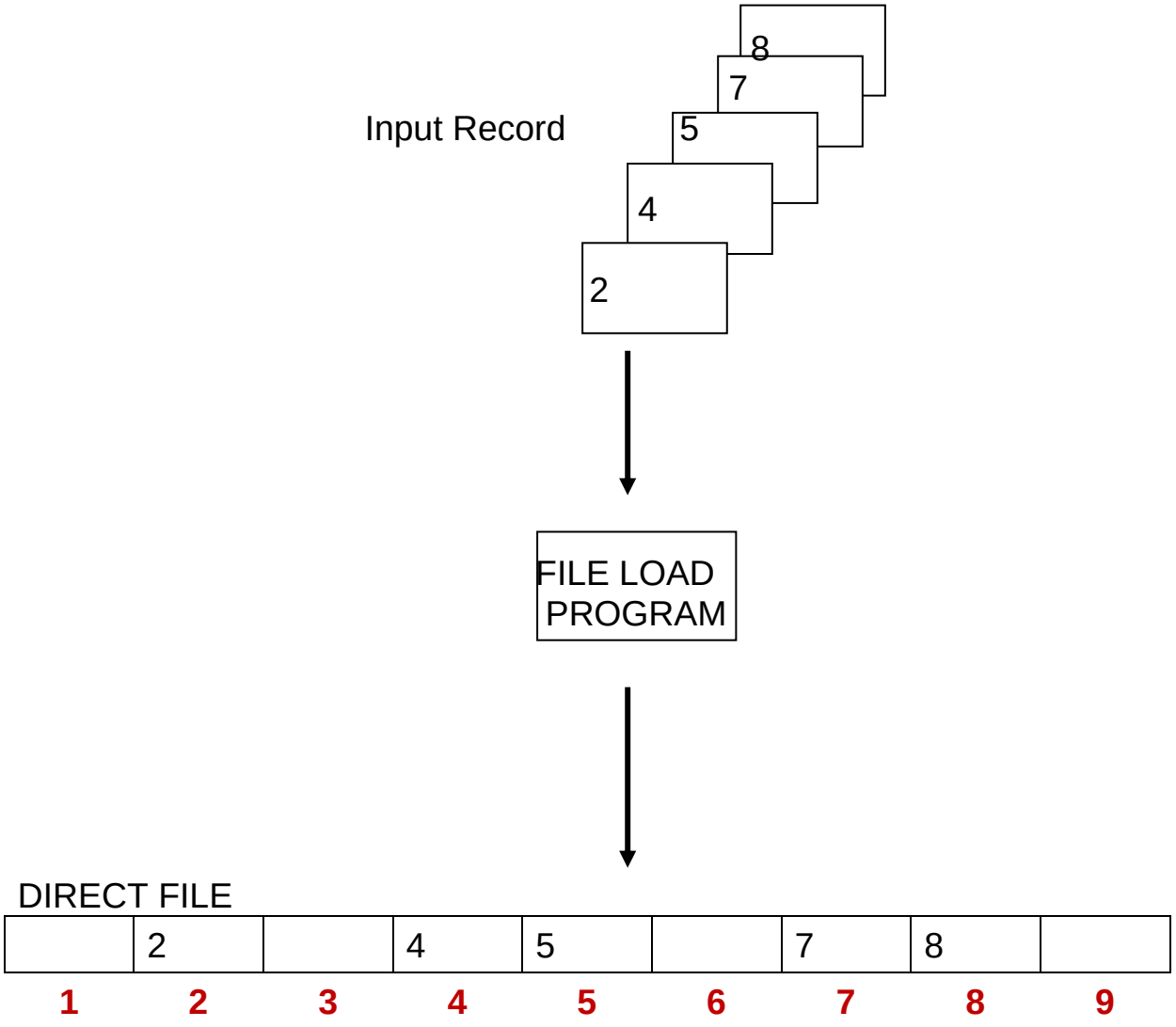
ORGANISASI BERKAS RELATIF

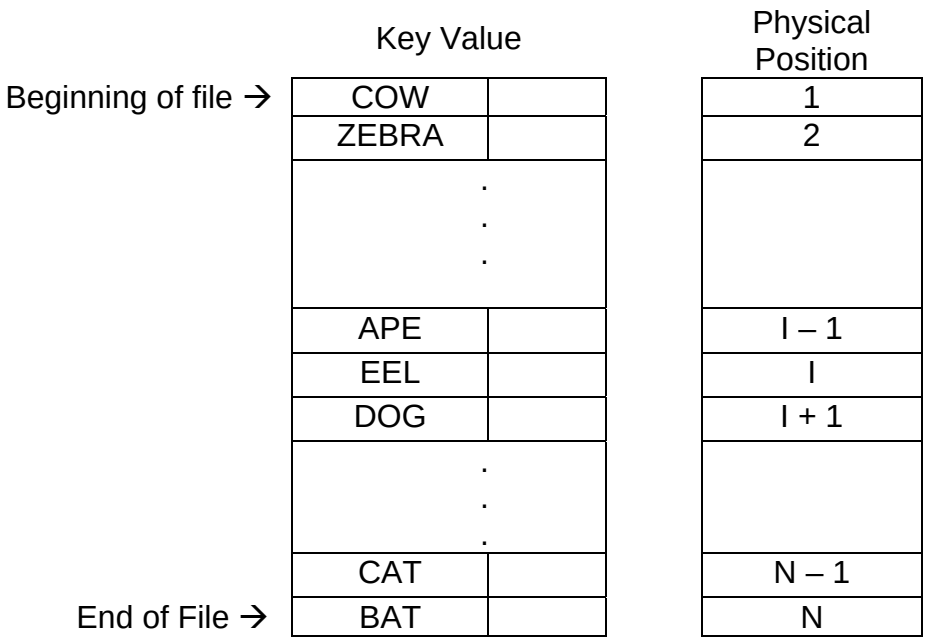
PENGERTIAN BERKAS RELATIF

Suatu cara yang efektif dalam mengorganisasi sekumpulan record yang membutuhkan akses sebuah record dengan cepat adalah Organisasi Berkas Relatif. Dalam berkas relatif ada hubungan antara key yang dipakai untuk mengidentifikasi record dengan lokasi record dalam penyimpanan sekunder.

Urutan record secara logik tidak ada hubungannya dengan urutan secara fisik. Record tidak perlu tersortir secara fisik menurut nilai key.



Gambar 1. Organisasi Direct/Langsung



Gambar 2. Dasar Organisasi Berkas Relatif

Bagaimana record yang ke-N dapat ditemukan?. Dalam hal ini, perlu kita buat hubungan yang akan menterjemahkan antara NILAI KEY dan ADDRESS.

Hubungan ini dinyatakan sebagai R, yang merupakan fungsi pemetaan:

$$R(\text{NILAI KEY}) \longrightarrow \text{ADDRESS}$$

dari nilai key ke address dalam penyimpanan sekunder.

PROSES

