ORGANISASI FILE

ORGANISASI FILE

- File diorganisasi (disusun) berdasarkan urutan-urutan record-record.
- Record-record dipetakan ke dalam blok-blok dalam harddisk
- blok berukuran tetap, 1 blok berisi lebih dari 1 record
- JENIS RECORD BERDASARKAN PANJANGNYA :
  - FIXED LENGTH RECORD
  - VARIABLE LENGTH RECORD

RECORD 1	0411500005	Ahmad Zaki	Cipondoh
RECORD 2	0422500025	Sinta	Kebayoran Lama
RECORD 3	0422500035	Indra Gunawan	Cipulir
RECORD 4	0433500058	Bekti Sularso	Cidodol
RECORD 5	0444500057	Tini Lestari	Cileduk

ORGANISASI FILE

FIXED LENGTH RECORD

- Record yang panjangnya tetap
- Misal : untuk membuat record mahasiswa

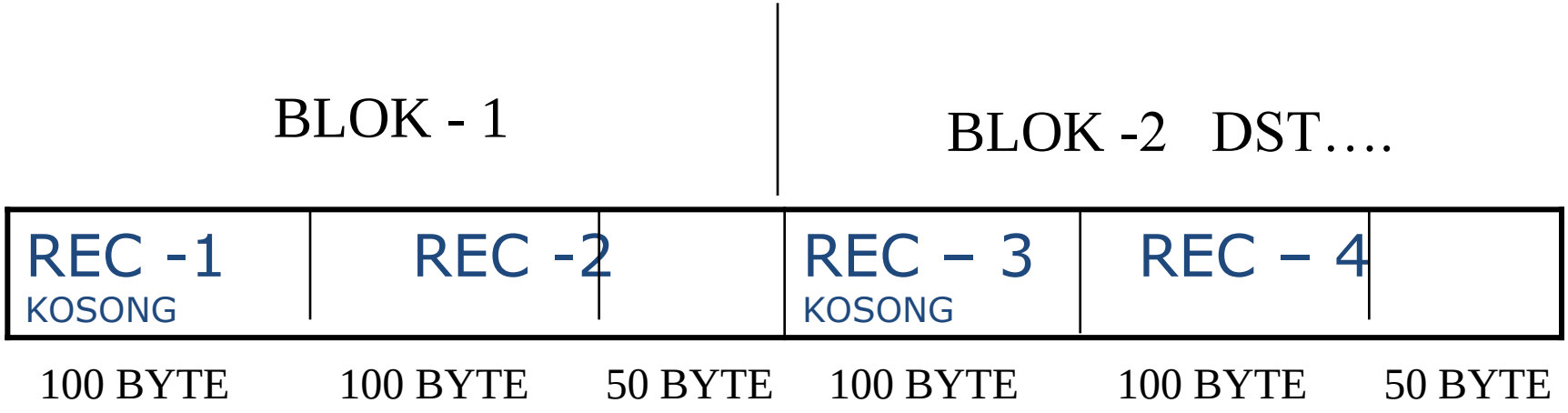
```
TYPE MAHASISWA = RECORD
                    NIM : CHAR(10);
                    NAMA : CHAR(40);
                    ALAMAT : CHAR(50);
                END
```

- Tiap karakter menyimpan 1 byte, maka record ke 1 untuk data mahasiswa di atas akan menyimpan 100 byte, kemudian 100 byte untuk record yang kedua dan seterusnya.

ORGANISASI FILE

- Penempatan record pada blok disebut blocking
- Metode blocking untuk record berukuran tetap adalah fixed length blocking
- Misal :

  1 block dapat menyimpan 250 byte, jika 1 record panjangnya 100 byte maka BLOCKING SBB:



ORGANISASI FILE

- Kelebihan fixed length record :

  mudah dalam pemrograman, karena untuk menyisipkan atau menghapus record mudah karena panjang recordnya sama

- Kekurangan fixed length record :

  boros tempat penyimpanan

ORGANISASI FILE

VARIABLE LENGTH RECORD

- Record yang panjangnya tidak tetap
- Misal : untuk membuat record mahasiswa

```
TYPE MAHASISWA = RECORD
                     NIM : VARCHAR(10);
                     NAMA : VARCHAR(40);
                     ALAMAT : VARCHAR(50);
                 END
```

- Panjang tiap record berbeda-beda tergantung dari isi dari masing-masing record
- Penempatan record dalam blok tergantung dari panjang record
- Metode blocking untuk record berukuran tidak tetap ada dua :
  - Variable length spanned blocking
  - Variable length unspanned blocking

