**Materi 3 Perkuliahan ke-4**

***Database* dan Sistem Manajemen *Database***

**Tujuan pembelajaran**

Setelah mempelajari bab ini, anda diharapkan :

* Mengenal isi dari manajemen data – bagaimana data diorganisasikan, disimpan, diakses dan digunakan
* Mengerti bahwa aplikasi menentukan jenis pengolahan, yang selanjutnya menentukan jenis penyimpanan sekunder
* Mengenal bagaimana konsep *database* berkembang dan pengaruhnya pada pemrosesan komputer
* Mengetahui apa itu sistem manajemen *database*  *(database management system)* – DBMS, fungsinya dan cara penggunaannya
* Memiliki pengertian yang lebih baik mengenai peran pengelola *database*
* Mengetahui keuntungan dan kerugian penggunaan DBMS untuk mengelola *database*

**Pendahuluan**

Manajemen data merupakan bagian dari manajemen sumber daya informasi serta memastikan bahwa sumber daya data perusahaan mencerminkan secara akurat sistem fisik yang diwakilinya. Sumber daya data disimpan dalam penyimpanan sekunder, yang dapat mengambil bentuk berurutan *(sequential)* atau akses langsung *(direct access)*. Pita magnetik (*magnetic tape*) merupakan medium penyimpanan berurutan yang paling populer dan piringan magnetik (*magnetic disk*) merupakan cara utama mencapai akses langsung. Namun, suatu teknologi akses langsung baru, yaitu piringan optik (*optical disk*), semakin populer.

Sebelum era *database*, perusahaan mengalami keterbatasan dalam manajemen data mereka karena cara pengaturan data di penyimpanan sekunder. Usaha mula-mula untuk mengatasi kendala ini meliputi penyortiran dan penggabungan *file*, pemrograman komputer yang ekstensif untuk mencari dan mencocokkan catatan *file*, serta indeks *file* dan kaitan yang dibangun ke dalam catatan data. Konsep *database* dibangun di atas indeks dan kaitan untuk mencapai suatu hubungan logis antara beberapa *file*.

 Perangkat lunak yang mengelola *database* disebut sistem manajemen *database* *(database management system)* - DBMS. Semua DBMS memiliki suatu pengolah bahasa deskripsi data *(data description language processor)* yang digunakan untuk menciptakan *database*, serta suatu pengelola *database* yang menyediakan isi *database* bagi pemakai. Pemakai menggunakan manipulasi data dan *query language*. Orang yang bertanggung jawab atas *database* dan DBMS adalah penge-lola *database (database administrator),* atau disingkat DBA. DBMS menyediakan keuntungan yang nyata bagi perusahaan yang menggunakan komputer mereka sebagai suatu sistem informasi.

**Hirarki Data**

Perusahaan secara tradisional mengorganisasikan data mereka dalam suatu hirarki yang terdiri dari elemen, catatan *(record),* dan *file*.

* **Elemen data** *(Data element)* adalah unit data terkecil, tidak dapat lagi dibagi menjadi unit yang berarti. Dalam catatan gaji, elemen data berupa nama, nomor pegawai, nomor jaminan sosial, upah perjam, dan jumlah tanggungan.
* Satu tingkat hirarki yang lebih atas adalah catatan. Satu **catatan** *(record)* terdiri dari semua elemen data yang yang berhubungan dengan suatu objek atau kegi-atan tertentu. Misalnya, ada catatan yang menjelaskan tiap jenis persediaan dan tiap penjualan.
* Semua catatan yang sejenis disusun menjadi satu *file*. Satu *file* adalah suatu kumpulan catatan data (*data record*) yang berhubungan dengan suatu subyek tertentu. Misalnya,*file* Pesanan Pernbelim Terbuka menjelaskan pesanan pem-bei ian yang telah dipesan ke pemasok namun belum diterima.

Karena itu, hirarki data tradisional adalah:

• *File*

 • Catatan

• Elemen data

*File* adalah tingkat tertinggi dan elemen data adalah tingkat terendah.

**Manajemen Data**

Pada Bab 2 kita telah melihat bahwa manajemen sumber daya informasi (*information resources management)*, atau IRM, adalah keseluruhan usaha perusahaan untuk menciptakan dan memelihara sumber daya informasi. Karena data adalah suatu sumber daya, maka perlu dikelola, dan proses ini disebut manajemen data. **Manajemen data** adalah bagian dari manajemen sumber daya informasi yang mencakup semua kegiatan yang memastikan bahwa sumber daya data perusahaan akurat, mutakhir, aman dari gangguan dan tersedia bagi pemakai.

 **Kegiatan Manajemen Data**

Kegiatan manajemen data mencakup:

* **Pengumpulan data**  Data yang diperlukan dikumpulkan dan dicatat dalarn suatu formulir yang disebut dokumen sumber (*source document)* yang ber-fungsi sebagai *input* bagi sistem. Misalnya, data yang menjelaskan suatu penjualan dimasukkan ke formulir pesanan penjualan.
* **Integritas dan pengujian**  Data tersebut diperiksa untuk meyakinkan konsis-tensi dan akurasinya berdasarkan suatu peraturan dan kendala yang telah diten-tukan sebelumnya.
* **Penyimpanan** Data disimpan pada suatu medium seperti pita magnetik atau piringan magnetik.
* **Pemeliharaan** Data baru ditambahkan, data yang ada diubah, dan data yang tidak lagi diperlukan dihapus agar sumber daya data tetap mutakhir.
* **Keamanan** Data dijaga untuk mencegah penghancuran, kerusakan, atau penyalahgunaan.
* **Organisasi** Data disusun sedemikian rupa untuk memenuhi kebutuhan informasi pemakai.
* **Pengambilan** Data tersedia bagi pemakai.

 Sebelum era komputer, semua kegiatan dilakukan oleh pegawai administrasi, didukung oleh *punched-card* dan *keydriven machine* yang primitif. Sekarang, orang-orang masih diperlukan untuk banyak pengumpulan dan pengujian data, tetapi komputer telah mengambil sebagian bcsar tanggung jawab manajemen data.

**Penyimpanan Sekunder**

Semua komputer meliputi sejenis penyimpanan sekunder untuk melengkapi penyimpanan primer yang berada di dalam CPU. Dua jenis penyimpanan sekunder yang utama adalah berurutan dan akses langsung.

**Penyimpanan Berurutan**

**Penyimpanan berurutan** *(sequential storage)* adalah suatu organisasi atau pcnyusunan data di suatu medium penyimpanan yang terdiri dari satu catatan mengikuti satu catatan lain dalam suatu urutan tertentu. Misalnya, catatan pegawai disusun dalam urutan nomor pegawai. Saat penyimpanan berurutan digunakan, data pertama harus diproses pertama, data kedua diproses kedua, dan seterusnya sampai akhir *file* itu tercapai.

Sebagian media penyimpanan komputer hanya dapat memproses data yang disusun secara berurutan. *File* pita magnetik dan *punched card* yang digunakan oleh komputer pertama bersifat berurutan. *File* *punched card* telah menghilang, tetapi pita magnetik masih tetap digunakan.

**Penyimpanan Pita Magnetik**

Pita magnetik yang digunakan untuk menyinipan data komputer memiliki bentuk fisik yang sama dengan pita audio. Pita ini terdiri dari bahan plastik yang dilapis dengan suatu zat yang memungkinkan perekaman. Dalam hal pita komputer, data dicatat dalam bentuk bit-bit magnetik. Bit-bit itu mewakili tiap karakter disusun melintang terhadap lebar pita. Kerapatan pencatatan biasanya dapat mencapai 1.600 bit per inci (bpi).

Media pita magnetik pertama terdiri dari gulungan besar, tetapi sekarang *cartridge* lebih disukai karena dapat menyimpan lebih banyak data dan lebih sedikit memerlukan tempat. Sebagian bcsar sistem komputer *mainframe* memiliki satu atau beberapa ***tape unit*** dan ***tape drive*** yang membaca dan menulis data pita baik dalam bentuk gulungan atau *cartridge*. *Cartridge* tidak mengubah cara penggunaan pita magnetik; baik gulungan maupun *cartridge* menerapkan prinsip yang sama.