Pada waktu sebuah record ditulis kedalam berkas relatif, fungsi pemetaan R digunakan untuk menterjemahkan NILAI KEY dari record menjadi ADDRESS, dimana record tersebut disimpan.

Begitu pula pada waktu akan me-*retrieve* record dengan nilai key tertentu, fungsi pemetaan R digunakan terhadap nilai key tersebut, untuk menterjemahkan nilai key itu menjadi sebuah address dalam penyimpanan sekunder, dimana record tersebut ditemukan.

Organisasi berkas relatif ini tidak menguntungkan bila penyimpanan sekundernya berupa media SASD seperti magnetic tape. Berkas relatif harus disimpan dalam media DASD, seperti magnetic disk atau drum. Juga dimungkinkan untuk mengakses record-record dalam berkas relatif

secara berurutan, tetapi perlu diketahui bahwa nilai key tidak terurut secara logik.

Contoh:

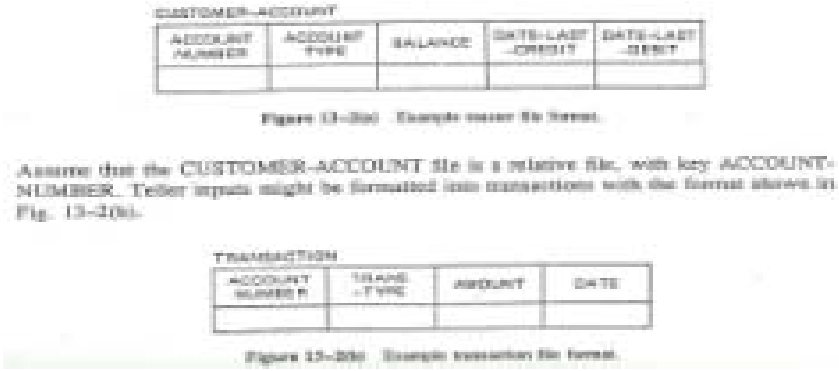
Record pada gambar 2, di-*retrieve* secara berurutan;

COW, ZEBRA, … , APE, EEL, DOG, … , CAT, BAT

Karena kemampuan mengakses record tertentu secara cepat, maka organisasi berkas relatif paling sering digunakan dalam proses interactive.