ORGANISASI FILE

RECORD 1	0411500005	Ahmad Zaki	Cipondoh
RECORD 2	0422500025	Sinta	Kebayoran Lama
RECORD 3	0422500035	Indra Gunawan	Cipulir
RECORD 4	0433500058	Bekti Sularso	Cidodol
RECORD 5	0444500057	Tini Lestari	Cileduk

Panjang record 1 = 28 byte

Panjang record 2 = 29 byte

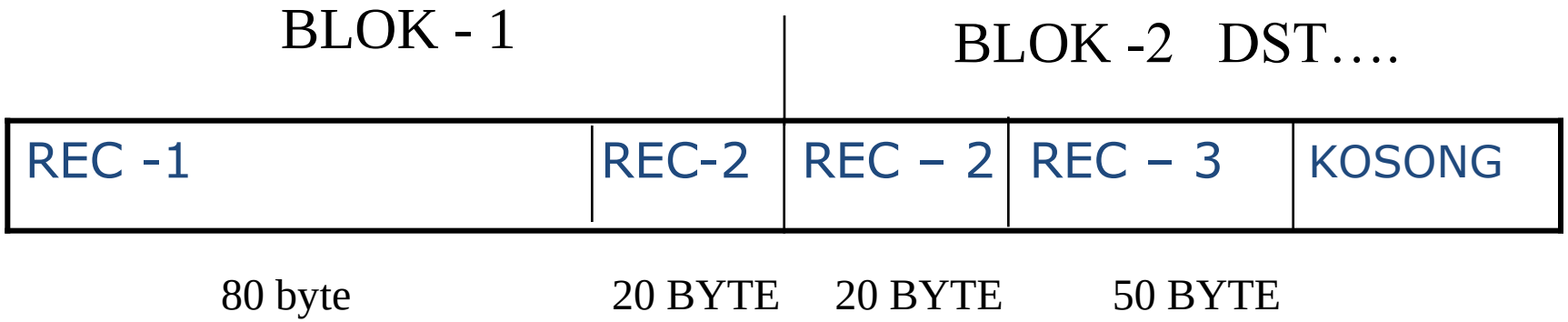
Panjang record 3 = 30 byte dst…

ORGANISASI FILE

- Variable length spanned blocking :

  Record ditempatkan dalam blok sesuai dengan ukurannya jika panjang record tidak dapat dimuat dalam 1 blok maka record dapat muat dalam blok terpisah (1 record dapat dipotong)

- Misal : 1 blok dapat memuat 100 byte.
  - PANJANG RECORD 1 = 80 BYTE
  - PANJANG RECORD 2 = 40 BYTE
  - PANJANG RECORD 3 = 50 BYTE

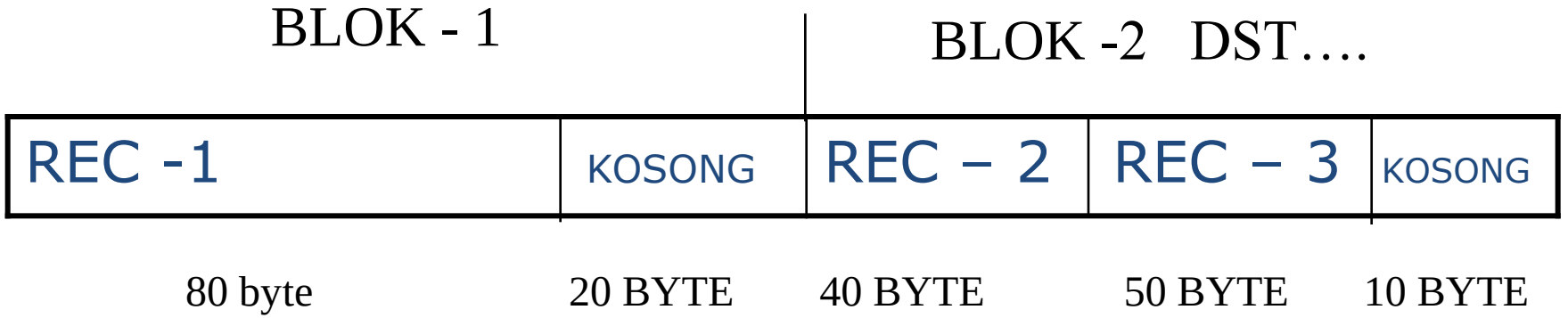


ORGANISASI FILE

- Variable length unspanned blocking :

  record ditempatkan dalam blok sesuai dengan ukurannya jika panjang record tidak dapat dimuat dalam 1 blok maka record dapat muat dalam blok terpisah (1 record tidak boleh dipotong)

