Pertemuan 2 :
LINGKUNGAN BASIS DATA

Tujuan Instruksional Khusus :

- Mahasiswa dapat menjelaskan tingkatan arsitektur basis data
- Mahasiswa dapat menjelaskan konsep *data independence*, komponen DBMS, fungsi DBMS serta bahasa yang digunakan di dalam DBMS
- Mahasiswa dapat mejelaskan perbedaan model data berbasis objek, *record*, konseptual, dan fisik
- Mahasiswa dapat menjelaskan fungsi dan isi dari *data dictionary*
- Mahasiswa dapat menjelaskan perbedaan arsitektur DBMS *multi user*

Deskripsi Singkat :

Basis data merupakan sumber informasi yang dapat dipakai bersama. Setiap pemakai membutuhkan pandangan yang berbeda terhadap data yang disimpan di dalam basis data. Untuk memenuhi kebutuhan tersebut, terdapat arsitektur komersial DBMS yang didasarkan pada perluasan arsitektur yang disebut sebagai arsitektur ANSI-SPARC. Oleh karena itu, materi ini akan membahas tingkatan arsitektur basis data, karakteristik fungsional DBMS, bahasa yang digunakan di dalam DBMS serta model-model data. Materi-materi tersebut mengetengahkan latar belakang informasi yang penting pada DBMS

Bahan Bacaan :

1. Connoly, Thomas; Begg, Carolyn; Strachan, Anne; ***Database Systems : A Practical Approach to Design, Implementation and Management***, 3rd edition, Addison Wesley, 2001.

2. Korth, H.; ***Database System Concept***, 4th edition, Mc Graw Hill, New York, 1991.

LINGKUNGAN BASIS DATA

Tujuan utama dari sistem basis data adalah menyediakan pemakai melalui suatu pandangan abstrak mengenai data, dengan menyembunyikan detail dari bagaimana data disimpan dan dimanipulasikan. Oleh karena itu, titik awal untuk perancangan sebuah basis data haruslah abstrak dan deskripsi umum dari kebutuhan-kebutuhan informasi suatu organisasi harus digambarkan di dalam basis data.

Lebih jauh lagi, jika sebuah basis data merupakan suatu sumber yang bisa digunakan bersama maka setiap pemakai membutuhkan pandangan yang berbeda-beda terhadap data di dalam basis data. Untuk memenuhi kebutuhan ini, arsitektur komersial basis data yang banyak digunakan telah tersedia saat ini dan telah mengalami perluasan yaitu arsitektur ANSI-SPARC.

Materi ini menyediakan latar belakan informasi yang penting pada basis data, diantaranya tiga tingkatan arsitektur ANSI-SPARC, pengenalan model data, fungsi yang disediakan oleh DBMS multi user.

Tiga Tingkatan Arsitektur Basis data ANSI-SPARC

Ada 3 tingkat dalam arsitektur basis data yang bertujuan membedakan cara pandang pemakai terhadap basis data dan cara pembuatan basis data secara fisik.
3 tingkatan arsitektur basis data :

1. *Tingkat Eksternal (External Level)*

   Tingkat eksternal merupakan cara pandang pemakai terhadap basis data. Pada tingkat ini menggambarkan bagian basis data yang relevan bagi seorang pemakai tertentu. Tingkat eksternal terdiri dari sejumlah cara pandang yang berbeda dari sebuah basis data. Masing-masing pemakai merepresentasikan dalam bentuk yang sudah dikenalnya. Cara pandang secara eksternal hanya terbatas pada entitas, atribut dan hubungan antar entitas (*relationship*) yang diperlukan saja.
2. *Tingkat Konseptual (Conseptual Level)*

Tingkat konseptual merupakan kumpulan cara pandang terhadap basis data. Pada tingkat ini menggambarkan data yang disimpan dalam basis data dan hubungan antara datanya.

Hal-hal yang digambarkan dalam tingkat konseptual adalah :

- semua entitas beserta atribut dan hubungannya
- batasan data
- informasi semantik tentang data
- keamanan dan integritas informasi

