dan garis batas
- Merupakan suatu bentuk seni

Arsitektur PL merupakan representasi yang memungkinkan untuk:

1. Melakukan analisis terhadap efektivitas perancangan dan disesuaikan dengan kebutuhan yang dinyatakan sebelumnya
2. Melakukan pertimbangan alternatif arsitektural pada tahap dimana perubahan rancangan dapat dilakukan dengan cara yang relatif mudah
3. Mengurangi risiko yang berhubungan dengan konstruksi PL

Alasan arsitektur PL:

- Representasi arsitektur PL adalah sesuatu yang memungkinkan terjadinya komunikasi di antara semua pihak yang tertarik pada pengembangan sistem berbasis komputer
- Arsitektur yang dibuat di awal perancangan akan memiliki efek yang menentukan pada semua pekerjaan RPL selanjutnya
- Arsitektur menggambarkan model yang relatif kecil dan mudah dipahami, dan menggambarkan bagaimana sistem distrukturkan dan bagaimana komponen di dalamnya saling bekerja sama.

A. Deskripsi Arsitektural

Sasaran dari deskripsi arsitektural:

- Untuk menetapkan kerangka kerja konseptual dan kosa kata yang digunakan selama perancangan arsitektur PL
- Untuk menyediakan panduan yang rinci pada waktu merepresentasikan deskripsi arsitektural
- Untuk memandu praktek perancangan yang baik

B. Keputusan Arsitektural

Pola Deskripsi Keputusan Arsitektur

a. Permasalahan Perancangan

Deskripsikan permasalahan perancangan arsitektural yang akan diselesaikan.

b. Penyelesaian

Menentukan pendekatan yang dipilih untuk menyelesaikan permasalahan yang bekaitan dengan perancangan

c. Kategori

Spesifikasi kategori perancangan yang akan diselesaikan permasalahannya, seperti perancangan data, struktur isi dan komponen, integrasi, presentasi

Pola Deskripsi Keputusan Arsitektur (Lanjutan)

d. Asumsi-asumsi

Indikasikan asumsi saat menentukan keputusan. Misalnya standar teknologi, pola yang tersedia, permasalahan yang berkaitan dengan sistem/PL

e. Alternatif-alternatif

Secara singkat deskripsikan alternatif yang akan dipertimbangkan dan mengapa ditolak

f. Argumen

Jelaskan mengapa memilih penyelesaian di atas dan alternatif-alternatif lainnya

Pola Deskripsi Keputusan Arsitektur (Lanjutan)

g. Keputusan yang berhubungan

Keputusan terdokumentasi yang berhubungan dengan keputusan yang diambil

h. Implikasi

Indikasikan konsekuensi perancangan akibat penentuan keputusan. Apakah penyelesaian akan berakibat pada perancangan lainnya?

i. Perhatian yang berhubungan

Adakah kebutuhan lain yang berhubungan dengan keputusan yang diambil?

Pola Deskripsi Keputusan Arsitektur (Lanjutan)

j. Produk kerja

Indikasikan dimana keputusan yang diambil akan tercermin dalam deskripsi arsitektur

k. Catatan

Rujukan catatan tim lainnya yang sebelumnya telah digunakan untuk membuat keputusan

Beberapa pertimbangan dalam keputusan Arsitektur:

1. Adakah arsitektur aplikasi generik yang dapat bertindak sebagai *template* untuk sistem yang sedang dirancang?
2. Bagaimana sistem akan didistribusikan ke sejumlah perangkat keras?
3. Pola atau gaya arsitektur apa yang digunakan?
4. Pendekatan fundamental apa yang digunakan untuk menyusun sistem?
5. Bagaimana komponen struktural dalam sistem akan terdekomposisi menjadi sub-komponen?

Beberapa pertimbangan dalam keputusan Arsitektur:

6. Strategi yang akan digunakan untuk mengontrol pengoperasian komponen dalam sistem
7. Organisasi arsitektur apa yang terbaik untuk memberikan persyaratan sistem non-fungsional?
8. Bagaimana desain arsitektur akan dievaluasi?
9. Bagaimana arsitektur sistem didokumentasikan?

3. TAMPILAN ARSITEKTURAL

1. **Tampilan Logis**
   Abstraksi dalam sistem sebagai objek atau kelas objek.
2. **Tampilan Proses**
   Menunjukkan bagaimana (pada saat *run-time*) sistem terdiri dari proses yang saling berinteraksi.
3. **Tampilan Pengembangan**
   PL diuraikan untuk pengembangan, yaitu menunjukkan *detail* dalam komponen yang akan diimplementasikan oleh pengembang tunggal atau tim pengembang.
4. **Tampilan Fisik**
   Menunjukkan perangkat keras sistem dan bagaimana komponen PL didistribusikan di seluruh sistem.