- Misal : 1 blok dapat memuat 100 byte.
  - PANJANG RECORD 1 = 80 BYTE
  - PANJANG RECORD 2 = 40 BYTE
  - PANJANG RECORD 3 = 50 BYTE



ORGANISASI FILE

- Kelebihan variable length record :

  hemat tempat penyimpanan

- Kekurangan variable length record :

  sulit digunakan dalam pemrograman, karena panjang record berbeda maka tiap akhir record digunakan symbol end of record yang menandakan record sudah berakhir

ORGANISASI FILE

- Record tersusun dalam sebuah file
- Beberapa cara pengorganisasian (penyusunan) record dalam sebuah file adalah sebagai berikut :
  - ORGANISASI FILE HEAP
    - Tiap record ditempatkan di mana saja di dalam file selama masih terdapat tempat untuk record tersebut
    - Tidak ada pengurutan dalam record
  - ORGANISASI FILE SEKUENTIAL
    - Penempatan Record Diurutkan Sekuential Berdasarkan Sebuah Key
  - ORGANISASI FILE HASHING
    - Fungsi hash yang menghitung beberapa attribut dari record. Hasil dari fungsi akan menempatkan lokasi dari record tersebut

ORGANISASI FILE

Beberapa konsep dasar

- Field
  Satuan informasi terkecil yang menyusun record

- Record
  Kumpulan dari field yang berhubungan satu sama lain

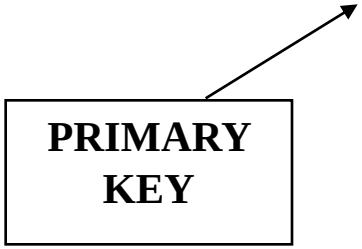
- File
  Kumpulan dari record-record

- Basis data
  Kumpulan file yang digunakan oleh program aplikasi serta membentuk hubungan tertentu di antara record-record di file-file tersebut

ORGANISASI FILE

- Key

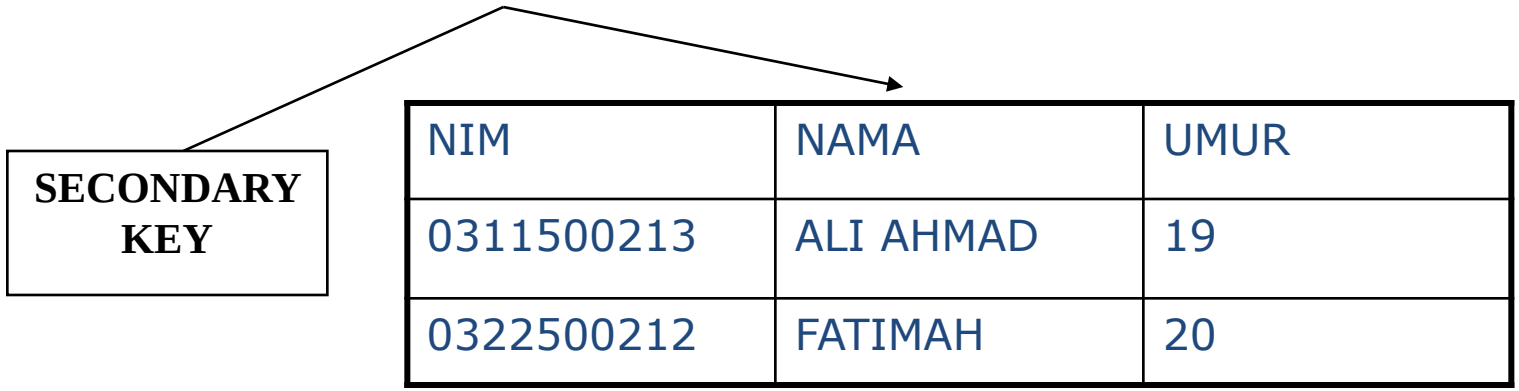
  Elemen record yang dipakai untuk menemukan record tersebut pada waktu akses
- Jenis-jenis key:
  - Primary key
  - Field yang mengidentifikasikan sebuah record dalam file
  - Bersifat unik