**Catatan Pita Magnetik**

Semua elemen data yang termasuk dalam suatu catatan dicatat mengikuti panjang pita seperti tampak dalam Gambar 10.1. Semua catatan membentuk *file*, seperti tampak pada Gambar 10.2. jarak-jarak yang kosong memisahkan catatan-catatan.

**Memperbarui *File* Pita Magnetik**

Sejumlah *file* yang menyediakan gambaran konseptual dari pf:rusahaan disebut ***file* *master***. Ada *File* *master* Persediaan, *File* *master* Pelanggan, *File* *master* Pegawai, dan sebagainya. Tiap *file* *master* berisi data tentang suatu subyek tertentu. *File* *master* diperbarui dengan data dari *file* transaksi. Suatu ***file* transaksi** berisi data yang menggambarkan kegiatan perusahaan seperti penjualan, pembelian, dan waktu kerja pegawai. Proses memperbarui *file* ini disebut **pemeliharaan *file***(*file* *maintenance*), dan meliputi penambahan catatan baru, penghapusan catatan, dan mengubah catatan. Saat suatufi/e *master* pita magnetik dipelihara, tidak praktis untuk menulis kembali catatan yang diperbarui ke area pita yang sama tempat catatan itu semula dibaca. Catatan yang diperbarui harus ditulis di atas pita lain. Karena alasan tersebut, pemeliharaan *file* dari suatu pita magnetik menghasilkan satu pita kedua, yang telah diperbarui. Gambar 10.3 menggambarkan proses tersebut. *File* asa! Disebut ***file* *master*** **lama** dan *file* yang telah diperbarui disebut ***file* *master* baru.** Selama pemeliharaan *file* dari suatu *file* *master* berurutan, catatan-catatan dalam *file* transaksi harus

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nomor pelanggan | Nomor pesanan | Tanggal pesanan  | Nomor wiraniaga | Nomor baranag | Penjelasan barang | jumlah |  |  |

 |

**Gambar 10.1** Suatu catatan pita magnetik

****

**Gambar 10.2** *file* pita magetik

dalarn urutan yang sama dengan catatan di *file master. file* transaksi dalam gambar tercatat di atas pita magnetik, namun medium *input* apa pun dapat digunakan.

**Penggunaan Pita Magnetik**

 Pita magnetik sangat cocok untuk digunakan sebagai **medium penyimpanan historis**. Perusahaan dapat menyimpan data pada pita dan menyimpan pita tersebut sebagai catatan kegiatan bisnis. Pita magnetik juga digunakan sebagai ***file backup*** dari *file master* yang tertulis pada alat penyimpanan akses langsung. *File backup* dapat digunakan jika terjadi sesuatu pada *file master* akses langsung.

 Pita magnetik dapat pula berfungsi sebagai **medium *input***. Sebagian *cash register* di toko eceran menyertakan unit pita magnetik yang mencatat data saat penjualan terjadi. Setelah toko tutup, komputer sentral, mungkin di kota lain, secara otomatis mengambil data dari pita itu.

 Akhirnya, pita magnetik dapat berfungsi sebagai **medium komunikasi** yang dapat dikirimkan melalui pos. Perusahaan yang lebih besar diharuskan menyerahkan data pajak mereka ke IRS secara demikian.

|  |
| --- |
| Memperbarui file master *File* transaksi *File* *master* lama*File* *master* baru |

**Gambar 10.3** Memperbaiki *file* pita magnetik

**Penyimpanan Akses Langsung**

**Penyimpanan akses langsung** (*direct access storage)* adalah suatu cara mengor-ganisasikan data yang memungkinkan catatan-catatan ditulis dan dibaca tanpa pen-carian secara berurutan. Unit perangkat keras yang memungkinkan ha1 ini disebut ***direct access storage device* (DASD).** DASD memiliki mekanisme membaca dan menulis yang dapat diarahkan ke lokasi mana pun dalam medium penyimpanan. Walau beberapa teknologi DASD telah dibuat, yang paling populer adalah piringan magnetik.

**Penyimpanan Piringan Magnetik**

Piringan *(disk)* yang digunakan untuk mencatat data komputer biasanya terbuat dari metal dan dilapisi bahan perekaman yang sama dengan pita magnetik. Beberapa piringan dapat disusun menjadi suatu **tumpukan piringan** *(disk stack)* vertikal, seperti tampak pada Gambar 10.4. Semua piringan itu ditempelkan ke satu poros tunggal, dan mereka berputar melalui satu mekanisme akses yang terdiri dari se-jumlah lengan akses *(access arm).* Pada ujung tiap lengan akses, terdapat *read/ write head* yang membaca data dari piringan dan menulis data ke pita. Seperti tarnpak pada gambar, data tercatat pada permukaan piringan dalam bentuk sejumlah jalur (*track).* **Jalur** adalah suatu pola melingkar dari bit-bit data. Mekanisme akses ditempatkan di atas jalur dan dapat membaca dari jalur atau menulis data ke atasnya saat piringan berputar. Tumpukan piringan dimasukkan dalam suatu ***disk drive*** atau ***disk unit.*** Konfigurasi *mainframe* besar atau komputer mini mencakup beberapa *disk drive* untuk memberikan kapasitas yang memadai. Tidak jarang 100 *drive* atau lebih terpasang untuk menyediakan trilyunan *byte* data. Dalam konfigurasi komputer mikro, *disk drive* berupa *diskette drive* dan *hard disk.*

|  |
| --- |
|  |

**Gambar 10.4** Tumpukan Piringan

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **2** | **0** | **9** |  | **0** | **7** |  | **0** | **0** | **3** |

Track number read/write head number record number  |

**Gambar 10.5** alamat DASD

**Membaca dan Menulis Data pada Piringan**

Saat data akan dibaca atau ditulis pada piringan, pertarna-tama perlu menempatkan mekanisme akses pada jalur yang tepat dan kemudian mengaktifkan *read/write head* yang tepat. Mekanisme akses itu harus memperoleh alamat ternpat catatan itu berada pada piringan. ***Disk address*** menentukan nomor jalur, nomor *read/write head*, serta biasanya nomor catatan pada jalur - catatan ], catatan 2, dan seterusnya. Gambar 10.5 menggambarkan alamat DASD.

**Pembuatan Alamat Catatan**

Ada tiga pendekatan dasar untuk menghasilkan alamat yang diperlukan DASD untuk mengakses suatu catatan. Pendekatan tersebut adalah direct, *hashing*, dan indexed sequential.

***Direct Addressing*** Pada *direct addressing,* kunci catatan *(record key*) berfungsi sebagai aIamat. **Kunci** *(key)* adalah elemen data yang mengidentifikasi catatan suatu *file*. Misalnya, kunci dari *File master* Pegawai adalah nomor pegawai. Jika pembuatan alamat DASD pada Gambar 10.5 menggunakan *direct addressing*, nomor karyawan itu menjadi 20907003. Nomor itu hanya perlu dipotong-pctong untuk menghasilkan komponen-komponen alamat yang diperlukan.

Penerjemahan langsung dari suatu kunci *file* menjadi alarnat penyimpanan di piringan sangat tidak Iazim dalarn aplikasi bisnis. Sangat sedikit *file* yang memiliki kunci-kunci yang cocok untuk digunakan sebagai suatu alamat.

 ***Hashing*** Menghitung alamat dari kunci itu dapat dilakukan. Alamat diu-bah oleh suatu algoritma yang disebut ***hashing scheme*** atau ***randomizing formula*** untuk menghasilkan alamat.