4. GAYA ARSITEKTUR

Gaya arsitektur mendeskripsikan kategori sistem yang mencakup:

- **Kumpulan komponen**, seperti sistem basis data dan modul-modul yang melaksanakan fungsi tertentu yang diperlukan oleh sistem
- **Penghubung (konektor)** yang memungkinkan komunikasi, koordinasi, dan kerja antar komponen
- **Batasan** yang mendefinisikan bagaiman komponen dapat iintegrasikan untuk membentuk suatu sistem/PL
- **Model semantik** yang memungkinkan perancang sistem memahami properti keseluruhan sistem

Gaya dan Struktur Arsitektur (Persyaratan Non-Fungsional)

1. **Kinerja (*Performance*)**

   Arsitektur harus dirancang agar semua komponen dapat digunakan pada berbagai komputer/prosesor, dan mendistribusikan di seluruh jaringan.

2. **Keamanan (*Security*)**

   Menggunakan struktur berlapis untuk melindungi aset yang paling penting di lapisan terdalam, dengan tingkat validasi keamanan yang tinggi.

3. **Keamanan (*Safety*)**

   Operasi yang terkait dengan keselamatan terletak di salah satu komponen tunggal atau komponen kecil.

Gaya dan struktur arsitektur (persyaratan non-fungsional)

4. Ketersediaan (*Availability*)

Arsitektur harus dirancang untuk menyertakan komponen redundan sehingga dimungkinkan saat mengganti dan memperbarui komponen tanpa menghentikan sistem.

5. Pemeliharaan (*Maintainability*)

Arsitektur sistem harus dirancang menggunakan komponen mandiri yang dapat diubah dengan mudah. Struktur data bersama harus dihindari.

Struktur Dasar Arsitektur

- Arsitektur PL merepresentasikan suatu struktur dimana beberapa kumpulan entitas (komponen) dihubungkan dengan sejumlah relasi (konektor).
- Komponen dan konektor dihubungkan dengan properti yang dapat membedakan jenis komponen dan konektor yang digunakan.

a. Struktur Fungsional

- Komponen merepresentasikan fungsi atau entitas.
- Konektor merepresentasikan antarmuka untuk melewatkan data ke suatu komponen.
- Properti mendefinisikan sifat dari komponen dan mengorganisasikan antarmuka.

Struktur Dasar Arsitektur (Lanjutan)

b. Struktur Implementasi

- Komponen berbentuk paket, kelas, objek, prosedur, fungsi, metode, dll, yang merupakan sarana untuk mengemas fungsionalitas komponen pada berbagai peringkat abstraksi.
- Konektor meliputi kemampuan untuk melewatkan data dan kendali, berbagi data, menggunakan, dan menginstansiasi.
- Properti pada komponen fokus pada karakteristik kualitas, seperti kemampuan untuk *maintenance* dan *reuse* yang dihasilkan saat struktur diimplementasikan.

Struktur Dasar Arsitektur (Lanjutan)

c. Struktur Konkurensi

- Komponen merepresentasikan unit-unit konkurensi yang terorganisasi sebagai pekerjaan paralel (*thread*).
- Konektor mencakup sinkronisasi, prioritas, mengirim data, dan menjalankan proses/fungsi.
- Properti mencakup prioritas, kemampuan untuk meramalkan, dan waktu eksekusi.

d. Struktur Fisik

- Komponen merupakan perangkat keras fisik.
- Konektor merupakan antarmuka antar komponen perangkat keras.
- Properti berkaitan dengan kapasitas, *bandwidth*, kinerja, dan atribut lainnya.

Struktur Dasar Arsitektur (Lanjutan)