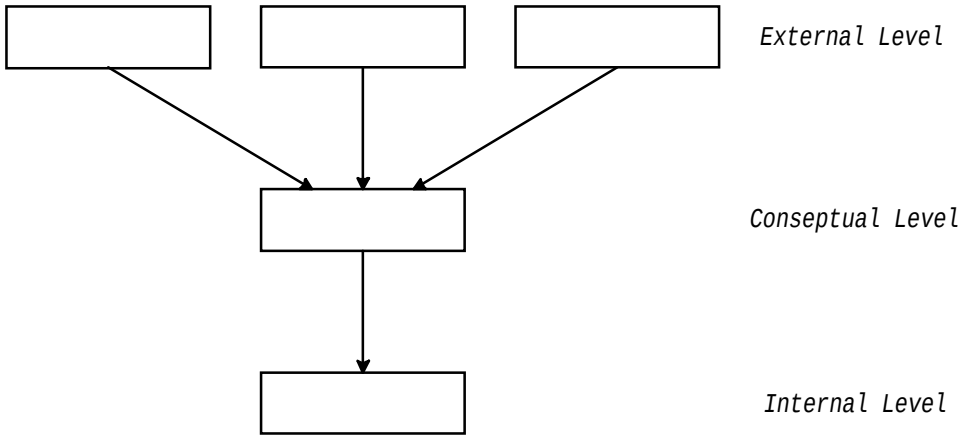
Semua cara pandang pada tingkat eksternal berupa data yang dibutuhkan oleh pemakai harus sudah tercakup di dalam tingkat konseptual atau dapat diturunkan dari data yang ada. Deskripsi data dari entitas pada tingkat ini hanya terdiri dari jenis data dan besarnya atribut tanpa memperhatikan besarnya penyimpanan dalam ukuran byte.

3. *Tingkat Internal (Internal Level)*

Tingkat internal merupakan perwujudan basis data dalam komputer. Pada tingkat ini menggambarkan bagaimana basis data disimpan secara fisik di dalam peralatan *storage* yang berkaitan erat dengan tempat penyimpanan / *physical storage*.

Tingkat internal memperhatikan hal-hal berikut ini :

- alokasi ruang penyimpanan data dan indeks
- deskripsi record untuk penyimpanan (dengan ukuran penyimpanan untuk data elemen
- penempatan record
- pemampatan data dan teknik encryption



Gambar 1. Tingkatan Arsitektur Basis data

Contoh : sebuah file Pegawai yang akan didefinisikan

Tingkat Eksternal (External Level) :

Cobol

```
01 PEG_REC.
    02 PEG_NO     PIC X(6).
    02 DEPT_NO    PIC X(4).
    02 GAJI       PIC 9(6).
```

Tingkat Konseptual (Conceptual Level) :

PEGAWAI		
NOMOR_PEGAWAI	CHARACTER	6
NOMOR_DEPT	CHARACTER	4
GAJI	NUMERIC	6

Tingkat Internal (Internal Level) :

FILE_PEGAWAI	LENGTH = 22
PREFIX	TYPE = BYTE (6), OFFSET = 0
EMP#	TYPE = BYTE (6), OFFSET = 6, INDEX = EMPX
DEPT#	TYPE = BYTE (4), OFFSET = 12
PAY	TYPE = FULLWORD, OFFSET = 16

Data Independence

Tujuan utama dari 3 tingkat arsitektur adalah memelihara kemandirian data (*data independence*) yang berarti perubahan yang terjadi pada tingkat yang lebih rendah tidak mempengaruhi tingkat yang lebih tinggi.

Ada 2 jenis *data independence*, yaitu

1. *Physical Data Independence*
   bahwa *internal schema* dapat diubah oleh DBA tanpa menggangu *conceptual schema*. Dengan kata lain *physical data independence* menunjukkan kekebalan *conceptual schema* terhadap perubahan *internal schema*.
2. *Logical Data Independence*
   bahwa *conceptual schema* dapat diubah oleh DBA tanpa menggangu *external schema*. Dengan kata lain *logical data independence* menunjukkan kekebalan *external schema* terhadap perubahan *conceptual schema*.

Prinsip *data independence* adalah salah satu hal yang harus diterapkan di dalam pengelolaan sistem basis data dengan alasan-alasan sbb :

1. DBA dapat mengubah isi, lokasi, perwujudan dalam organisasi basis data tanpa mengganggu program-program aplikasi yang sudah ada.
2. Pabrik / agen peralatan / *software* pengolahan data dapat memperkenalkan produk-produk baru tanpa mengganggu program-program aplikasi yang sudah ada.
3. Untuk memindahkan perkembangan program-program aplikasi
4. Memberikan fasilitas pengontrolan terpusat oleh DBA demi keamanan dan integritas data dengan memperhatikan perubahan-perubahan kebutuhan pengguna.

Bahasa Dalam DBMS

DBMS (*Database Management systems*) adalah kumpulan program yang mengkoordinasikan semua kegiatan yang berhubungan dengan basis data. Dengan adanya berbagai tingkatan pandangan dalam suatu basis data maka untuk mengakomodasikan masing-masing pengguna dalam piranti lunak manajemen basis data biasanya terdapat bahasa-bahasa tertentu yang disebut ***Data Sub language***.