 Walau *hashing* melaksanakan jasa yang berharga, *hashing* memiliki masalah. Algoritma *hashing* dan set kunci-kunci *file* jarang menghasilkan satu set *disk address* yang secara seragam mengisi piringan. Mungkin terdapat banyak jalur yang tidak terisi atau tidak digunakan. Selain itu, algoritma itu mungkin menghasilkan *disk address* yang sama untuk lebih dari satu kunci. Masalah ini dipecahkan dengan *overflow area.* ***Overflow area*** adalah suatu bagian dari DASD yang dicadangkan untuk penyimpanan catatan-catatan yang kuncinya telah ditentukan. Kunci-kunci duplikat itu disebut ***synoym***. Mengambil catatan dari  *overflow area* menambah waktu akses.

***Indexed Sequential*** Jika suatu *file* diorganisasikan secara *indexed sequential*, catatan-catatan dicatat pada piringan secara berurutan. Sistem mengingat *disk address* aktual tempat satu subset (misalnya setiap beberapa ratus catatan) dari *file* disimpan. Kemudian kunci catatan ini dan dan *disk address-*nya ditempatkan dalam satu *file* atau tabel tersendiri yang disebut **indeks**. Subset dari catatan mung-kin ditentukan sebeltunnya menurut persentase; contohnya, tiap catatan keseratus akan disertakan dalam indeks. Untuk menentukan lokasi catatan ke-745, dimulai dengan mencari catatan ketujuh dalam indeks kemudian merneriksa piringan mulai dari catatan ke-700 dan terus hingga catatan ke-45 tercapai. Pada dasarnya ini seperti cara menggunakan Halaman Kuning *(Yellow Pages*) untuk mencari nomor telepon restoran *pizza*. Dari daftar isi terlihat bahwa daftar *pizza* mulai dari hala-man 80, karena itu Anda dapat langsung melewati halaman 1 sampai 79. Penggunaan indeks kadang-kadang lebih lambat dari *direct addressing* atau *hashing.* Namun, mengurangi tempat yang terbuang.

Sebelum era *database,* analis sistem dan programer disibukkan dengan menetapkan *file addressing* dan struktur indeks yang optimal. Sekarang hanya DBA (pengelola *database)* yang menghabiskan waktunya untuk hal-hal tersebut, sehingga para spesialis informasi lain dapat berfokus pada masalah-masalah bisnis.

**Memperbarui *File* DASD**

 Gambar 10.6 menggambarkan cara perneliharaan *file* dilakukan pada *file* DASD. Pada Langkah 1, data transaksi dimasukkan dengan menggunakan *online keydriven device* seperti terminal CRT. Tetapi medium input apa pun dapat digunakan. Selain itu, data transaksi tidak perlu dalam suatu urutan tertentu. Karena mekanisme akses dapat diarahkan ke catatan mana pun untuk membaca atau menulis pada lokasi tersebut,*file* kedua tidak perlu dibuat, seperti halnya pita magnetik. Pada Langkah 2, catatan yang telah diperbarui ditulis ulang ke lokasi asalnya.

**Penggunaan DASD**

 DASD adalah ***medium file master*** yang baik. *File-file* dapat diperbarui saat transaksi terjadi, menghasilkan catatan kegiatan pertisahaan yang mutakhir. Penggunaan lain yang populer adalah sebagai **medium penyimpanan sementara** untuk me-nampung data semi-terproses. Misalnya, data dapat ditransfer dari satu program ke program lain dalarn bentuk piringan. DASD dapat pula digunakan sebagai **medium *input*** dengan cara yang sama seperti pita magnetik. DASD tidak baik untuk penyimpanan historis, karena tumpukan piringan lebih mahal dari pada gulungan pita atau *cartridge.*

|  |
| --- |
| DatatransaksiMemasukkan data transaksiLangkah 1Memperbarui file masterMaster file Langkah 2 |

**Gambar 10.6** Memperbarui Fle DASD

**Piringan Optik**

Hampir sepanjang era komputer, piringan magnetik telah terbukti tidak terkalahkan sebagai DASD. Berbagai teknologi lain telah dicoba; termasulk *magnetic drums, bins of magnetic tape loops, magnetic cards, magnetic bubbles, dan silicon chips.*hanya *magnetic drums* yang berdampak nyata, digunakan untuk akses cepat oleh *mainframe.*

Teknologi DASD baru yang berpeluang terbaik untuk ditetapkanskebajuggaai medium penyimpanan sekunder adalah piringan optik. Suatu **piringan optik** disebut ***laser disk*** atau ***compact disk (CD),*** menggambarkan data dengan kombinasi goresan-goresan halus di atas permukaan piringan yang diciptakan oleh sinar laser. Goresan-goresan ini berupa parit, gelembung, atau perubahan keadaan *crys-talline*. Sinar laser berintensitas rendah digunakan untuk membaca goresan-goresan tersebut. Keunggulan utama dari piringan optik adalah kapasitasnya yang besar. Piringan optik 51/4- inci dapat memuat antara 440 *megabytes (*MB), atau jutaan *byte*, dan 1 *gigabyte (*GB), atau milyaran *byte,* dengan rata-rata 650 MB. Kapasitas ini sekitar 540 kali juinlah data yang dapat disimpan dalam satu disket, jika kapa-sitas satu disket dianggap 1,5 MB. Kapasitas satu piringan ini dapat lebih diting-katkan dengan menumpukkan piringan-piringan optik itu dalam suatu alat otomatis penyusun piringan yang disebut ***jukebox.***

Piringan optik pertama memiliki kemampuan menulis sekali; disebut **WORM *(write-once, read-many)****.* Data hanya dapat direkam sekali pada piringan, tetapi data terscbut dapat dibaca berulang kali sekehendak kita. Istilah **CD-ROM *(compact disk*** *-* ***read-only memory)*** digunakan untuk menggambarkan teknologi WORM yang telah populer untuk digunakan dengan komputer mikro. WORM ideal bagi penyimpanan historis, karena data tidak dapat diubah setelah direkam.

Pada akhir 1980-an, tersedia piringan optik yang dapat dihapus dan ditulis kembali berulang kali. Piringan jenis ini disebut ***rewritable disk***. *Rewritable disk* dapat digunakan seperti piringan magnetik, walau waktu aksesnya lebih lambat. Sebagian pemasok piringan optik sekarang menawarkan *drive* yang dapat menangani baik WORM maupun *rewritable disk. Drive* ini disebut ***drive multifungsi.***

Suatu waktu, piringan optik akan menggantikan pita magnetik sebagai medium penyimpanan historis yang disukai. Kunci agar piringan optik dapat menggantikan piringan magnetik adalah waktu akses. Sebelum peningkatan kecepatan itu tercapai, perusahaan akan menggunakan piringan optik bersama-sama dengan piringan magnetik. Data yang kurang aktif akan disimpan dalam unit-unit piringan optik, dan data yang lebih aktif akan disimpan dalam penyimpanan piringan magnetik untuk akses yang cepat.

**Hubungan Penyimpanan Sekunder dengan Pemrosesan**

Terdapat dua cara utama untuk rnengolah data - pengolahan *batch* dan pengolahan *online*. **Pengolahan *batch*** mencakup pengumpulan transaksi dan pemrosesan semuanya sekaligus, dalam *batch*. **Pengolahan *online*** mencakup pengolahan transaksi satu persatu, kadang saat terjadinya transaksi itu. Karena pengolahan *online* berorientasi transaksi, istilah **pemrosesan transaksi** sering digunakan. Kita akan menggunakan istilah pengolahan *online*, karena pemrosesan transaksi juga dapat berarti pengolahan data.

Apa yang menentukan jenis pemrosesan? Aplikasi perusahaan yang menentukan. iika pemrosesan tidak perlu dilakukan saat terjadinya transaksi, pengolahan *batch* dapat dikerjakan, dan pita magnetik atau piringan magnetik dapat digunakan.

Sistem gaji adalah contoh aplikasi pengolahan *batch* yang baik. Sebaliknya, jika terdapat a1asan untuk memproses transaksi satu persatu, pengolahan *online* mertipakan pilihannya dan suatu jenis DASD diperlukan.