e. Struktur Pengembangan

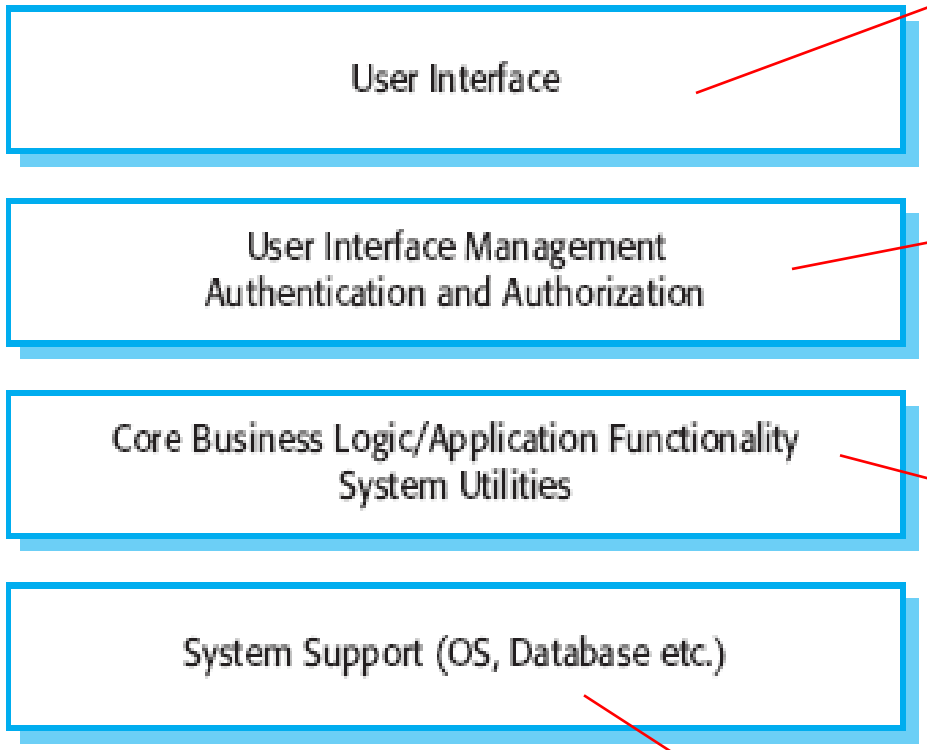
- Mendefinisikan komponen, produk kerja, dan sumber informasi lainnya.
- Konektor merepresentasikan relasi antar produk kerja.
- Properti mengidentifikasi karakteristik tiap-tiap *item*.

5. POLA ARSITEKTUR (*Architectural Patterns*)

A. Lapisan Arsitektur (*Layered Architecture*)

- Pemahaman tentang pemisahan dan independensi sangat penting untuk desain arsitektur karena memungkinkan perubahan secara lokal.
- Menambahkan tampilan baru atau mengubah tampilan yang ada dapat dilakukan tanpa perubahan apa pun pada data dalam model.

Gambar Generik Arsitektur Lapisan



Lapisan atas menyediakan fasilitas antarmuka pengguna

Lapisan aplikasi: komponen fungsionalitas aplikasi dan komponen utilitas

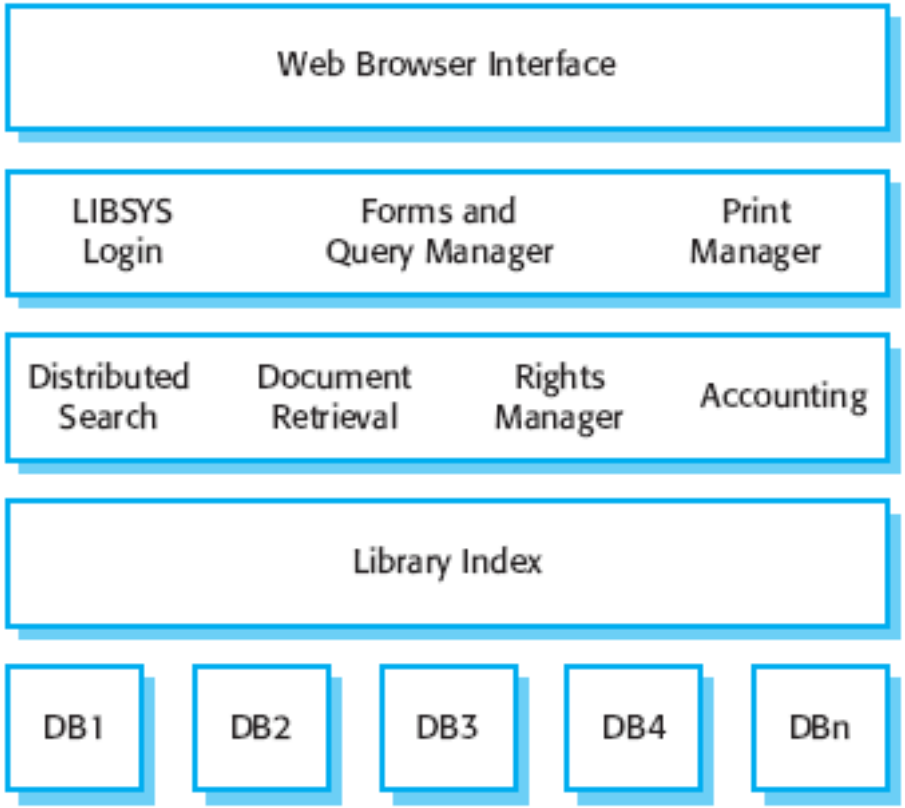
Lapisan ketiga: manajemen antarmuka pengguna dan menyediakan otentikasi dan otorisasi pengguna