NIM	NAMA	UMUR
0311500213	ALI AHMAD	19
0322500212	FATIMAH	20

ORGANISASI FILE

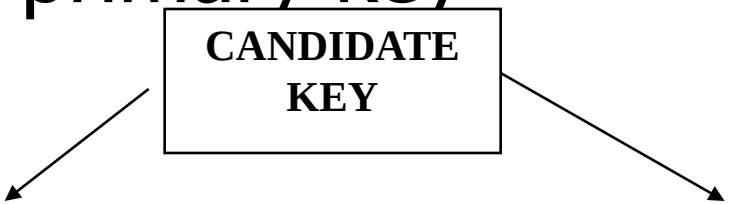
- Secondary key
  - field yang mengidentifikasikan sebuah record dalam file
  - tidak bersifat unik



NIM	NAMA	UMUR
0311500213	ALI AHMAD	19
0322500212	FATIMAH	20

ORGANISASI FILE

- Candidate key
  Field-field yang bisa dipilih (dipakai) menjadi primary key

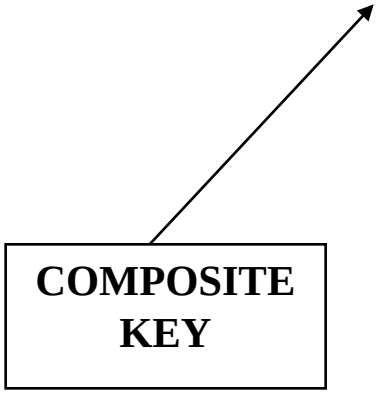


NIM	NAMA	NO. KWITANSI	JUMLAH
0311500012	AHMAD	KW-001	3000000
0322500023	RINA	KW-002	5000000

ORGANISASI FILE

- Composite key
  Primary key yang dibentuk dari beberapa field

HARI	RUANG	MATA KULIAH
SELASA	4.2.2	JARINGAN KOMPUTER
SELASA	4.2.1	SISTEM BASIS DATA 1
RABU	4.2.2	PANCASILA

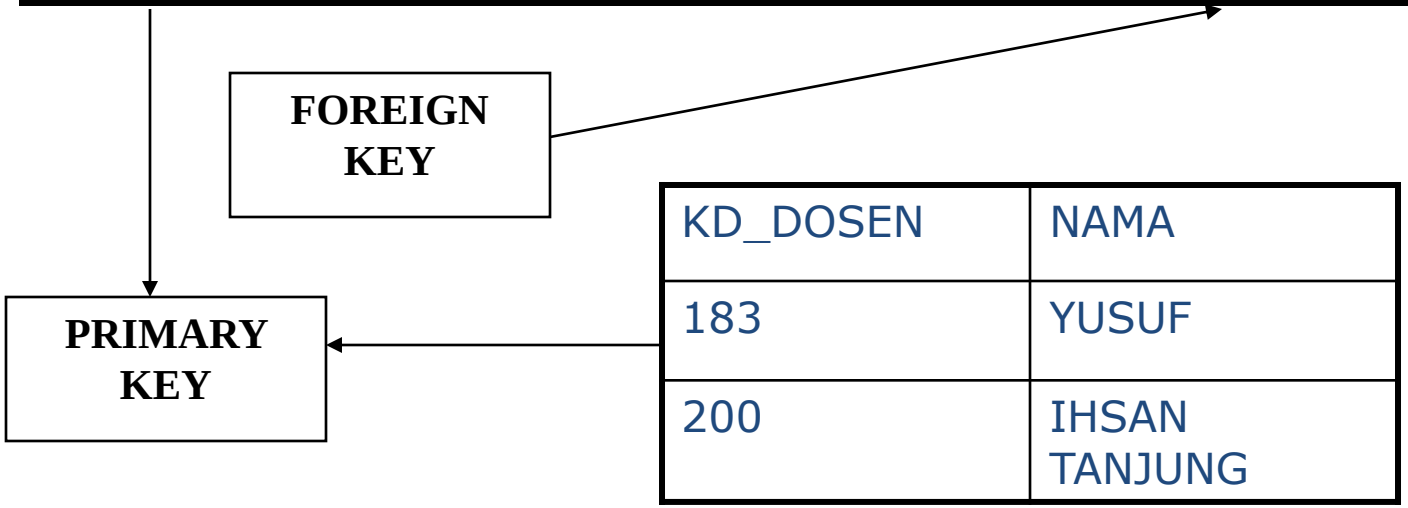


ORGANISASI FILE

- Foreign key

  Field yang bukan key, tetapi adalah key pada file yang lain

KD_MK	NM_MK	SKS	KD_DOSEN
K82	SBD-1	2	183
K29	JARKOM	3	200



ORGANISASI FILE

- File sekuential didesign untuk efisiensi pemrosesan rekord pada saat pengurutan berdasarkan beberapa key
- File dengan data yang tersusun dalam suatu urutan tertentu
- Tiap Record Mempunyai Field Yang Sama & Dengan Susunan Yang Sama