**Pengolahan *batch***

Gambar 10.7 adalah suatu bagan arus sistem yang menggambarkan pengolahan *batch*. Tujuan dari sistem ini adalah memperbarui tiga *file master* Persediaan, Piutang, dan Analisis Penjualan. Perusahaan biasanya memperbarui *file* batch mereka secara harian, yang disebut **siklus harian**.

 *File* pertama yang diperbarui adalah *file* Persediaan, yang disusun dalam urutan nomor jenis barang. Nomor jenis barang menjadi kuncinya.

 Karena catatan-catatan transaksi harus dalam urutan yang sama dengan *file* *master,* catatan-catatan itu disortir dalam Langkah 1, dan *file* Persediaan diperbarui dalam Langkah 2. Langkah 3 dan 4 memperbarui *file* Piutang, dan Langkah 5 dan 6 mengerjakan hal yang sama pada *file* Analisis Penjualan.

Pengolahan *batch* adalah cara paling efisien untuk menggunakan perangkat keras komputer karena merupakan proses jalur perakitan *(assembly-line).* Gerakan dan tempat penyimpanan yang terbuang hanya sedikit. Kelemahan utama dari pengolahan *batch* adalah kenyataan bahwa *file* baru menjadi mutakhir setelah dilakukan siklus harian. lni berarti manajemen tidak selalu memiliki informasi paling mutakhir yang menggambarkan sistem

**Pengolahan *online***

 Pengolahan *online* dikembangkan untuk mengatasi masaiah *file* yang ketinggalan jaman. Terobosan teknologi yang memungkinkan pengolahan *online* adalah penyim-panan piringan magnetik.

Gambar 10.8 menggambarkan pendekatan *online* untuk memperbarui tiga *file* yang sama seperti pada contoh *batch*. Setiap transaksi diproses pada **semua** *file* *master* yang berkaitan sementara data transaksi berada dalam penyimpanan prirner. Catatan Persediaan yang sesuai dibaca pada penyimpanan primer, diperbarui dengan data transaksi, kemudian ditulis kembali pada DASD. Kemudian catatan Piutang diperbarui dengan cara yang sama, dilanjutkan dengan catatan Analisis Penjualan. Ketiga *file* DASD tersebut diperbarui sebelum transaksi selanjutnya dimasukkan.

**Sistem *Realtime***

Isitlah *realtime* sering digunakan berhubungan dengan sistem komputer. Anda mungkin mendengar seseorang berkata, "Kami memiliki sistern realtirae," atau, "Sistem kami beroperasi secara *realtime*." **Sistem *realtime*** adalah suatu sistem yang mengendalikan sistem fisik. Sistem ini mengharuskan komputer berespon cepat pada status sistem fisik.

 Misalkan, Anda ingin menulis cek untuk membayar pembelian di suatu toko serba ada dan petugasnya menanyakan nomor kartu identitas Anda. Petugas itu memasukkan nomor tersebut ke dalam *cash register* yang dihubungkan dengan kom-puter, dan komputer melakukan pemeriksaan kredit. Jika kredit Anda baik., Anda diijinkan untuk melakukan pembelian; jika tidak, Anda pergi dengan tangan kosong. Komputer menentukan apakah suatu penjualan terjadi. Komputer mengen-dalikan sistem fisik.

Sistem *realtime* adalah bentuk khusus dari sistem *online*. Sistem *online* menyediakan sumber daya konseptual yang mutakbir, dan sistem *realtime* memperluas kemampuan tersebut dengan menggunakan sumber daya konsetual untuk menentukan operasi dari sistem fisik.

|  |
| --- |
| *Batched**Transaction*dataSortir menjadi urutan nomor jenis barangLangkah 1 Data transaksi yang di sortir File persediaan lamaMemperbarui file persediaanLangkah 2File persediaan baruData transaksiSortir menjadi urutan nomor pelangganLangkah 3Data transaksi yang disortir File piutang dagang lama Memperbarui file piutang dagangLangkah 4File piutang dagang baru Data transaksi Sortir menjadi urutan nomor wiraniagaLangkah 5File transaksi yang disortirFile analisis penjualan lamaMemperbarui file analisis penjualanLangkah 6File analisis penjualan baru |

**Gambar 10.7**  Pengolahan Batch

|  |
| --- |
| Persediaan Piutang AnalisispenjualanMemperbarui Memasukkan satu catatan transaksi |

**Gambar 10.8** pengolahan online

**Era Sebelum *Database***

Era penggunaan komputer yang ada sebelum konsep *database* ditandai dengan pengulangan data *(data redundancy),* ketergantungan data *(data dependency),* dan kepemilikan data yang tersebar *(diffused data ownership).*

**Pengulangan Data**

Selama paruh pertama abad ini, saat perusahaan memproses data secara manual dan dengan *punched card* dan *keydriven machine*, semua keputusan yang berhubungan dengan data dibuat sccara individual, tanpa memperhatikan keseluruhan rencana data *(data plan).* Contohnya, saat tiap sistem pengolahan data dirancang,*file* data *input* yang diperlukan oleh sistem tersebut diciptakan tanpa mempertimbangkan bagaimana data tersebut mempengaruhi sistem latin. Mungkin banyak, atau bahkan seluruh, data dalam suatu *file* baru telah terdapat dalam *file* yang sudah ada. Hasilnya adalah **pengulangan data** atau duplikasi.

 Tempat yang terbuang karena catatan-catatan data diulang dalam banyak *file* cukup buruk, tetapi akibat dari duplikasi lebih buruk lagi. Satu masalahnya karena tidak ada standar dalam penamaan elemen-elemen data. Contohnya, nama *file* "Persediaan" pada satu sistem persediaan mengacu pada persediaan yang tersirnpan dalam gudang. Sistem lain dalam perusahaan yang sama mengartikan "Persediaan" sebagai semua persediaan yang ada, termasuk barang yang baru tiba di dok penerimaan. Manajemen yang meminta informasi dapat memperoleh jawaban-jawaban yang berbeda, tergantung sistem apa yang digunakan untuk menghastikan laporan.

Masalah lain adalah tidak adanya keselarasan dalam memperbarui data du-plitikat. Berbagai salinan data sering diperbarui pada selang waktu yang bertainan. Satu *file* mungkin diperbarui setiap hari, yang kin mingguan, dan *file* yang lain lagi bulanan. Laporan yang didasarkan pada satu *file* mungkin bertentangan dengan laporan yang didasarkan pada *file* lain, dan pemakai tidak menyadari perbedaannya.

**Ketergantungan Data**

**Ketergantungan data** mengacu pada penggabungan yang erat antara spesifikasi data dan program komputer. Karakteristik data seperti panjang *file*, panjang catatan, dan lain-lain dikodekan ke dalam tiap program yang mengakses data tersebut. Situasi ini berarti bahwa setiap kali suatu *file* berubah, semua program yang mengakses *file* itu harus dimodifikasi. Dalam beberapa kasus semua program yang mengakses *file* yang berubah itu bahkan tidak teridentifikasti sampai program-program itu gagal karena mereka tidak mampu menangani format yang baru.

**Kepemilikan Data yang Tersebar**

 Sangat sulit untuk menyeragamkan kepemilikan dan standarisasi data bagi suatu sistem informasi. Kenyataannya, tiap programer yang diakses oleh programnya. Situasi pada tahun 1950-an dan awal 1960-an ini mungkin merupakan penyebab gagalnya sebagian besar sistem informasi manajemen pada akhir 1960- an. Para spesialis informasi tampaknya tidak mampu menghasilkan sistem-sistem yang memberikan informasi yang konsisten, akurat, dan dapat diandalkan.

**Kebangkitan Era *Database***

Para spesialis informasi mencari cara memecahkan permasalahan yang disebabkan oleh cara pengorganisasian data secara dan usaha mereka mengarah ke organisasi logis. **Organisasi logis** *(logical organization)* mengintegrasikan data dari beberapa lokasi fisik yang berbeda dan merupakan cara pemakai melihat data. Contohnya, seorang manajer melihat bahwa semua informasi pada suatu laporan memi-liki integrasi logis walau datanya mungkin diambil dari beberapa *file* yang terpisah secara fisik. **Organisasi fisik** *(physical organization)*, sebaliknya, merupakan cara *komputer* melihat data – sebagai *file-file* yang terpisah.