Lapisan terendah: PL pendukung sistem (basis data dan OS)

Pola Arsitektur Berlapis

Name	Layered Architecture
Deskripsi	Mengatur sistem ke dalam lapisan dengan fungsi terkait. Lapisan menyediakan layanan ke lapisan di atasnya sehingga lapisan tingkat terendah mewakili layanan inti yang kemungkinan akan digunakan di seluruh sistem.
Contoh	Sebuah model berlapis dari suatu sistem untuk berbagi dokumen hak cipta yang disimpan di media penyimpanan.
Saat digunakan	• saat membangun fasilitas baru di atas sistem yang ada • ketika pengembangan tersebar di beberapa tim dengan tanggung jawab masing-masing tim • ketika ada persyaratan untuk keamanan multi-level
Keuntungan	Memungkinkan penggantian seluruh lapisan selama antarmuka dipertahankan. Fasilitas redundan (misal otentikasi) dapat disediakan di setiap lapisan untuk meningkatkan keandalan sistem.
Kerugian	Lapisan tingkat tinggi mungkin harus berinteraksi langsung dengan lapisan tingkat yang lebih rendah daripada melalui lapisan tepat di bawahnya. Kinerja dapat menjadi masalah karena beberapa tingkat interpretasi permintaan layanan diproses pada setiap lapisan.

Contoh arsitektur lapisan, dengan lapisan bawah menjadi basis data individual di setiap pustaka pada Sistem Perpustakaan



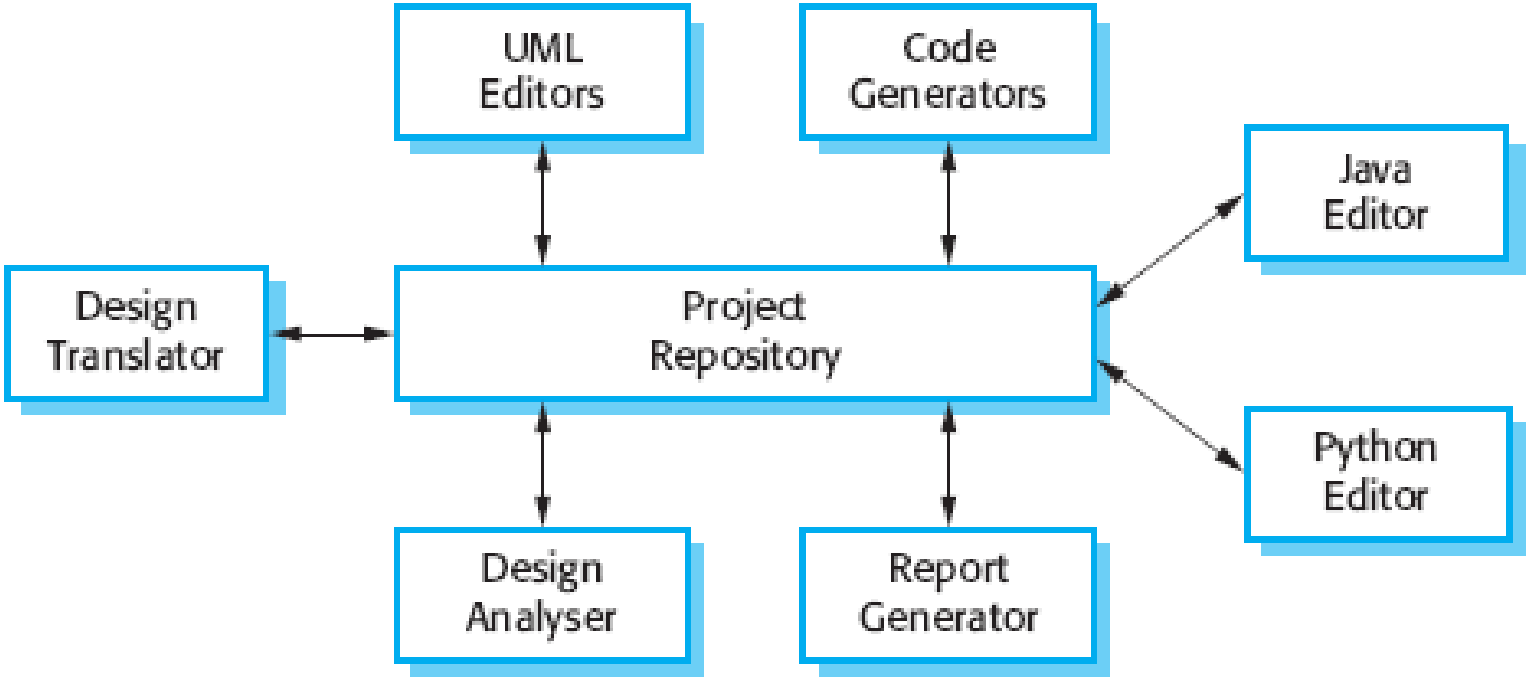
B. Arsitektur Repositori (*Repository Architecture*)