Data sub language adalah subset bahasa yang dipakai untuk operasi manajemen basis data. Dalam penggunaan biasanya dapat ditempelkan (*embedded*) pada bahasa tuan rumah (Cobol, PL/1, dsb). Secara umum maka setiap pengguna basis data memerlukan bahasa yang dipakai sesuai tugas dan fungsinya.

Dalam basis data secara umum dikenal 2 data sub language :

1. *Data Definition Language (DDL)*
   Bahasa yang digunakan dalam mendefinisikan struktur atau kerangka dari basis data, di dalamnya termasuk *record*, elemen data, kunci elemen, dan relasinya
2. *Data Manipulation Language (DML)*
   Bahasa yang digunakan untuk menjabarkan pemrosesan dari basis data, fasilitas ini diperlukan untuk memasukkan, mengambil, mengubah data. DML dipakai untuk operasi terhadap isi basis data

   Ada 2 jenis DML :
   1. *Procedural DML*
      Digunakan untuk mendefinisikan data yang diolah dan perintah yang akan dilaksanakan.
   2. *Non Procedural*
      Digunakan untuk menjabarkan data yang diinginkan tanpa menyebutkan bagaimana cara pengambilannya.

Secara khusus pengguna menggunakan berbagai bahasa :
Programmer aplikasi menggunakan bahasa-bahasa seperti Cobol, Informix, dll (*host language*) yang ditempelkan dengan bahasa yang dipakai dalam DBMS. Pemakai terminal menggunakan bahasa Query (misal SQL) atau menggunakan program aplikasi (yang dirancang oleh programmer). Sedangkan DBA lebih banyak menggunakan bahasa DDL dan DML yang tersedia dalam DBMS.

DBMS mempunyai tugas untuk menangani semua bentuk akses kepada basis data, secara konsep :

1. Pengguna menyatakan permintaan akses menggunakan DBMS
2. DBMS menangkap dan menginterpretasikan
3. DBMS mencari :

- eksternal / *conceptual mapping*
- *conceptual schema*
- konseptual / *internal mapping*
- *internal schema*

4. DBMS melaksanakan operasi yang diminta terhadap basis data tersimpan.

Proses 1 s/d 4 dapat dilakukan secara interactive atau dicompile dulu.

Model Data

Model data adalah kumpulan konsep yang terintegrasi yang menggambarkan data, hubungan antara data dan batasan-batasan data dala suatu organisasi. Fungsi dari sebuah model data untuk merepresentasikan data sehingga data tersebut mudah dipahami.

Untuk menggambarkan data pada tingkat eksternal dan konseptual digunakan model data berbasis objek atau model data berbasis *record*.

1. *Model Data Berbasis Objek*

   Model data berbasis objek menggunakan konsep entitas, atribut dan hubungan antar entitas. Beberapa jenis model data berbasis objek yang umum adalah :

   - *entity-relationship*
   - *semantic*
   - *functional*
   - *object-oriented*

2. *Model Data Berbasis Record*

   Pada model data berbasis *record*, basis data terdiri dari sejumlah *record* dalam bentuk yang tetap yang dapat dibedakan dari bentuknya. Ada 3 macam jenis model data berbasis *record* yaitu :