Tugas spesialis informasi adalah menyediakan organisasi logis yang diperlukan oleh pemakai dalam kendala dari organisasi fisik. Pemakai sering menginginkan data dari satu *file* tunggal untuk disusun dalam suatu urutan yang berbeda dari urutan dalam medium penyimpanan, mereka juga ingin mengumpulkan data beberapa *file*. Berbagai teknik dikembangkan untuk memenuhi kebutuhan-kebutuhan tersebut dengan menggunakan integrasi logis.

**Integrasi Logis dalam Satu *File***

 Dua pendekatan memungkinkan catatan-catatan pada satu *file* dipilih berdasarkan karakteristiknya daripada berdasarkan kuncinya.. Pendekatan ini disebut *inverted file* dan *linked list*. Keduanya membutuhkan DASD.

 ***Inverted File*** adalah suatu *file* yang disimpan dalam suatu urutan tertentu, tetapi suatu indeks yang menyertainya memungkinkan catatan-catatan dari *file* itu dipilih dalam urutan yang berbeda.

 *Inverted file* dirancang untuk memecahkan jenis masalah berupa permintaan manajer atas laporan yang mendaftarkan hanya catatan-catatan tertentu dalam suatu *file*. Misalnya, manajer penjualan mungkin ingin melihat daftar dari penjualan yang dilakukan oleh Wiraniaga 23. Tiap catatan dalam *file* harus diperiksa oleh komputer untuk menentukan apakah itu merupakan catatan Wiraniaga 23. Jika ya, catatan itu dipilih dan digunakan untuk mencetak laporan. Suatu *file* yang berisi ribuan catatan mungkin hanya menyertakan belasan catatan mengenai Wiraniaga 23, tetapi *setiap* catatan dalam *file* itu harus dibaca. Pendekatan ini sangat tidak efisien.

Para spesialis informasi sadar mereka dapat menciptakan indeks bagi *File master* Wiraniaga yang mengidentifikasi semua catatan dari setiap wiraniaga. ***Inverted file index*** seperti itu, juga sering disebut **indeks sekunder**, digambarkan dalam Gambar 10.9. Saat diperlukan, indeks tersebut dibaca ke dalam penyimpanan primer dan program menelusuri kolom nomor wiraniaga untuk mencari Wiraniaga 23. Saat baris tersebut ditemukan program memeriksa ke arah kanan dan mengambil nomor-nomor catatan yang diperlukan. Pada contoh, terdapat satu catatan untuk tiap pelanggan Wiraniaga 23 yang telah melakukan pembelian. Catatan yang diperlukan dapat dipilih tanpa membaca seluruh *file.*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nomorwiraniaga | Namawiraniaga | Pelanggan 1 | Pelaggan 2 | Pelanggan 3 | Pelanggan n |
| 16 |  | 17042 | 21096 |  |  |
| 20 |  | 41854 |  |  |  |
| 23 |  | 23694 | 25410 | 30102 | 30111 |
| 31 |  | 310002 |  |  |  |
| 56 |  | 34107 | 13109 |  |  |
| 92 |  | 20842 |  |  |  |
| 98 |  | 61634 |  |  |  |
| 104 |  | 10974 |  |  |  |
| 110 |  | 16342 | 64210 | 51263 | 41782 |

 |

**Gambar 10.9** *inverted file index*

***Linked List*** Teknik Iain dapat mencapai hasil yang sama. Misalkan manajer yang sama menginginkan laporan yang sama, tetapi spesialis informasi ingin menghindari penggunaan indeks untuk mendapatkan akses yang lebih cepat. Suatu *field* tersendiri, yaitu *salesperson link*, ditambahkan pada tiap catatan dalam *File master* Wiraniaga, seperti tampak pada Gambar 10.10. *Field* tersebut berisi ***link***, atau ***pointer***, yang menghubungkan semua catatan dari setiap wiraniaga. *File* yang berisi *link field* disebut ***linked list.***

Yang tampak dalam gambar hanya *link* untuk Wiraniaga 23. Program memilih catatan dengan menelusuri tiap catatan dalam *file* hingga catatan pertama dari Wiraniaga 23 ditemukan. *Link field* dalam catatan pertama disebut ***head***, dan mengidentifikasi catatan selanjutnya dari Wiraniaga 23. Catatan selanjutnya diambil dan *link field*-nya mengidentifikasi catatan ketiga. Proses ini berlanjut sampai catatan terakhir terambil. *Link field* dalam catatan terakhir berisi kode khusus yang mengidentifikasikan dirinya sebagai ***tail***.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NomorPelanggan |  |  |  | Nomor wiraniaga | Link wiraniaga |
| 22504 |  |  |  |  |  |
| 23694 |  |  |  | 23 | 25410 |
| 24782 |  |  |  |  |  |
| 25409 |  |  |  |  |  |
| 25401 |  |  |  | 23 | 30102 |
| 26713 |  |  |  |  |  |
| 28914 |  |  |  |  |  |
| 30004 |  |  |  | 23 |  |
| 300102 |  |  |  | 23 | 30111 |
| 30111 |  |  |  |  | \* |
| 30417 |  |  |  |  |  |
| 31715 |  |  |  |  |  |
|  Catatan data |

 |

**Gambar 10.10** *Linked list*

*Inverted file* dan *linked list* memberikan suatu cara untuk mengintegrasikan secara logis catatan-catatan yang tersebar secara fisik dalam satu *file*. Juga terdapat kebutuhan untuk mencapai hasil yang sama antara beberapa *file*.

 **Integrasi Logis antara Beberapa *file***

Pada pertengahan tahun 1960-an General Electric memodifikasi COBOL untuk membuat data dapat diambil dari beberapa *file*. *Link* digunakan untuk saling meng-hubungkan catatan-catatan dalam *satu file* dengan catatan-catatan yang berhubungan secara logis di *file-file* lain. Sistem GE ini dinamai IDS, yaitu *Integrated Data Store*, dan merupakan langkah maju pertama menuju satu *database* terintegrasi dari beberapa *file*.

**Konsep *Database***

***Database***adalah suatu koleksi data komputer yang terintegrasi, diorganisasikan dan disimpan dalam suatu cara yang memudahkan pengambilan kembali. DASD harus digunakan. Gambar 10.11 menunjukkan bahwa banyak *file* perusahaan dapat terintegrasi secara logis. Integrasi Iogis dari catatan-catatan dalam banyak *file* ini disebut **konsep *database***. Garis-garis dalam gambar mewakili integrasi logis.

 Dua tujuan utama dari konsep *database* adalah meminimumkan pengulangan dan mencapai independensi data. **Independensi data** adalah kernampuan untuk membuat perubahan dalam struktur data tanpa membuat perubahan pada program yang memproses data. Independensi data dicapai dengan menempatkan spesifikasi dalam tabel dan kamus yang terpisah secara fisik dari program. Program mengacu pada tabel untuk mengakses data. Perubahan pada struktur data hanya dilakukan sekali, yaitu dalam tabel.

|  |
| --- |
|  **File pelanggan****File statistik penjualan****Fie piutang dagang****file wiraniaga** **File hutang dagang****File pemasok****File buku besar****File pesanan pembelian****File persediaan****File pembeli** |

**Gambar 10.11** *Database* Terdiri Dari Satu Atau Beberapa *File*

Saat perusahaan mengadopsi konsep *database*, hirarki data menjadi:

 • *Database*

 • *File*

 • Catatan

• Elemen data

 *File-file* tersendiri dapat tetap ada, mewakili komponen-komponen utama dari *database;* namun, organisasi fisik dari data tidak menghambat pemakai. Tersedia berbagai cara untuk mengintegrasikan isi dari *file-file* yang memiliki hubungan logis.