- Bagaimana satu set komponen yang saling berinteraksi dapat berbagi data.
- Model ini cocok untuk aplikasi di mana data dihasilkan oleh satu komponen dan digunakan oleh yang lain
- Tidak perlu mentransmisikan data secara eksplisit dari satu komponen ke komponen lainnya. Tetapi komponen harus beroperasi di sekitar model data repositori yang disepakati.
- Pola repositori berkaitan dengan struktur statis dari suatu sistem dan tidak menunjukkan organisasi *run-time*.

Repositori Arsitektur

Nama	Repository
Deskripsi	Semua data dalam sistem dikelola di repositori pusat yang dapat diakses oleh semua komponen .sistem
Contoh	Contoh dari IDE dimana komponen menggunakan repositori, dan setiap PL menghasilkan informasi yang kemudian tersedia untuk digunakan oleh alat lain.
Saat digunakan	Ketika sistem dengan sejumlah besar informasi yang dihasilkan disimpan untuk waktu yang lama.
Keuntungan	• Komponen dapat mandiri, karena tidak perlu mengetahui keberadaan komponen lain. • Perubahan yang dilakukan oleh satu komponen dapat disebarkan ke semua komponen. • Semua data dapat dikelola secara konsisten karena semuanya ada di satu tempat.
Kerugian	Masalah dalam repositori mempengaruhi seluruh sistem.

Contoh repositori arsitektur untuk sebuah IDE



Menunjukkan IDE yang mencakup alat yang berbeda untuk mendukung pengembangan berbasis model. Repositori dalam kasus ini adalah lingkungan yang dikendalikan oleh versi yang melacak perubahan pada PL dan memungkinkan *rollback* ke versi sebelumnya.

C. Client–Server Architecture

- Sebuah sistem yang mengikuti pola *client-server* diatur sebagai satu set layanan *server*, dan *client* yang mengakses dan menggunakan layanan.
- Komponen utama dari model ini adalah:
  1. **Server** memberikan layanan ke komponen lain. Contoh: *server* menawarkan layanan pencetakan, *server file* yang menawarkan layanan manajemen file, dan *server* kompilasi yang menawarkan layanan kompilasi bahasa pemrograman.
  2. *Client* yang menggunakan layanan yang ditawarkan oleh *server*.
  3. Jaringan yang memungkinkan *client* untuk mengakses layanan.

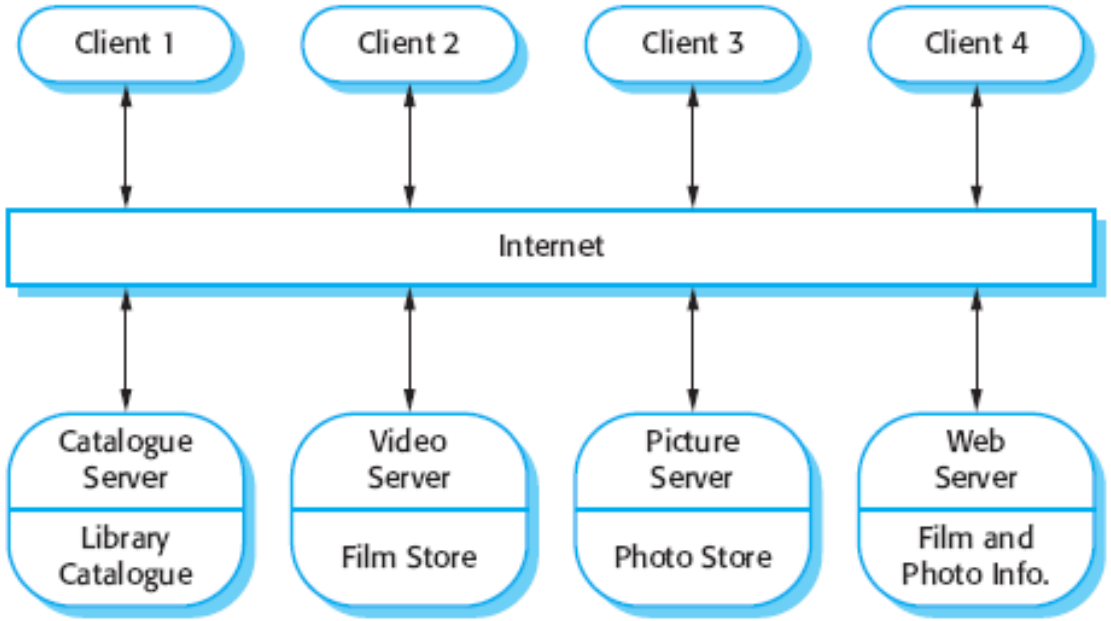
Client–Server Architecture (Lanjutan)