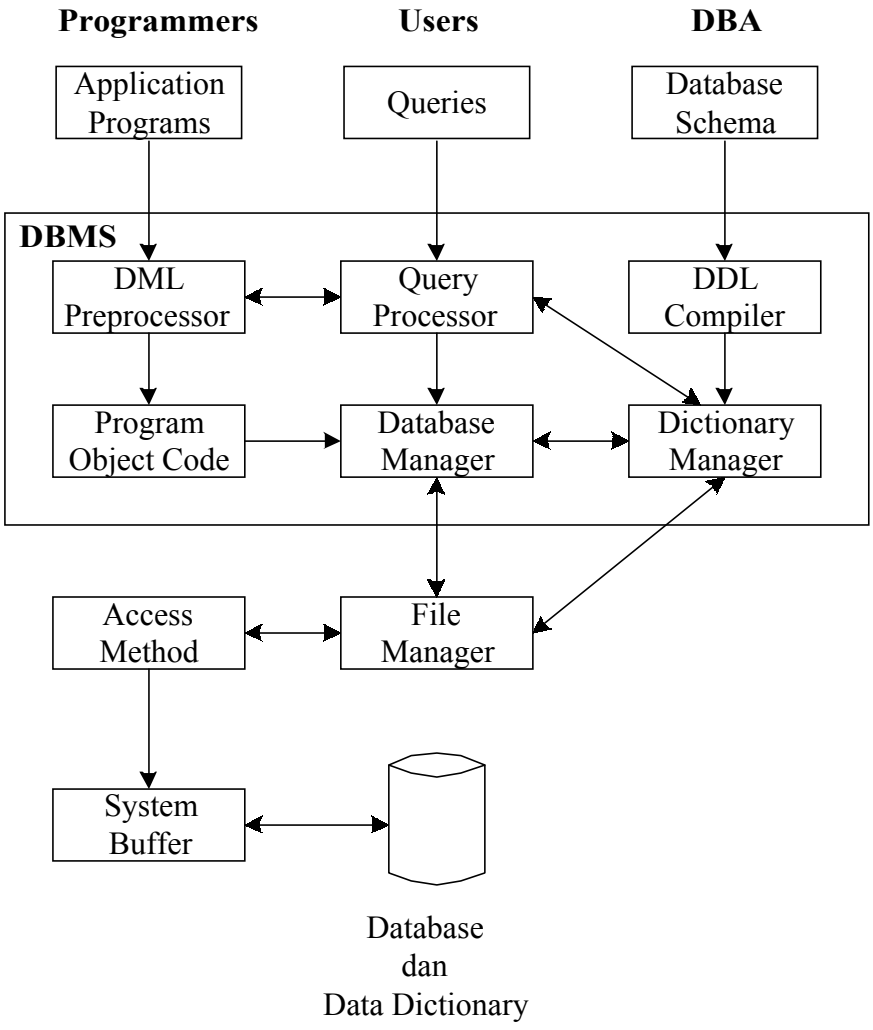
   - model data relasional (*relational*)
   - model data hierarkhi (*hierarchical*)
   - model data jaringan (*network*)

Fungsi DBMS

Layanan-layanan yang sebaiknya disediakan oleh *database management system* adalah :

1. Penyimpanan, pengambilan dan perubahan data
   Sebuah DBMS harus menyediakan kemampuan menyimpan, mengambil dan merubah data dalam basis data.
2. Katalog yang dapat diakses pemakai
   menyediakan sebuah katalog yang berisi deskripsi item data yang disimpan dan diakses oleh pemakai.
3. Mendukung Transaksi
   Menyediakan mekanisme yang akan menjamin semua perubahan yang berhubungan dengan transaksi yang sudah ada atau yang akan dibuat.
4. Melayani kontrol *concurrency*
   Sebuah DBMS harus menyediakan mekanisme yang menjamin basis data ter-update secara benar pada saat beberapa pemakai melakukan perubahan terhadap basis data yang sama secara bersamaan.
5. Melayani *recovery*
   Menyediakan mekanisme untuk mengembalikan basis data ke keadaan sebelum terjadinya kerusakan pada basis data tersebut.
6. Melayani autorisasi
   Sebuah DBMS harus menyediakan mekanisme untuk menjamin bahwa hanya pemakai yang berwenang saja yang dapat mengakses basis data.
7. Mendukung komunikasi data
   Sebuah DBMS harus mampu terintegrasi dengan software komunikasi.
8. Melayani integrity
   Sebuah DBMS bertujuan untuk menjamin semua data dalam basis data dan setiap terjadi perubahan data harus sesuai dengan aturan yang berlaku.
9. Melayani *data independence*
   Sebuah DBMS harus mencakup fasilitas untuk mendukung kemandirian program dari struktur basis data yang sesungguhnya.
10. Melayani *utility*
    Sebuah DBMS sebaiknya menyediakan kumpulan layanan *utility*.

Komponen DBMS



Gambar 2. Komponen DBMS

1. *Query Processsor*

   Komponen yang merubah bentuk *query* ke dalam instruksi tingkat rendah ke *database manager*

2. *Database Manager*

   Database manager menerima *query* dan menguji skema eksternal dan konseptual untuk menentukan apakah *record-record* dibutuhkan untuk memenuhi permintaan. Kemudian DM memanggil *file manager* untuk menyelesaikan permintaan

3. *File Manager*

Memanipulasi penyimpanan file dan mengatur alokasi ruang penyimpanan pada disk.