**Struktur *Database***

 Integrasi logis *file* dapat dicapai secara eksplisit atau secara implisit.

**Hubungan Eksplisit**

*Inverted index* dan *link field* menetapkan hubungan eksplisit antara data yang terintegrasi secara logis dalam *file yang sama*. 1ndeks dan *field* ada secara fisik dan harus disatukan ke dalam *file* saat dibuat. Jika tidak, permintaan manajer atas informasi yang terintegrasi secara logis hanya dapat dipenuhi dengan pemrograman dan penyortiran khusus yang memakan waktu dan mahal.

 Satu pendekatan untuk menetapkan hubungan eksplisit antara catatan-catatan dari *beberapa file* adalah dengan menyusun catatan-catatan tersebut dalam suatu hirarki. Ini disebut **struktur hirarkis**. Dalam struktur seperti ini, setiap catatan pada satu tingkat dapat dihubungkan ke berbagai catatan yang setingkat lebih rendah. Suatu catatan yang memiliki anak catatan disebut ***parent,*** dan anak catatan itu disebut ***children***. Diagram pada bagian atas Gambar 10.12 memberikan contoh dari struktur hirarkis.

Suatu ciri penting dari Gambar 10.12 adalah *link field* yang menetapkan hubungan-hubungan. *Link* ini digambarkan oleh garis tipis yang menghubungkan elemen-elemen data dari berbagai catatan di bagian bawah. Setelah Anda mengambil catatan seorang wiraniaga (misalkan Wiraniaga 23), *link* pada catatan ini akan mengarahkan Anda ke catatan lain yang berhubungan secara logis dengan wiraniaga itu. *Link field* dalam catatan kedua mengarah ke catatan ketiga, dan seterusnya, menciptakan suatu reaksi berantai melalui seluruh kelompok *file*.

 Walau struktur hirarkis merupakan langkah raksasa menuiu penghapusan kendala-kendala fisik, penggunaan hubungan eksplisit memiliki kelemahan. Kelornpok *file* yang harus terintegrasi secara logis perlu diidentifikasi *sebelum* dibuat *database*. Hal ini membatasi manajer untuk membuat **permintaan ad. hoc** - permintaan khusus untuk mengkombinasikan informasi yang tidak ditentukan sebelumnya.

**Hubungan Implisit**

 Pada awal 1970-an Edgar F. Codd dan C.J. Date, keduanya dari IBM tetapi bekerja secara terpisah, mengembangkan suatu pendekatan untuk menetapkan hubungan antar catatan yang tidak harus dinyatakan secara eksplisit. *Link field* khusus tidak perlu disertakan dalam catatan. Pendekatan Codd dan Date telah dinamakan **struktur relasional**, dan menggunakan **hubungan implisit**, yaitu hubungan yang dapat dinyatakan secara tidak langsung dari catatan data yang telah ada.

|  |
| --- |
| Catatan piutang dagangCatatan statistik penjualanCatatan pelangganCatatan wiraniaga **CATATAN WIRANIAGA**Nomor wiraniaga Nama wiraniagaNomor kantor penjualan **CATATAN PELANGGAN**Dll. Nomor pelanggan*Link* ke catatan statistik penjualan Nama pelanggan*Link*  ke catatan pelanggan Nomor wiranaiaga Dll.**CATATAN STATISTIK PENJUALAN** *Link*  ke catatan wiraniagaNomor wiraniaga *Link* ke catatan piutang dagangNomor jenis barangNomor pelanggan **CATATAN PIUTANG DAGANG**Jumlah penjualan Nomor pelanggan*Link* ke catatan wiraniaga Nomor faktur Tanggal faktur Nilai faktur *Link* ke catatan pelanggan |

**Gambar 10.12** Hubungan Eksplisit Antar *File*

Misalkan kita ingin menggunakan dua tabel data untuk mempersiapkan suatu laporan. Data dalam *database* relasional ada dalam bentuk tabel-tabel yang disebut *flat files*. ***Flat file*** adalah suatu penyusunan data dua dimensi dalam kolom-kolom dan baris-baris.

Kedua tabel data tampak dalam Tabel 10.1 Kita menginginkan laporan men-daftarkan para wiraniaga dalam Wilayah 1, dengan menunjukkan nomor-nomor wiraniaga dan nama mereka. Kedua tabel diperlukan. Tabel A menyediakan cara mengidentifikasi catatan-catatan untuk Wilayah 1, dan Tabel B menyediakan nama-nama wiraniaga. Hubungan implisit ditetapkan oleh *field* nomor wiraniaga di kedua tabel menyatukan nomor-nomor wilayah dalam Tabel A dengan nama-nama wiraniaga dalam Tabel B.

TABEL 10.1

Contoh Dua Flat *File*

|  |
| --- |
| Tabel A |
| No. Wiraniaga | Wilayah |
| 112128153159162166 | 132112 |

|  |
| --- |
| Tabel B |
| No. Wiraniaga | Nama |
| 112128153159162166 | AdamsWinklerHouseFrancisWillisGroveton  |

 Keuntungan utama dari struktur relasional bagi CBIS adalah fleksibilitas yang ditawarkannya dalam rancangan dan penggunaan *database*. Pemakai dan spesialis informasi dibebaskan dari keharusan mengidentifikasi semua informasi yang diper-lukan sebelum menciptakan *database*.

**Perangkat Lunak *Database***

Perangkat Iunak yang menetapkan dan memelihara integrasi logis antar *file*, baik eksplisit maupun implisit, disebut **sistem manajemen** ***database*** *(database management system)* - **DBMS**. IDS dari General Electric adalah contoh pertarnanya, dan kemudian diikuti oleh usaha serupa dari pemasok perangkat keras dan perangkat lunak lain. Contoh DBMS yang menggunakan struktur hirarkis adalah IMS *(Information Management System)* dari IBM, dan System 2000 dari Intel. Sistem-sistem tersebut, sebagian besar masih digunakan, sangat mahal, biayanya sekitar $100.000.

Gelombang selanjutnya dari inovasi DBMS menampilkan perangkat lunak relasional, dan sejumlah paket awal ditujukan bagi pemakai *mainframe*. SQL/DS *(Structured Query Language/Data System)* dan QBE *(Query By Example)* dari IBM, dan ORACLE dari Relational Software Inc., semuanya diterima dengan baik. Pada saat yang hampir bersamaan, sekitar tahun 1980, pemasok perangkat lunak mulai mengembangkan paket-paket DBMS berskala Iebih kecil untuk pasar komputer mikro. DBMS berbasis komputer mikro pertama yang sangat berpengaruh adalah dBASE II, yang dipasarkan oleh Ashton-Tate (sekarang merupakan bagian dari Borland International, Inc.)

 Selama tahun-tahun terakhir ini, pengembangan DBMS berfokus pada pasar komputer mikro dan telah menerapkan struktur relasional.

**Menciptakan *Database***

Proses menciptakan *database* mencakup tiga langkah utarna. Pertama, Anda menentukan data yang dibutuhkan. Kedua, Anda menjelaskan data tersebut. Ketiga, memasukkan data ke dalam *database.*

**Menentukan Kebutuhan Data**

Definisi dari kebutuhan data adalah langkah kunci mencapai CBIS. Ada dua pendekatan dasar berorientasi pemakai dan model perusahaan.

**Pendekatan Berorientasi Masalah** Saat perusahaan mengambil pendekatan berorientasi masalah, mereka mengikuti urutan langkah-langkah.yang digambarkan dalam Gambar 10.13. Pertama, *masalah didefinisikan*. Kemudian *keputusan* yang diperlukan untuk memecahkan masalah didefinisikan, dan untuk tiap keputusan didefinisikan *informasi* yang diperlukan. Selanjutnya, *pemrosesan* yang diperlukan untuk menghasilkan informasi ditentukan, dan akhirnya *data* yang diperlukan oleh pemrosesan ditetapkan.