- Arsitektur *client-server* dianggap sebagai arsitektur sistem terdistribusi, tetapi model logis dari layanan independen yang berjalan pada *server* terpisah dapat diimplementasikan pada satu komputer
- Penggunaan yang efektif dapat dilakukan dari sistem jaringan dengan banyak prosesor terdistribusi.
- Sangat mudah untuk menambahkan server baru dan mengintegrasikannya dengan seluruh sistem atau meng-*upgrade server* secara transparan tanpa mempengaruhi bagian lain dari sistem.

Arsitektur *Client-Server*

Name	Client-Server
Deskripsi	Fungsionalitas sistem diatur ke dalam layanan, dengan setiap layanan yang dikirim dari *server* terpisah
Contoh	Contoh dari perpustakaan film/video
Saat digunakan	Ketika data dalam database harus diakses dari berbagai lokasi.
Keuntungan	• Server dapat didistribusikan melalui jaringan. • Fungsi umum dapat tersedia untuk semua *client* dan tidak perlu diterapkan di semua layanan.
Kerugian	• Setiap layanan dapat terjadi kegagalan sehingga rentan terhadap penolakan layanan atau kegagalan *server*. • Kinerja tidak dapat diprediksi karena tergantung pada jaringan dan juga sistem.

Contoh sistem perpustakaan film/video



- Dalam sistem ini, beberapa server mengelola dan menampilkan berbagai jenis media.
- Server video dapat menangani kompresi dan dekompresi video dalam berbagai format.
- Katalog harus dapat menangani pertanyaan dan menyediakan tautan ke dalam sistem informasi web yang mencakup data tentang film dan klip video, dan e-commerce mendukung penjualan foto, film, klip video.