ORGANISASI FILE

STRUKTUR FILE

- Untuk memungkinkan record tersusun secara urut perlu ditentukan key dari tiap record

- Pembacaan secara serial (satu persatu) sesuai dengan urutan keynya disebut pembacaan secara sequential

Nip	Nama	Pekerjaan
000021	Abu Bakar	Manajer
000032	Fatimah	Sekretaris
000042	Asma	Presiden direktur

ORGANISASI FILE SEKUENTIAL

Insert sebuah record

- insert berarti menambahkan sebuah data baru ke dalam file
- insert pada ujung akhir sebuah file, hanyalah menambah banyaknya data waktu yang dibutuhkan kecil

1	2	3	4	5	6	7	8	9	…
A	B	C	D	E	F	…	…	…	…

INSERT X PADA AKHIR RECORD

1	2	3	4	5	6	7	8	9	…
A	B	C	D	E	F	X	…	…	…

ORGANISASI FILE SEKUENTIAL

Insert sebuah record

- Insert berarti menambahkan sebuah data baru ke dalam file
- Insert pada ujung akhir sebuah file, hanyalah menambah banyaknya data waktu yang dibutuhkan kecil

1	2	3	4	5	6	7	8	9	…
A	B	C	D	E	F	…	…	…	…

INSERT X PADA AKHIR RECORD

1	2	3	4	5	6	7	8	9	…
A	B	C	D	E	F	X	…	…	…

ORGANISASI FILE SEKUENTIAL

Insert ditengah file mengakibatkan pergeseran ataupun perubahan struktur data yang tidak sederhana

1	2	3	4	5	6	7	8	9	…
A	B	C	D	E	F	…	…	…	…

INSERT X PADA RECORD KE 3

1	2	3	4	5	6	7	8	9	…
A	B	X	C	D	E	F	…	…	…

→ RECORD KE-3 DST BERGESER

ORGANISASI FILE SEKUENTIAL

DELETE SEBUAH RECORD

- Menghapus sebuah record
- mencari lokasi data & menghapus isinya, agar bisa dipakai oleh data yang lain
- setelah itu dilakukan pergeseran ataupun pengaturan struktur data kembali

1	2	3	4	5	6	7	8	9	…
A	B	C	D	E	F	…	…	…	…

→ HAPUS

BILA RECORD D DIHAPUS, MAKA AKAN TERJADI PEMBACAAN DAN PENULISAN ULANG RECORD E, F, DST

1	2	3	4	5	6	7	8	9	…
A	B	C	E	F	…	…	…	…	…

ORGANISASI FILE SEKUENTIAL

Kadangkala delete dilakukan dengan hanya memberi tanda saja (tombstone / flag), tanpa dilakukan penghapusan ataupun pengaturan struktur datanya

1	2	3	4	5	6	7	8	9	…
A	B	C	D	E	F	…	…	…	…

→ HAPUS

1	2	3	4	5	6	7	8	9	…
A	B	C	*	E	F	…	…	…	…

→ record yang sudah dihapus "Delete"

ORGANISASI FILE INDEX

KONSEP DASAR

- Sebuah File Akan Terus Diakses Untuk Mencari Datanya (Fetch Data) Untuk Kemudian Data Tersebut Diambil Dari File (Retrieve Data)
- Untuk mencari data pada sebuah tabel dapat dilakukan secara sekuential. Namun cara pencarian ini akan memakan waktu lama jika file terdiri dari banyak record

SEKUENTIAL SEARCH MULAI DARI RECORD-1 ……. DST

0411500005	Ahmad Zaki	Cipondoh
0422500025	Sinta	Kebayoran Lama
0422500035	Indra Gunawan	Cipulir
0433500058	Bekti Sularso	Cidodol
0444500057	Tini Lestari	Cileduk