4. DML *Preprocessor*

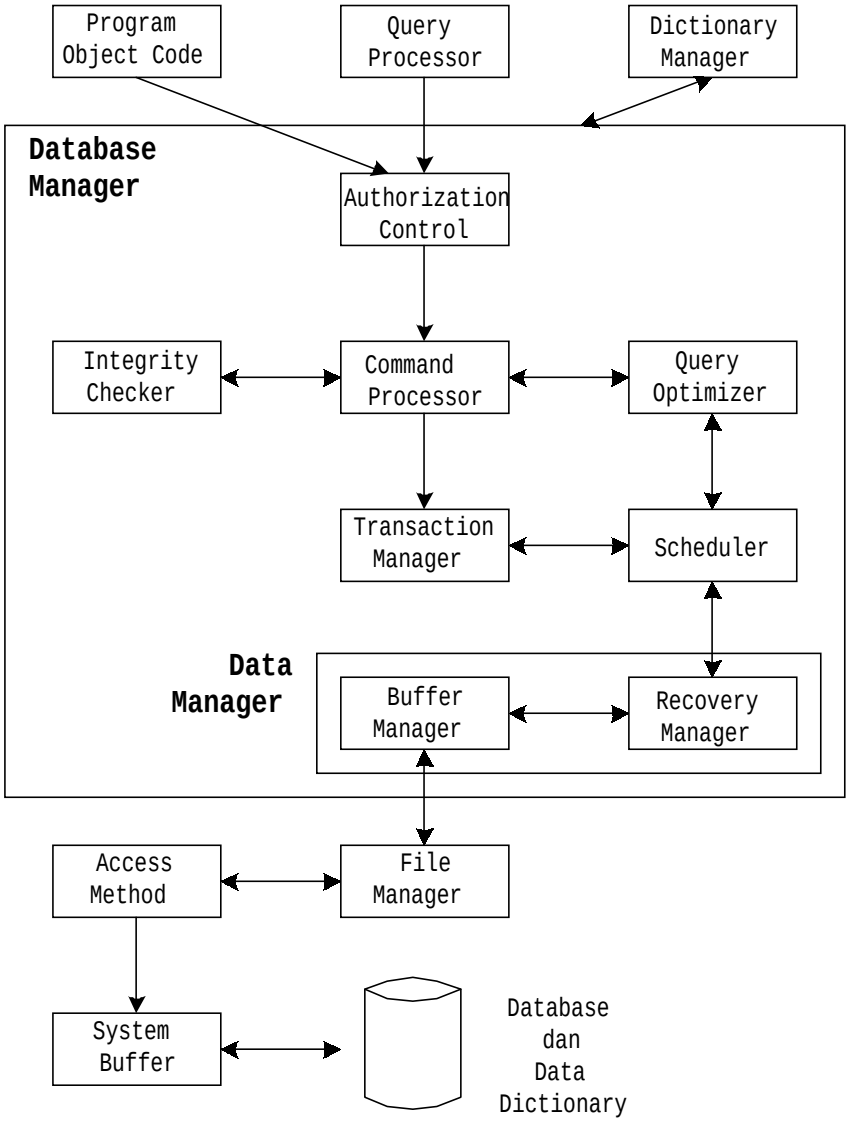
   Modul yang merubah perintah DML *embedded* ke dalam program aplikasi dalam bentuk fungsi-fungsi yang memanggil dalam *host language*.

5. DDL *Compiler*

   Merubah perintah DDL menjadi kumpulan tabel yang berisi metadata.

6. *Dictionary Manager*

   Mengatur akses dan memelihara *data dictionary*. *Data dictionary* diakses oleh komponen DBMS yang lain.



Gambar 3. Komponen *Software* Utama *Database Manager*

Komponen *software* utama *database manager* adalah

1. *Authorization Control*

   Modul yang memeriksa apakah pemakai mempunyai wewenang untuk menyelesaikan operasi
2. *Command Processor*

   Memeriksa apakah pemakai mempunyai wewenang untuk menyelesaikan operasi
3. *Integrity Checker*

   Untuk semua operasi yang merubah basis data, *integrity checker* memeriksa operasi yang diminta memerlukan batasan integritas.
4. *Query Optimizer*

   Modul ini menentukan strategi yang optimal untuk eksekusi *query*
5. *Transaction Manager*

   Modul ini mengerjakan proses-proses yang dibutuhkan operasi yang diterima transaksi
6. *Scheduler*

   Modul ini bertanggung jawab untuk menjamin operasi secara bersamaan terhadap basis data sehingga berjalan tanpa ada masalah antara yang satu dengan yang lain.
7. *Recovery Manager*