Contoh:

Sebuah on-line sistem perbankan yang mempunyai sebuah master file dan sebuah transaksi file. Field ACCOUNT NUMBER dipakai sebagai nilai key untuk kedua berkas tersebut. Pada saat nilai key ACCOUNT NUMBER dimasukan kedalam transaksi, nilai key tersebut akan me-*retrieve* secara langsung record yang ada pada master file.



Gambar 3. Contoh Format Berkas Transaksi

Jika Trans-Type = ‘I’, maka BALANCE ACCOUNT akan ditampilkan di layar.

Jika Trans-Type = ‘C’ atau ‘D’, maka record-record dari master file Customer Account akan dimodifikasi dengan AMOUNT dan DATE yang ada di transaction file, dimana ACCOUNT NUMBER yang menentukan lokasi record dalam berkas tersebut.

Catatan:

- Kita tidak perlu mengakses semua record master file, cukup mengakses langsung record yang dikehendaki.
- Record dari berkas relatif dapat di-*update* langsung tanpa perlu merekam kembali semua record.
- Keuntungan dari berkas relatif ini adalah kemampuan mengakses record secara langsung. Sebuah record dapat di *retrieve*, insert, modifikasi atau di *delete*, tanpa mempengaruhi record lain dalam berkas yang sama.

Ada 3 teknik dasar yang digunakan untuk menyatakan fungsi pemetaan R, dimana R(NILAI KEY) → ADDRESS

1. Direct Mapping (Pemetaan Langsung)
2. Directory Look Up (Pencarian Tabel)
3. Calculation (Kalkulasi)

TEKNIK PEMETAAN LANGSUNG

Teknik ini merupakan teknik yang sederhana untuk menterjemahkan nilai record key menjadi address. Ada 2 cara dalam pemetaan langsung, yaitu:

1. Absolute Addressing (Pengalamatan Mutlak)
2. Relative Addressing (Pengalamatan Relatif)

Pengalamatan Mutlak

R(NILAI KEY) ⟶ ADDRESS
NILAI KEY = ALAMAT MUTLAK

Jika nilai key yang diberikan oleh pemakai program sama dengan ADDRESS sebenarnya dari record tersebut pada penyimpanan sekunder. Pada waktu record tersebut disimpan, lokasi penyimpanan record (nomor silinder, nomor permukaan, nomor record) bila dipakai Cylinder Addressing atau (nomor sektor, nomor record) bila dipakai Sector Addressing harus ditentukan oleh pamakai.

Keuntungan dari Pengalamatan Mutlak

- Fungsi pemetaan R sangat sederhana.
- Tidak membutuhkan waktu lama dalam menentukan lokasi record pada penyimpanan sekunder.

Kelemahannya

- Pemakai harus mengetahui dengan pasti record-record yang disimpan secara fisik
- Alamat mutlak adalah device dependent. Perbaikan atau pengubahan device, dimana berkas berada akan mengubah nilai key.
- Alamat mutlak adalah address space dependent. Reorganisasi berkas relatif akan menyebabkan nilai key berubah.

Pengalamatan Relatif

R(NILAI KEY) ⟶ ADDRESS
NILAI KEY = ALAMAT RELATIF

Alamat relatif dari sebuah record dalam sebuah berkas adalah urutan record tersebut dalam berkas. Sebuah berkas dengan N record mempunyai record dengan alamat relatif dari himpunan (1,2,3, …, N -2, N -1). Record yang ke I mempunyai alamat relatif I atau I – 1 (bila mulai dihitung dari 0).

Keuntungan dari Pengalamatan Relatif

- Fungsi pemetaan R sangat sederhana.
- Nilai key dari sebuah record dapat ditentukan lokasi recordnya dalam sebuah penyimpanan sekunder tanpa memerlukan waktu proses yang berarti.

Kelemahannya

- Alamat Relatif adalah bukan device dependent.
- Alamat Relatif adalah address space dependent.
- Terjadinya pemborosan ruangan.

TEKNIK PENCARIAN TABEL

Dasar pemikiran pendekatan pencarian tabel adalah sebuah tabel atau direktori dari nilai key dan address. Untuk menemukan sebuah record dalam berkas relatif, pertama dicari dalam direktori nilai key dari record tersebut, yang akan menunjukan alamat dimana record tersebut berada dalam penyimpanan.