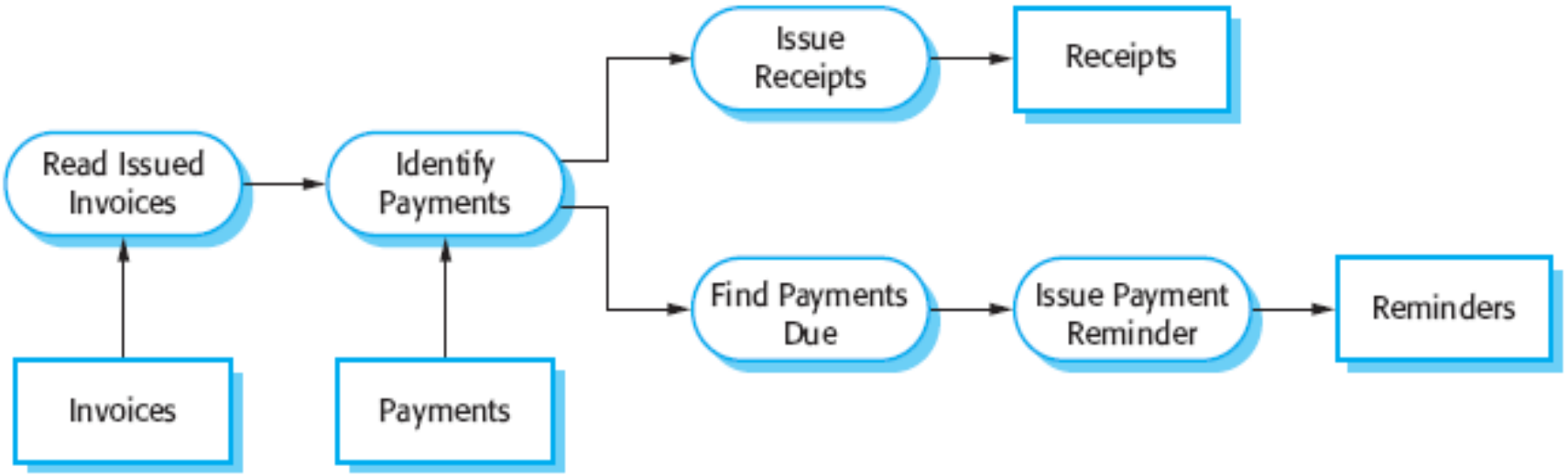
D. Pipe and Filter Architecture

- Model dari sistem *run-time* di mana transformasi secara fungsional memproses input dan menghasilkan output.
- Aliran data bergerak secara berurutan (seperti dalam pipa).
- Setiap langkah pemrosesan diimplementasikan sebagai transformasi.
- Transformasi dapat dilakukan secara berurutan/paralel.
- Data diproses oleh transformasi per-item-nya atau dalam satu *batch*.
- *Pipe* digunakan untuk melewati aliran teks dari satu proses ke proses lainnya.
- *Filter* digunakan pada transformasi untuk menyaring data.

Pipe and Filter Architecture

Name	Pipe and Filter
Deskripsi	Pengolahan data diatur dalam suatu sistem sehingga setiap komponen pemrosesan (*filter*) bersifat diskrit dan melakukan satu jenis transformasi data.
Contoh	Contoh pada sistem untuk memproses faktur.
Saat digunakan	Umumnya digunakan dalam aplikasi pemrosesan data (baik batch atau berbasis transaksi) di mana input diproses dalam tahap terpisah untuk menghasilkan output.
Keuntungan	• Mudah dimengerti dan mendukung transformasi *reuse*. • Gaya alur kerja cocok dengan struktur proses bisnis. • Dapat diimplementasikan sebagai sistem sekuensial/konkuren.
Kerugian	• Format transfer data harus disepakati di antara transformasi komunikasi. • Setiap transformasi harus memahami input dan tidak mempublikasikan outputnya ke bentuk yang tidak dipahami. • Meningkatkan *overhead* sistem, berarti bahwa tidak mungkin menggunakan kembali transformasi fungsional yang menggunakan struktur data yang tidak kompatibel.

Contoh pada sistem untuk memproses faktur



Suatu organisasi telah menerbitkan faktur kepada pelanggan. Seminggu sekali, pembayaran yang telah dilakukan direkonsiliasi dengan faktur.

Untuk faktur yang telah dibayarkan, diberikan tanda terima. Untuk faktur yang belum dibayar dalam waktu pembayaran yang ditentukan, diberikan pesan untuk mengingatkan

6. ARSITEKTUR APLIKASI

- Sistem aplikasi dimaksudkan untuk memenuhi kebutuhan bisnis yang memiliki banyak kesamaan dan menggunakan aplikasi tertentu.
- Arsitektur aplikasi dapat diimplementasikan kembali ketika mengembangkan sistem baru, tetapi untuk banyak sistem bisnis, penggunaan kembali aplikasi dimungkinkan tanpa implementasi ulang

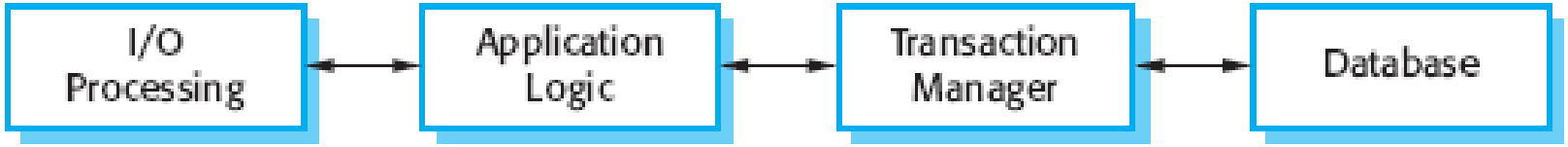
A. Sistem Pemrosesan Transaksi (*Transaction Processing Systems*)

- Aplikasi pemrosesan transaksi adalah aplikasi yang berpusat pada database yang memproses permintaan pengguna untuk informasi dan memperbarui informasi dalam basis data.
- Merupakan jenis sistem bisnis interaktif yang paling umum, di mana pengguna membuat permintaan *asynchronous* untuk layanan
- Transaksi basis data adalah urutan operasi yang diperlakukan sebagai unit tunggal, dan semua operasi dalam transaksi harus diselesaikan sebelum perubahan basis data dibuat permanen.