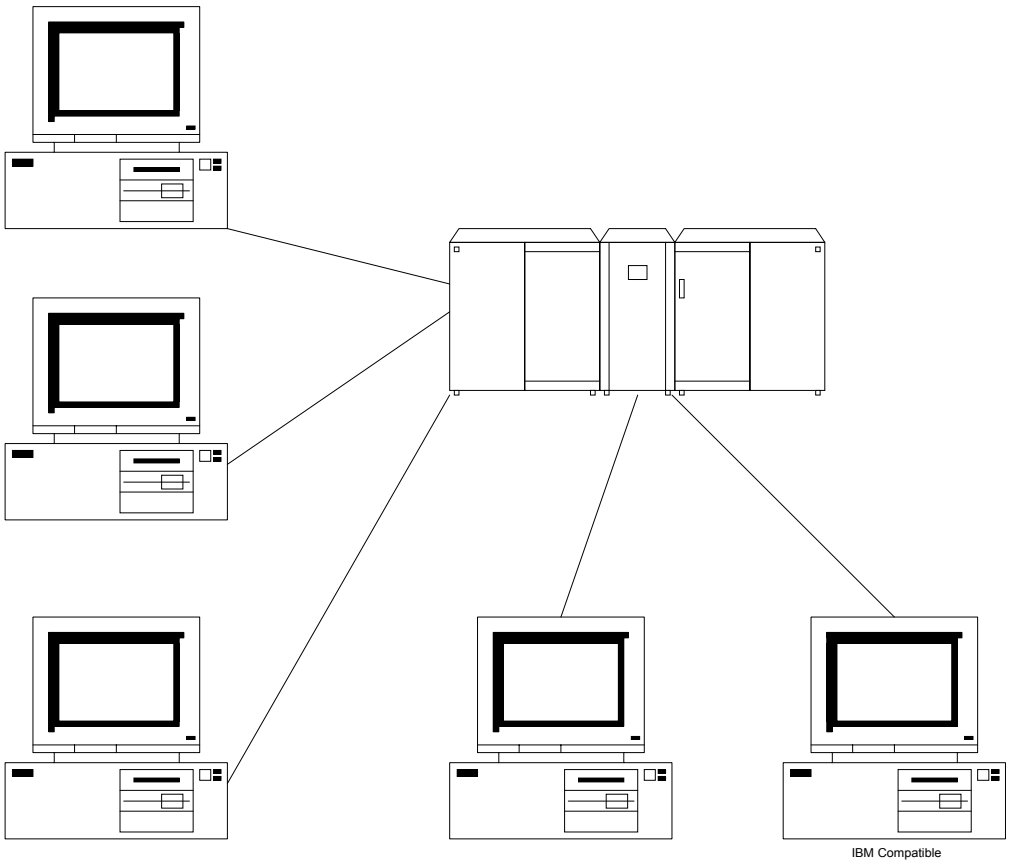
   Modul ini menjamin basis data tetap konsisten walaupun terjadi kerusakan.
8. *Buffer Manager*

   Modul ini bertanggung jawab terhadap pemindahan data antara *main memory* dan *secondary storage*, seperti *disk* dan *tape*.

Arsitektur DBMS Multi User

Teleprocessing

Arsitektur tradisional untuk sistem multi user adalah *teleprocessing*, dimana satu komputer dengan sebuah CPU dan sejumlah terminal seperti pada gambar di bawah ini.