ORGANISASI FILE INDEX

- **PADA DASARNYA TERDAPAT 2 MACAM PENGURUTAN :**
  - **Pengurutan secara indeks**
    Berdasarkan urutan dari sebuah nilai
  - **Pengurutan secara hash**
    Berdasarkan fungsi hash yang digunakan

- **TIAP PENGURUTAN MEMPERHATIKAN FAKTOR-FAKTOR, YAITU :**
  - **TIPE AKSES**
    Tipe akses dalam mencari record. Yang lebih dipilih tentunya yang lebih efisien
  - **WAKTU AKSES**
    Waktu yang dibutuhkan untuk menemukan sebuah record
  - **WAKTU HAPUS**
    Waktu yang dibutuhkan untuk menghapus sebuah item
  - **RUANG SPASI**
    Ruang tambahan yang diminta oleh stuktur index.

INDEX YANG TERURUT

- Untuk mengatasi pencarian record dalam sebuah file secara acak, dapat digunakan struktur index**.**
- Tiap struktur index dihubungkan sesuai dengan key yang dicari (search key)
- Sebuah file dapat mempunyai beberapa file indeks, dengan search key yang ber beda-beda.
- Jika search key yang dipakai adalah primary key pada sebuah file master maka file index yang dibuat disebut primary indeks
- Jika Search Key Yang Dipakai Adalah Bukan Primary Key Pada Sebuah File Master Maka File Index Yang Dibuat Disebut Secondary Indeks

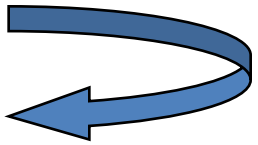
INDEX YANG TERURUT

- File index terdiri dari nomor record serta field yang digunakan sebagai search key

- Sebelum Mencari Data Pada File Master, Data Dicari Terlebih Dahulu Pada File Index, Jika Data Tersebut Ada, Maka File Index Langsung Menunjuk Lokasi Dari Data Tersebut Pada File Master

INDEX YANG TERURUT

NIM	NO. REC
0233500058	1
0322500025	2
0411500005	3
0422500035	4
0444500057	5



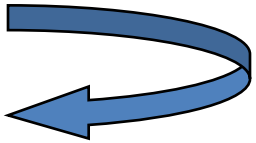
FILE INDEX YANG BERJENIS PRIMARY INDEKS

FILE MASTER DENGAN NIM SEBAGAI PK

NO. REC	NIM	NAMA	ALAMAT
1	0233500058	Ahmad Zaki	Cipondoh
2	0322500025	Sinta	Kebayoran Lama
3	0411500005	Indra Gunawan	Cipulir
4	0422500035	Bekti Sularso	Cidodol
5	0444500057	Tini Lestari	Cileduk

INDEX YANG TERURUT

NAMA	NO. REC
Ahmad Zaki	1
Bekti Sularso	4
Indra Gunawan	3
Sinta	2
Tini Lestari	5



FILE INDEX YANG BERJENIS SECONDARY INDEKS

FILE MASTER DENGAN NIM SEBAGAI PK

NO. REC	NIM	NAMA	ALAMAT
1	0411500005	Ahmad Zaki	Cipondoh
2	0322500025	Sinta	Kebayoran Lama
3	0422500035	Indra Gunawan	Cipulir
4	0233500058	Bekti Sularso	Cidodol
5	0444500057	Tini Lestari	Cileduk

PRIMARY INDEKS

- Pada file indeks yang menggunakan primary indeks, semua file master telah diurutkan berdasarkan primary key.

  File indeks juga telah diurutkan berdasarkan primary key

  semua file yang ada di atas disebut file indeks sekuential.

- Record indeks terdiri dari search key dan pointer yang menunjuk pada satu atau lebih record.