Data dalam direktori tersebut disusun secara urut menurut nilai key, sehingga pencarian nilai key dalam direktori lebih cepat dengan binary search dibanding sequential search. Alternatif lain, direktori dapat disusun dalam Binary Search Tree, M-way Search Tree atau B-Tree.

Directory

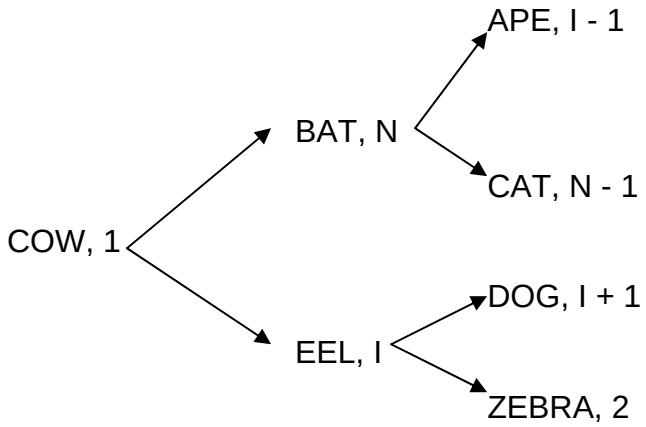
Key	Address
APE	I – 1
BAT	N
CAT	N – 1
.	
COW	1
DOG	I + 1
EEL	I
.	
ZEBRA	2

File Relatif
COW
ZEBRA
.
APE
EEL
DOG
.
CAT
BAT

Alamat Relatif
1
2
I – 1
I
I + 1
N – 1
N

Gambar 4. Berkas Relatif dengan Struktur Tabel

DIRECTORY



Gambar 5. Berkas Relatif dengan Struktur Pohon

Keuntungan dari Pencarian Tabel

- Sebuah record dapat diakses dengan cepat, setelah nilai key dalam direktori ditentukan.
- Nilai key dapat berupa field yang mudah dimengerti, seperti PART NUMBER, NPM, karena nilai key tersebut akan diterjemahkan menjadi alamat.
- Nilai key adalah address space independent, dimana reorganisasi berkas tak akan memepengaruhi nilai key, yang berubah adalah alamat dalam direktori.

TEKNIK KALKULASI ALAMAT

R (NILAI KEY) $\longrightarrow$ ADDRESS

Adalah dengan melakukan kalkulasi terhadap nilai key, hasilnya adalah alamat relatif. Ide dasar dari kalkulasi alamat adalah mengubah jangkauan nilai key yang mungkin, menjadi sejumlah kecil alamat relatif.

Salah satu kelemahan dari teknik pengalamatan relatif adalah ruang harus disediakan sebanyak jangkauan nilai key, terlepas dari berapa banyak nilai key.

Salah satu masalah dari teknik ini adalah ditemukannya alamat relatif yang sama untuk nilai key yang berbeda.
Keadaan dimana:

$$\left.\begin{array}{l} R(K1) = R(K2) \\ K1 \neq K2 \end{array}\right\} \text{disebut benturan atau collision}$$

Sedangkan nilai key K1 dan K2 disebut synomin.
Synonim adalah dua atau lebih nilai key yang berbeda pada hash ke home address yang sama.

Teknik-teknik yang terdapat pada Kalkulasi Alamat

- Scatter storage techniques
- Randomizing techniques
- Key-to-address transformation methods
- Direct addressing techniques
- Hash table methods
- Hashing

Disini yang akan kita bahas mengenai teknik hashing. Kalkulasi terhadap nilai key untuk mendapatkan sebuah alamat disebut fungsi hash.