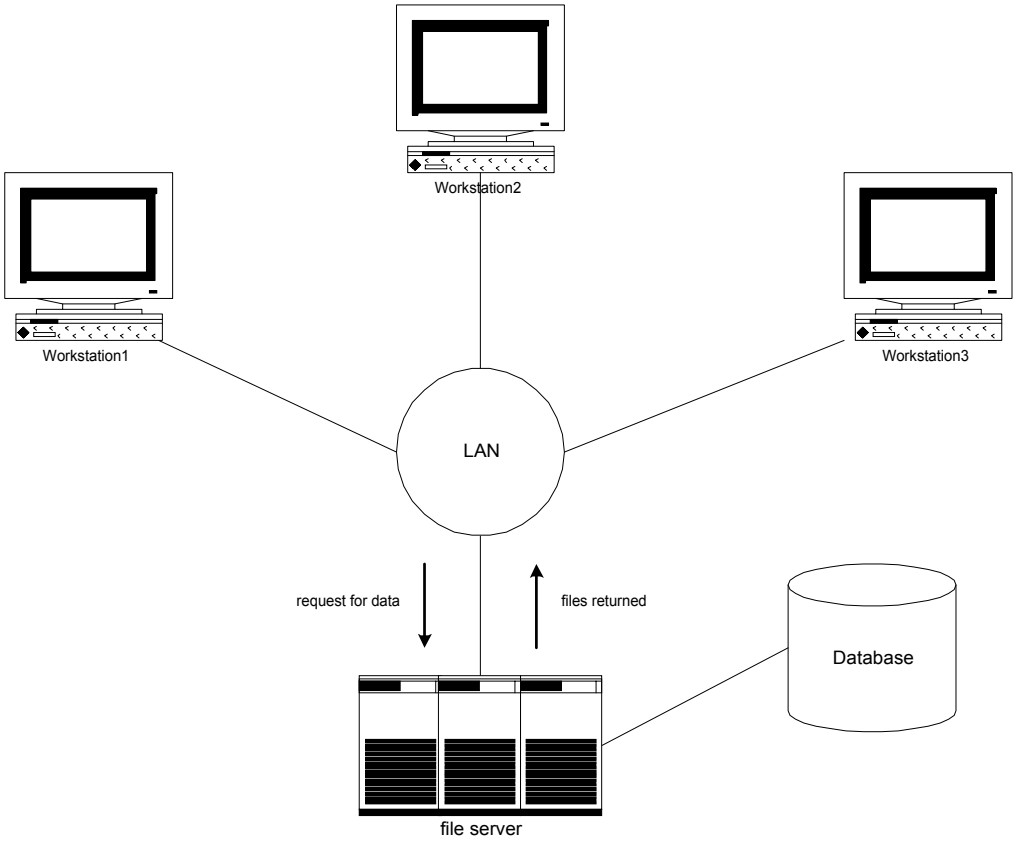
Gambar 4. Arsitektur *Teleprocessing*

Semua pemrosesan dikerjakan dalam batasan fisik komputer yang sama. Terminal untuk pemakai berjenis '*dumb*', yang tidak dapat berfungsi sendiri dan masing-masing dihubungkan ke komputer pusat. Terminal-terminal tersebut mengirimkan pesan melalui subsistem pengontrol komunikasi pada sistem operasi ke program aplikasi, yang bergantian menggunakan layanan DBMS.

Dengan cara yang sama, pesan dikembalikan ke terminal pemakai. Arsitektur ini menempatkan beban yang besar pada komputer pusat yang tidak hanya menjalankan program aplikasi tetapi juga harus menyelesaikan sejumlah pekerjaan pada terminal seperti format data untuk tampilan di monitor.

File-Server

Proses didistribusikan ke dalam jaringan sejenis LAN (*Local Area Network*). *File server* mengendalikan file yang diperlukan oleh aplikasi dan DBMS. Meskipun aplikasi dan DBMS dijalankan pada masing-masing *workstation* tetapi tetap meminta file dari *file server* jika diperlukan (perhatikan gambar di halaman berikut ini).



Gambar 5. Arsitektur File Server

Dengan cara ini, *file server* berfungsi sebagai sebuah *hard disk* yang digunakan secara bersamaan.

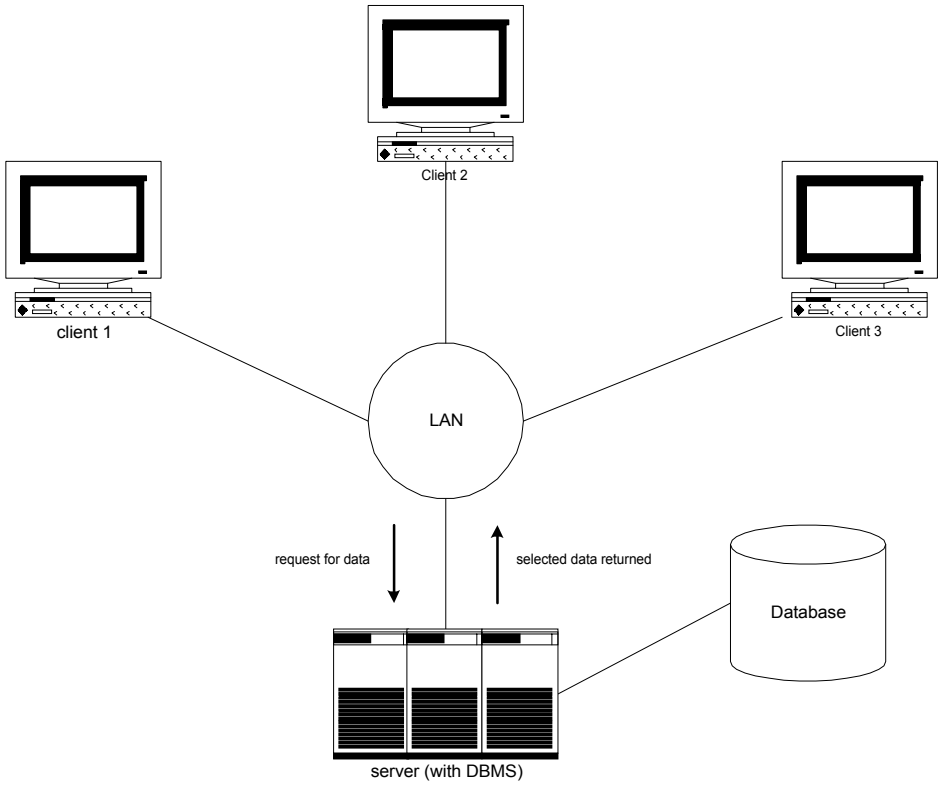
Kerugian arsitektur *file-server* adalah :

- Terdapat lalulintas jaringan yang besar
- Masing-masing *workstation* membutuhkan copy DBMS
- Kontrol terhadap *concurrency*, *recovery* dan *integrity* menjadi lebih kompleks karena sejumlah DBMS mengakses file secara bersamaan

Client Server

Untuk mengatasi kelemahan arsitektur-arsitektur di atas maka dikembangkan arsitektur *client-server*. *Client-server* menunjukkan cara komponen *software* berinteraksi dalam bentuk sistem.