  Pointer terdiri dari identifier dari blok tempat record berada dalam disk

- ADA 2 TIPE PENGURUTAN INDEKS YANG DIGUNAKAN
  - Dense index

    Semua nilai dari search key muncul pada file index
  - Sparse index

    Hanya sebagian dari nilai search key yang muncul pada file index

DENSE INDEKS DAN SPARSE INDEKS

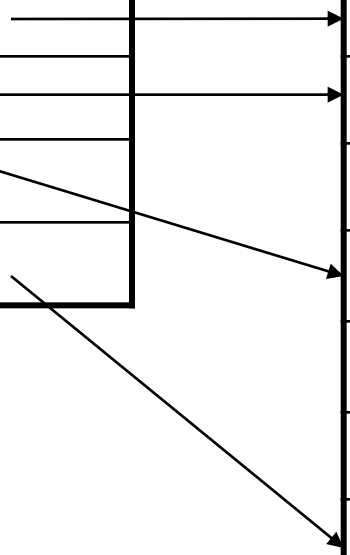
DENSE INDEX

FILE INDEX

CABANG	POINTER
BOGOR	
DAGO	
MALANG	
PADANG	

FILE MASTER

NO. REK	CABANG	JUMLAH
A-217	BOGOR	750
A-099	DAGO	450
A-101	DAGO	500
A-065	MALANG	300
A-135	MALANG	300
A-215	MALANG	700
A-201	PADANG	900
A-218	PADANG	700



DENSE INDEKS DAN SPARSE INDEKS

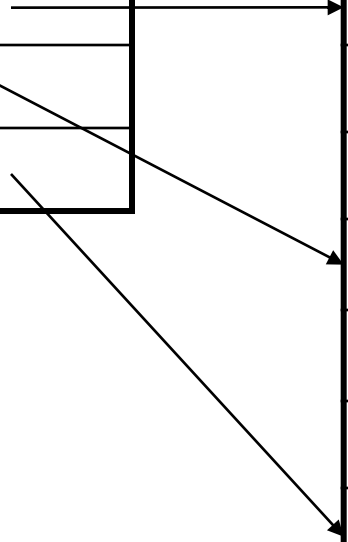
SPARSE INDEX

FILE INDEX

CABANG	POINTER
BOGOR	
MALANG	
PADANG	

FILE MASTER

NO. REK	CABANG	JUMLAH
A-217	BOGOR	750
A-099	DAGO	450
A-101	DAGO	500
A-065	MALANG	300
A-135	MALANG	300
A-215	MALANG	700
A-201	PADANG	900
A-218	PADANG	700



DENSE INDEKS DAN SPARSE INDEKS

- KELEBIHAN DENSE INDEX
  - Mencari lokasi record lebih cepat dibanding sparse index
- KEKURANGAN DENSE INDEX
  - Membutuhkan tempat indeks lebih besar dibanding sparse index
  - Jika file master berubah, maka file index juga harus dirubah (maintenace lebih sulit dibanding dengan sparse index))
- KELEBIHAN SPARSE INDEX
  - Membutuhkan tempat indeks lebih kecil dibanding dense index
  - Maintenace lebih mudah dibanding dengan dense index
- KEKURANGAN SPARSE INDEX
  - Mencari lokasi record lebih lambat dibanding dense index

MULTILEVEL INDEKS

- Meskipun menggunakan sparse index, file index dapat menjadi besar sehingga proses pencarian tidak efisien.

- Misal, jika file master mempunyai record 100.000, dengan tiap blok menyimpan 10 record. Jika 1 record pada file index menyimpan 1 blok. Maka file index mempunyai 10.000 record. File index yang terbentuk masih sangat besar untuk disimpan dalam sebuah disk.

- Jika file index tersebut tidak cukup dimuat di dalam main memory, maka pencarian data akan lambat.

- Untuk mengatasi masalah ini, maka dibuatlah sparse index pada primary index (multilevel index)

MULTILEVEL INDEKS

FILE INDEX LEVEL 1

CABANG	POINTER
ACEH	
JAKARTA	
PADANG	

FILE INDEX LEVEL 2

CABANG	CABANG	POINTER
ACEH	BOGOR	
	BALIKPAPAN	
	CIAMIS	
	DAGO	
JAKARTA	JAKARTA	
	MALANG	
	MEDAN	
PADANG	PADANG	

FILE MASTER

NO. REK	CABANG	JUMLAH
000001	BOGOR	750
:	:	:
001000	DAGO	500
:	:	:
002500	JAKARTA	300
:	:	:
005000	MALANG	700
:	:	:
075000	PADANG	900
:	:	:
100000	PADANG	700

INDEKS UPDATE

- File Indeks Harus Diupdate Jika Proses Insert Atau Delete Record Terjadi

- Insert Record
  - Pada dense indeks

    Jika nilai yang diinsert belum ada pada file indeks , maka nilai dari search key diinsert pada file indeks

  - Pada sparse indeks

    Jika pada file index, nilai yang yang diinsert sudah ada, maka file index tidak usah dirubah, sebaliknya jika pada file index nilai yang diinsert tidak ada, maka file index harus dirubah

INDEKS UPDATE

- Delete record

  Untuk menghapus record, record tersebut harus dicari dulu.