Aplikasi Pemrosesan Transaksi (Lanjutan)

- Dari perspektif pengguna, transaksi adalah setiap urutan operasi yang koheren yang memenuhi tujuan, seperti menemukan jadwal perkuliahan.
- Sistem pemrosesan transaksi dapat diatur sebagai arsitektur *'pipe and filter*' dengan komponen sistem sebagai input, pemrosesan, dan output.
- Misal: pelanggan menarik uang tunai dari ATM. Sistem ini terdiri dari dua komponen PL ATM dan PL pemrosesan akun di server basis data bank. Komponen I/O diimplementasikan sebagai PL di ATM dan komponen pemrosesan adalah bagian dari server database bank.

Contoh Aplikasi Pemrosesan Transaksi



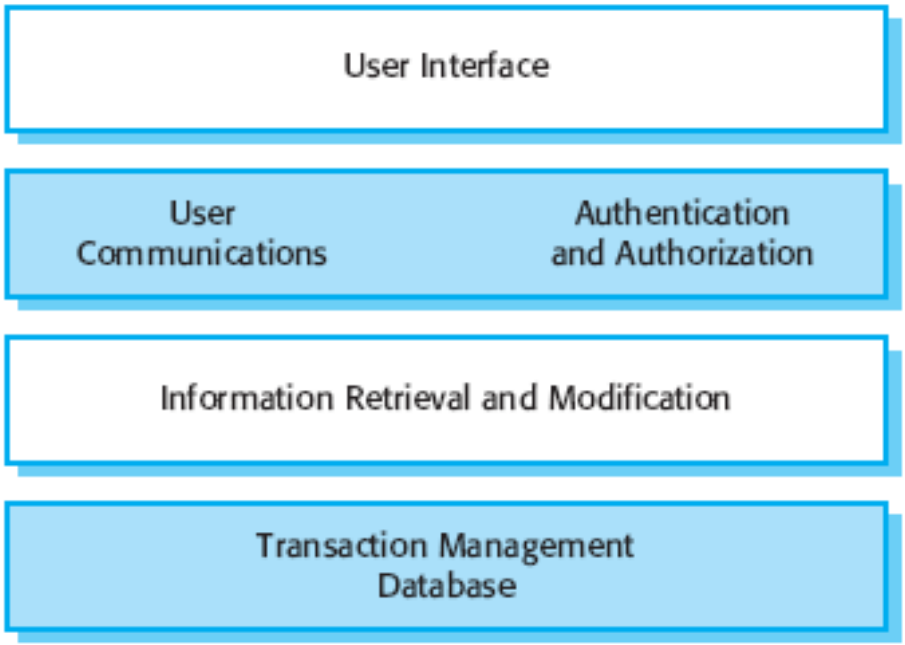
Penjelasan

- Pengguna membuat permintaan ke sistem melalui komponen pemrosesan I / O.
- Permintaan diproses oleh beberapa aplikasi logika.
- Transaksi dibuat dan diteruskan ke manajer transaksi, yang biasanya tertanam dalam sistem manajemen basis data.
- Setelah manajer transaksi memastikan bahwa transaksi sudah diselesaikan dengan benar, kemudian memberi sinyal ke aplikasi bahwa proses telah selesai

B. Sistem Informasi

- Semua sistem yang melibatkan interaksi dengan basis data dapat dianggap sebagai sistem informasi berbasis transaksi.
- Sistem informasi memungkinkan akses yang terkontrol ke basis informasi yang besar. Seperti katalog perpustakaan, jadwal penerbangan, atau catatan pasien di rumah sakit.
- Sebagai contoh dari instantiation model berlapis

Contoh Sistem Informasi



- Sistem dimodelkan menggunakan pendekatan berlapis di mana lapisan atas mendukung antarmuka pengguna dan lapisan bawah adalah database sistem.
- Lapisan komunikasi pengguna menangani semua I/O dari antarmuka pengguna, dan lapisan pencarian informasi untuk mengakses dan memperbarui database

C. Sistem Pemrosesan Bahasa (*Language Processing Systems*)

- Adalah sistem di mana maksud pengguna dinyatakan dalam bahasa formal (seperti Java).
- Memproses ke dalam bahasa formal, kemudian menafsirkan representasi secara internal.
- Sistem pemrosesan bahasa dengan *compiler*, yang menerjemahkan bahasa program tingkat tinggi ke dalam kode mesin.
- Sistem pemrosesan bahasa juga menerjemahkan bahasa alami atau buatan ke dalam representasi bahasa lain, dan bahasa pemrograman dapat mengeksekusi kode yang dihasilkan.

Contoh Sistem Pemrosesan Bahasa