**Pendekatan Model Perusahaan** Walau pendekatan berorientasi masalah memungkinkan kebutuhan data dari tiap sistem didefinisikan secara logis, kelemahannya adalah sukar mengaitkan data suatu sistem ke data sistem lain. Sistem-sistem tidak mudah berbagi data. Kelemahan ini diatasi dengan menentukan seluruh kebutuhan data perusahaan dan kemudian menyimpan data tersebut dalam *database*. Usaha pengembangan sistem selanjutnya kemudian mengambil data yang telah ada dalam *database*,

 Hal ini sebenarnya gagasan di belakang pendekatan model data perusahaan. Nama yang diberikan untuk deskripsi dari semua data perusahaan adalah **model data perusahaan**. Ini merupakan proses *top-down,* yang dimulai saat perencanaan strategis sumber daya informasi, seperti digambarkan dalam Gambar 1 0.14.

Satu cara yang baik untuk mendokumentasikan model data perusahaan adalah dengan menggunakan diagram hubungan entitas *(entitiy-relationship diagram)*, atau ERD. ERD akan dijelaskan dalam Modul Teknis 1.

**Menjelaskan Data**

 Setelah elemen-elemen data yang diperlukan ditentukan, mereka dijelaskan dalam bentuk kamus data *(data dictionary )*. **Kamus data** adalah suatu ensiklopedi dari informasi mengenai tiap elemen data. Kamus data akan dijelaskan dalam Modul Teknis II.

**Sistem Kamus data** Kamus data dapat berupa kertas atau *file* komputer. Jika berupa *file*, perangkat lunak khusus diperlukan untuk menciptakan dan memeliharanya, serta mempersiapkannya untuk digunakan. Perangkat lunak tersebut disebut **sistem kamus data** *(data dictionary system)*, atau **DDS**. DDS dapat diperoleh sebagai paket perangkat lunak terpisah atau sebagai modul-modul di dalam sistem seperti DBMS dan peralatan *computer-aided software engineering* (CASE).

**Data Description Language** Setelah kamus data diciptakan, penjelasannya harus dimasukkan dalam DBMS. DBMS menyertakan ***data description language (DDL)*** yang digunakan untuk menjelaskan data. Gambar 10.15 menunjukkan bagaimana DDL menggunakan kamus data untuk menghasilkan skema.

|  |
| --- |
| Spesifikasi dataMenentukan kebutuhan dataMenentukan pemrosesan yang diperlukanMenjelaskan kebutuhan informasiMengidentifikasi keputusan yang diperlukanMendefinisikan masalah |

**Gambar 10.13** Kebutuhan Data Dapat Didefinisikan Dengan Pendeketan Berorientasi

|  |  |
| --- | --- |
|

|  |
| --- |
| Perencanaan strategis sumber daya informasiMeciptakan model data perusahaan |

Mengembangkan *database**database*Model data perusahaan |

**Gambar 10.14** Kebutuhan Data Dapat Didefinisikan Dengan Menciptakan Model Data Perusahaan

**Skema** bukanlah data itu sendiri tetapi *penjelasan* dari data, Skema biasanya menentukan atribut atau karakteristik data seperti:

 • Nama elemen data

 • Alias (nama lain yang digunakan untuk elemen data yang sama)

 • Jenis data (angka, abjad, dan lain-lain)

 • Jumlah posisi

 • Jumlah posisi desimal (hanya untuk data angka)

• Berbagai aturan integritas data

Istilah **subskema** digunakan untuk subset dari keseluruhan deskripsi yang berhubungan dengan pemakai tertentu. Tiap pemakai rnemiliki kebutuhan data khusus, dan deskripsi elemen-elemen data tersebut diwakili oleh satu atau beberapa subskema.

Kombinasi dari skema, subskema dan kamus data mempersiapkan sistem bagi independensi data dan mengurangi pengulangan dengan mengarahkan semua pemakai *database* untuk melihat *database* sentral. Tidak pertu dibuat *file* tersendiri untuk tiap pemakai.

|  |
| --- |
| Memasukkan data kamusSkema*Data description language (DLL)*Kamus dataLangkah 1Langkah 2 |

**Gambar 10.15** Menjelaskan Isi *Database*

**Memasukkan Data**

 Setelah skema dan subskema diciptakan, data dapat dimasukkan ke dalam *database*. Hal ini dapat dilaksanakan dengan mentik data langsung ke dalam DBMS, membaca data dari pita atau piringan, atau men-*scan* data secara optis. Data siap untuk digunakan setelah berada dalam *database*.

**Menggunakan *database***

**Pemakai *Database*** dapat berupa *orang* atau *program aplikasi*. Orang biasanya menggunakan *database* dari terminal dan mengambil data dan informasi dengan menggunakan *query language*. ***Query*** adalah permintaan informasi dari *database*, dan *query language* adalah bahasa khusus yang *user-frinedly* yang memungkinkan kornputer menjawab *query*.

 Saat suatu program aplikasi seperti program gaji mengambil data dari *database* atau menyimpan data ke dalamnya, ***data manipulatIon language* (DML)** khusus digunakan. Pernyataan-pernyataan DML ditanamkan dalam program aplikasi di titik-titik yang perlu.

**Peristiwa DBMS**

Peristiwa yang terjadi saat program aplikasi mengambil data dari *database* diilustrasikan dalam Gambar 10.16. Pada Langkah 1 DML menentukan bagi DBMS data apa yang diperlukan. Pada Langkah 2 DBMS memeriksa skema dan subskema untuk menguji bahwa data ada dalam *database* dan bahwa program aplikasi berhak menggunakannya. Pada Langkah 3 DBMS meneruskan permintaan data ke.sistem operasi, yang pada Langkah 4 mengambil data dan memasukkannya ke dalam area penyimpanan *buffer* khusus dalam penyimpanan primer. Data itu kemudian ditransfer ke dalam area *input* program aplikasi pada Langkah 5. DBMS mengembalikan pengendalian ke program aplikasi pada Langkah 6, dan prograrn aplikasi menggunakan data pada Langkah 7.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Peyimpanan primer

|  |
| --- |
| Sistem Operasi |
|  Skema Subskema |
| Lang. 3 lang. 2 Langkah 1 DBMS program aplikasilangkah 6  lang.7  |
| Area penyimpanan *buffer*Langkah 5 | Area *input,output*Dan kerja pemakai |

langkah 4*database*langkah 4 Arus data Sinyal pengendali |

**Gambar 10.16** Peristiwa DBMS

 Seri peristiwa yang sama terjadi saat *query language* digunakan. Dalam hal ini *query language* adalah subset dari DBMS dan informasi yang diambil ditarnpilkan pada alat *output* pemakai.

**Suatu Model DBMS**

Model yang menunjukkan elemen-elemen utama DBMS digambarkan dalam Gambar 10.17. Elemen-elemen tersebut meliputi *data descrtption language processor, performance statistics processor*, modul *backup/recovery*, dan manajer *database.*

***Data Description Language Processor***

 ***Data description language processor*** mengubah kamus data menjadi skerna *database*. Ini merupakan DDL yang telzh dijelaskan sebelumnya. Semua DBMS memi-liki DDL.

***Performance Stalistics Processor***

***Perforrnance statistics processor*** memelihara statistik yang mengidentifikasi data apa yang sedang digunakan, siapa yang menggunakannya, kapan digunakan. dan seterusnya. Statistik ini digunakan dalam mengeiola *database*. DBMS berbasis komputer mikro biasanya tidak menyertakan elemen ini.

|  |  |  |
| --- | --- | --- |
| Data *description language procesor*Penjelasan *database* (skema)Log transaksiStatistik kinerja*database*Performance statistics processor

|  |
| --- |
| Manajer *database**data manipulation language (DML)**Query language* |
| Program aplikasi |

Modul backup/recoveryPermintaan informasiinformasiStatistik kinerja |

**Gambar 10.17** suatu model DBMS

**Modul *Backup*/Recovery**

Secara periodik, misalnya harian, dibuat suatu *backup* *copy* dari *database*. Saat transaksi-transaksi selanjutnya diproses terhadap *file master*, suatu **log transaksi** menyimpan catatan dari semua perubahan. Kemudian, saat terjadi bencana, *log* transaksi diproses terhadap *backup database* untuk merekonstruksi *database* tersebut. Modul *backup/recovery* menyelesaikan rekonstruksi ini, tetapi modul ini jarang terdapat dalam DBMS untuk komputer mikro.