  – Pada dense indeks

    Jika nilai yang didelete hanya satu pada file master, maka pada file indeks nilai yang didelete harus dihapus

    Jika nilai yang didelete lebih dari satu pada file master, maka pada file indeks, nilai yang didelete tidak perlu dihapus

  – Pada sparse indeks

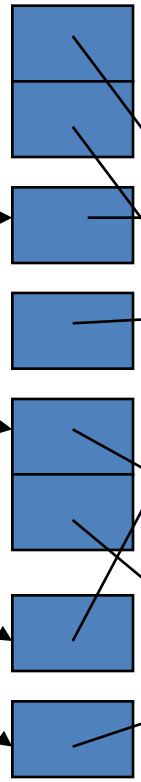
    Jika pada file index, nilai yang yang didelete ada, maka nilai tersebut pada file index harus dihapus, sebaliknya jika pada file index nilai yang didelete tidak ada, maka file index tidak dirubah

SECONDARY INDEKS

SECONDARY INDEKS HARUS BERUPA DENSE INDEKS

FILE INDEX

JUMLAH	POINTER
300	
450	
500	
700	
750	
900	



FILE MASTER

NO. REK	CABANG	JUMLAH
A-217	BOGOR	750
A-099	DAGO	450
A-101	DAGO	500
A-065	MALANG	300
A-135	MALANG	300
A-215	MALANG	700
A-201	PADANG	900
A-218	PADANG	700

FILE INDEKS B^+ -TREE

- Delete record

  Untuk menghapus record, record tersebut harus dicari dulu.

  – Pada dense indeks

    Jika nilai yang didelete hanya satu pada file master, maka pada file indeks nilai yang didelete harus dihapus

    Jika nilai yang didelete lebih dari satu pada file master, maka pada file indeks, nilai yang didelete tidak perlu dihapus

  – Pada sparse indeks

    Jika pada file index, nilai yang yang didelete ada, maka nilai tersebut pada file index harus dihapus, sebaliknya jika pada file index nilai yang didelete tidak ada, maka file index tidak dirubah

ORGANISASI FILE HASHING

- Keuntungan dari organisasi file index sekuential adalah untuk mencari lokasi data, harus mengakses struktur index nya.

- Pada Organisasi File Hash, Untuk Mencari Alamat Dari Record Secara Langsung Dengan Menghitung Fungsi Dari Nilai Search Key Dari Record (Memakai Perhitungan Matematis Untuk Menemukan Alamat Dari Sebuah Record)

- Agar Dapat Dilakukan Direct Access, Key Dari Record Dipakai Sebagai Alamat Di Dalam File

ORGANISASI FILE HASHING

Komponen Hashed File:

- File Space
  - Terbagi dalam slot-slot
  - Tiap slot menyimpan sebuah record

- Rumus

  Menghasilkan slot address, dihitung berdasarkan key dari sebuah record

ORGANISASI FILE HASHING

OVERVIEW HASHED FILE

- Berbasis kemampuan direct access ke dalam file dengan memanfaatkan relatif address

- RELATIF ADDRESS ADALAH:

  Sebuah Record Dapat Ditemukan Hanya Dengan Memanggilnya Lewat Nomor Urut Record Di Dalam File

- MASALAHNYA ADALAH:

  Membuat rumus untuk mengubah key dari sebuah record menjadi nomor urut (kat -> key to address transformation)

ORGANISASI FILE HASHING

KAT (KEY TO ADDRESS TRANSFORMATION) :

- Tujuannya untuk menghasilkan slot number yang berbeda bagi tiap record
- Dengan cara mengubah key menjadi relative address

Hambatan kat:

- Key umumnya sesuatu yang bersifat natural (nim / no_ktp / no_pegawai / dll)
- Natural key biasanya panjang (nim = 10 digit)

ORGANISASI FILE HASHING

PERSYARATAN KAT:

- Ukuran key harus diperpendek agar sesuai dengan slot address (relative address)
- Slot address yang dihasilkan harus unix
- Algoritma untuk membuat kat sangat banyak

ORGANISASI FILE HASHING

NIM	NAMA	SLOT ADDRESS
0011500001	BUDIMAN	1
0011500002	HERMAN	2
-	-	-
-	-	-
0011500105	ACHMAD	105
0011500106	ENDANG	106
0011500107	SEPHIA	107

KAT : 3 DIGIT TERAKHIR DARI NIM