Sesuai dengan namanya, ada sebuah pemroses *client* yang membutuhkan sumber dan sebuah *server* yang menyediakan sumbernya. Tidak ada kebutuhan *client* dan *server* yang harus diletakkan pada mesin yang sama. Secara ringkas, umumnya *server* diletakkan pada satu sisi dalam LAN dan client pada sisi yang lain.



Gambar 6. Arsitektur *Client Server*

Dalam konteks basis data, *client* mengatur *interface* berfungsi sebagai *workstation* tempat menjalankan aplikasi basis data. *Client* menerima permintaan pemakai, memeriksa sintaks dan *generate* kebutuhan basis data dalam SQL atau bahasa yang lain. Kemudian meneruskan pesan ke *server*, menunggu *response* dan bentuk *response* untuk pemakai akhir. *Server* menerima dan memproses permintaan basis data kemudian mengembalikan hasil ke *client*.

Proses-proses ini melibatkan pemeriksaan autorisasi, jaminan integritas, pemeliharaan *data dictionary* dan mengerjakan *query* serta proses *update*. Selain itu juga menyediakan kontrol terhadap *concurrency* dan *recovery*.

Ada beberapa keuntungan jenis arsitektur ini adalah :

- Memungkinkan akses basis data yang besar
- Menaikkan kinerja
- Jika *client* dan *server* diletakkan pada komputer yang berbeda kemudian CPU yang berbeda dapat memproses aplikasi secara paralel. Hal ini mempermudah merubah mesin *server* jika hanya memproses basis data.
- Biaya untuk *hardware* dapat dikurangi
- Hanya *server* yang membutuhkan storage dan kekuatan proses yang cukup untuk menyimpan dan mengatur basis data
- Biaya komunikasi berkurang
- Aplikasi menyelesaikan bagian operasi pada *client* dan mengirimkan hanya bagian yang dibutuhkan untuk akses basis data melewati jaringan, menghasilkan data yang sedikit yang akan dikirim melewati jaringan
- Meningkatkan kekonsistenan
- *Server* dapat menangani pemeriksaan integrity sehingga batasan perlu didefinisikan dan validasi hanya di satu tempat, aplikasi program mengerjakan pemeriksaan sendiri
- Map ke arsitektur *open-system* dengan sangat alami

Berikut ini adalah ringkasan fungsi *client-server*

Client	***Server***
Mengatur *user interface*	Menerima dan memproses basis data yang diminta dari *client*
Menerima dan memeriksa sintaks input dari pemakai	Memeriksa autorisasi
Memproses aplikasi	Menjamin tidak terjadi pelanggaran terhadap *integrity constraint*
Generate permintaan basis data dan memindahkannya ke *server*	Melakukan *query*/pemrosesan *update* dan memindahkan response ke client
Memberikan *response* balik kepada pemakai	Memelihara *data dictionary*
	Menyediakan akses basis data secara bersamaan
	Menyediakan kontrol *recovery*

Data Dictionary

Data dictionary adalah tempat penyimpanan informasi yang menggambarkan data dalam basis data. *Data dictionary* biasa disebut juga dengan *metadata* atau *data mengenai data*. Modul pengontrol otorisasi menggunakan *data dictionary* untuk memeriksa apakah seorang pemakai perlu mempunyai wewenang.

Untuk mengerjakan pemeriksaan tersebut *data dictionary* menyimpan :

- nama-nama pemakai yang mempunyai wewenang untuk menggunakan DBMS
- nama-nama *data item* yang ada dalam basis data
- *data item* yang dapat diakses oleh pemakai dan jenis akses yang diijinkan, misalnya: *insert, update, delete* atau *read*

Sedangkan untuk memeriksa integritas data, *data dictionary* menyimpan :

- nama-nama *data item* dalam basis data
- jenis dan ukuran *data item*
- batasan untuk masing-masing *data item*

Sistem *data dictionary* dapat dibedakan atas sistem aktif dan pasif. Sistem aktif selalu konsisten dengan struktur basis data karena secara otomatis dikerjakan oleh sistem. Sebaliknya, sistem pasif tidak konsisten terhadap perubahan basis data yang dilakukan oleh pemakai.