**Manajer *Database***

***Manajer database*** adalah elemen paling penting karena menangani permintaan data para pemakai. *Query language* dan DML adalah bagian dari manajer *database*. Manajer *database* juga menghasilkan statistik kinerja yang diproses oleh *performance statistics processor,* dan *log* transaksi diproses oleh modul *backup/recovery*. Semua DBMS menyertakan elemen ini.

 Manajer *database* adalah satu-satunya elemen DBMS yang ***main-rnemory resident***. Artinya tinggal dalam penyimpanan primer seterusnya. Elemen-elemen lain adalah ***transient routines***, disimpan dalam penyimpanan sekunder dan dibawa ke penyimpanan primer saat diperlukan.

**Pengelola *Database***

Seorang spesialis informasi yang bertanggung jawab atas *database* disebut **pengelola *database*** *(database administrator)* atau **DBA.** Tugas DBA terbagi dalam empat area utama: perencanaan, penerapan, operasi dan keamanan.

 • **Perencanaan *database*** meliputi bekerja sama dengan manajer untuk mendefinisikan skema perusahaan dan dengan pemakai untuk mendefinisikan subskema mereka. Selain itu, DBA berperan penting dalam memilih DBMS.

 • **Penerapan *database*** terdiri dari menciptakan *database* yang sesuai dengan spe-sifikasi dari DBMS yang dipilih, serta menetapkan dan menegakkan kebijakan dan prosedur penggunaan *database*.

 • **Operasi *database*** mencakup menawarkan program-program pendidikan bagi pemakai *database*, dan menyediakan bantuan saat diperlukan.

 • **Keamanan *database*** meliputi pemantauan kegiatan *database* dengan menggu-nakan statisitik yang disediakan DBMS. Selain itu, DBA memastikan bahwa *database* tetap aman. Masalah penting keamanan *database* akan dibahas dalam Bab 22.

Perusahaan-perusahaan besar memiliki beberapa spesialis *database*, yang di-pimpin oleh seorang manajemen DBA.

**Menempatkan *Database* dan DBMS dalam Perspektif**

DBMS memungkinkan untuk menciptakan *database* dalam penyimpanan akses langsung komputer, memelihara isinya, dan menyediakan isi tersebut bagi pemakai tanpa pemrograman khusus yang mahal.

Saat perusahaan atau pemakai individu memutuskan apakah akan menggu-nakan, keuntungan dan kerugiannya harus dipertimbangkan.

**Keuntungan DBMS**

 DBMS memungkinkan perusahaan maupun pemakai individu untuk:

 • **Mengurangi pengulangan data.** Jumlah total *file* dikurangi dengan menghapus *file*-*file* duplikat. Juga hanya terdapat sedikit data yang sama di beberapa *file*.

 • **Mencapai independensi data**. Spesifikasi data disimpan dalam skema daripada dalam tiap program aplikasi. Perubahan dapat dibuat pada struktur data tanpa mempengaruhi prograrn yang mengakses data.

 • **Mengintegrasikan data dari beberapa *file***. Saat *file* dibentuk sehingga me-nyediakan kaitan logis, organisasi fisik tidak lagi menjadi kendala.

• **Mengambil data dan Informasi secara cepat.** Hubungan-hubungan logis dan DN1L serta query language memungkinkan pernakai mengambil data dalam hitungan detik atau menit, yang sebeiumnya mungkin memerlukan beberapa jam atau hari.

 • **Meningkatkan keamanan.** Baik DBMS *mainframe* maupun komputer mikro dapat menyertakan beberapa lapis keamanan seperti kata sandi (password), directory pemakai, dan bahasa sandi (encryption). Data yang dikelola oleh DBMS juga lebih aman daripada data lain dalam perusahaan.

 Namun, DBMS tidak bebas dari kerugian

**Kerugian DBMS**

 Keputusan untuk menggunakan DBMS mengikat perusahaan atau pemakai untuk:

 • **Memperoleh perangkat lunak vang mahal**. DBMS *mainframe* masih sangat mahal. DBMS berbasis komputer mikro, walau biayanya hanya beberapa ratus dolar, dapat menggambarkan suatu organisasi kecil secara berarti.

 • **Memperoleh konfigurasi perangkat keras yang besar**. DBMS sering memerlukan kapasitas penyimpanan primer dan sekunder yang lebih besar daripada yang diperlukan oleh program aplikasi lain. Juga, kemudahan yang dibuat oleh DBMS dalam mengambil informasi mendorong, lebih banyak terminal pemakai yang disertakan dalarn konfigurasi daripada jika sebaliknya.

 • **Mempekerjakan dan mempertahankan staf DBA**. DBMS memerlukan pengetahuan khusus agar dapat memanfaatkan kemaimpuannya secara penuh. Pengetahuan khusus ini paling baik diberikan oleh para pengelola *database* (DBA).

 *Database* terkomputerisasi maupun DBMS bukanlah prasyarat mutlak untuk peme-cahan masalah. Namun, mereka memberikan dasar-dasar penggunaan komputer sebagai suatu sistem informasi bagi para spesialis informasi dan pemakai.

**Daftar Pustaka**

Appleton, Daniel S. " The Modern Data Dictionary." *Datamation* 33. (March 1. 1987):66-68.

Choobineh, Joobin; Mannino, Michael V.; dan Tseng, Veronica P. "A Form-Based Approach for *Database* Analysis Design." *Communications of the ACM* 35 (February 1992): 108-120.

 Cohen, Ethvard 1.; King, Gary M.; dan Brady, James T. `Storage Hierarchies*. IBM Systems Journal* 28 (Number 1, 1989): 62,76.

Davis, Dwight B. "Optical Drives' New Reading Habits." *Datamation* 37 (September 15, 1991): 71-73.

Davis, Leila. "New Uses For Optical Storage." *Datamation* 38 (March 15, 1992): 78ff

DeLoach, Allan. "The Path to Writing Efficient Queries in SQL/DS." *Database Programming & Design* 1 (1987): 26-32.

Eddison, Elizabeth B. "How to Plan Build Your Own *Database*." *DATABASE* 11 (June 1988): 15ff

Francis, Bob. "Optical Shines for Near Line Storage." *Datamation* 39 (November 15, 1993); 79-80.

 Gelb, Jack P. "System-managed Storage." *IBM Systems Journal* 28 (Number I. 1989): 77-103.

 Goodhue, Dale L.; Kirsch, Laurie J.; Quillard, Judith A.; dan Wybo, Michael D. "Strategic Data Planning: Lessons From the Field." *MIS Quarterly* 16 (Nlarch 1992): 11-34.

Madnick, Stuart E., dan Wang, YANG. Ricliard. "Evolution towards Strategic Applications of *Database* through Composite Information Systems."*Journal of Management Information System.*" 5 (Fall 1988): 5-22.

Navathe, Shamkant B. "Evolution of Data Modeling for *Database*." *Communications of the ACM* 35 (September 1992): 112-123.

Rapaport, Matthew. "Interview: Dr. Edgar F. Codd." *Database Programnzing & Design* 1 (February 1988); 60-65.

Ricciuti, Mike. "How To Clean Up Your Dirty Data." *Datamation 39* (August 15, 1993): 51-52.

Scheer, August-Wilhelm, dan Hars, Alexer. "Extending Data Modeling to Cover the Whole Enterprise." *Communications of the ACM* 35 (September 1992): 166-172.

The, Lee. " CD-ROM Reaches for Critical Mass*." Datamation* 38 (April 15, 1992): 47*ff*