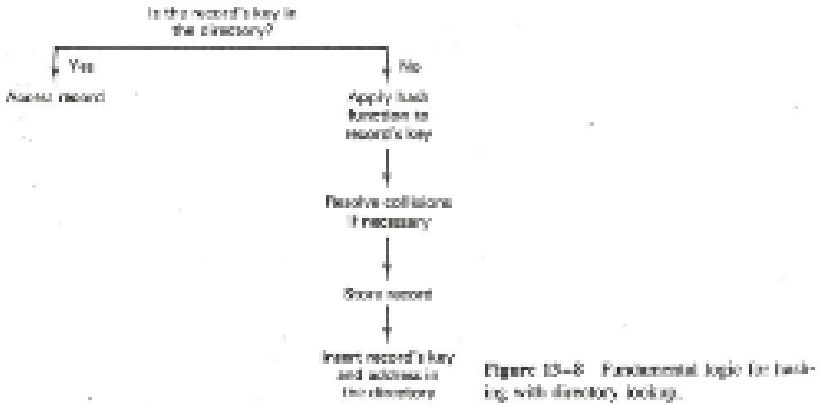
Keuntungan pemakaian Hashing

- Nilai key yang sebenarnya dapat dipakai karena diterjemahkan kedalam sebuah alamat.
- Nilai key adalah address space independent bila berkas direorganisasi, fungsi hash berubah tetapi nilai key tetap.

Kelemahannya

- Membutuhkan waktu proses dalam mengimplementasikan fungsi hash.
- Membutuhkan waktu proses dan akses I/O dalam mengatasi benturan.

Hashing dapat digunakan bersama-sama dengan pencarian tabel.



Gambar 6. Hashing dengan Directory Lookup

Penampilan fungsi Hash bergantung pada:

- Distribusi nilai key yang dipakai.
- Banyaknya nilai key yang dipakai relatif terhadap ukuran dari ruang alamat.
- Banyaknya record yang dapat disimpan pada alamat tertentu tanpa menyebabkan benturan.
- Teknik yang dipakai untuk mengatasi benturan

Beberapa fungsi Hash yang umum digunakan:

- Division Remainder
- Mid Square
- Folding

DIVISION REMAINDER

Pada division remainder, alamat relatif dari suatu nilai key merupakan sisa dari hasil pembagian nilai key tersebut dengan suatu bilangan yang disebut sebagai *bilangan pembagi.*

Contoh:

Bila DIV adalah pembagi, KEY adalah nilai key dan ADDR adalah alamat relatif, maka dalam bahasa Pascal, fungsi R(NILAI KEY) → ADDRESS dapat di implementasikan:

```
ADDR := KEY MOD DIV
```

Dalam bahasa COBOL:

```
DIVIDE KEY BY DIV GIVING TEMP REMAINDER ADDR
```

Sisa pembagian (sebagai hasil dari fungsi MOD pada Pascal), dapat dijabarkan sebagai berikut:

```
ADDR := KEY - DIV * TEMP
```

ADDR Harus merupakan bilangan integer.

Banyak faktor yang harus dipertimbangkan dalam pemilihan pembagi:

- Jangkauan dari nilai key yang dihasilkan dari opersi KEY MOD DIV adalah 0 sampai DIV-1. Nilai dari DIV menentukan ukuran "relatif address space". Jika diketahui berkas relatif terdiri dari N record dan dianggap hanya satu record dapat disimpan dalam sebuah alamat relatif, maka akan didapat DIV > N.

- Pembagi harus diseleksi untuk mengurangi benturan. Riset menunjukkan bahwa pembagi yang berupa bilangan genap akan

cenderung jelek, terutama dengan nilai key-nya yang dominan ganjil.

- Menurut riset dari W.Buchholz, sebaiknya pembagi itu merupakan bilangan prima. Tetapi riset lain dari V.Y.Lum, menyatakan pembagi yang bukan bilangan prima akan memberikan hasil yang sama baik, seperti bilangan prima.

- Menurut pendapatnya, bukan bilangan prima yang mempunyai faktor prima kurang dari 20 akan dapat memberikan jaminan hasil yang lebih baik.

- Walaupun kita telah menentukan pembagi dengan baik untuk mengatasi benturan, bila ruang alamat dari berkas relatif mendekati penuh, maka peluang terjadinya benturan akan meningkat.

Untuk mengukur kepenuhan berkas relatif digunakan Load Factor (Faktor Muat).

$$\text{Load Factor} = \frac{\text{banyak record dalam berkas}}{\text{max. banyak record dalam berkas}}$$

Biasanya load factor yang sering digunakan adalah 0.7 atau 0.8.
Jika load factor lebih besar dari 0.7 atau 0.8 maka berkas tersebut harus diperbesar dan di reorganisasi.

Jadi jika kita ingin menyimpan sebanyak n record pada suatu berkas dan load factor adalah 0.8, maka max. banyak record pada berkas adalah 1.25 n.

$$0.8 = \frac{n}{max}$$
$$max = 1.25\ n$$

Contoh:

Kita ingin membuat berkas yang terdiri dari 4000 record.
Load Factor (Faktor muat) = 0.8

maka max. banyak record pada berkas:

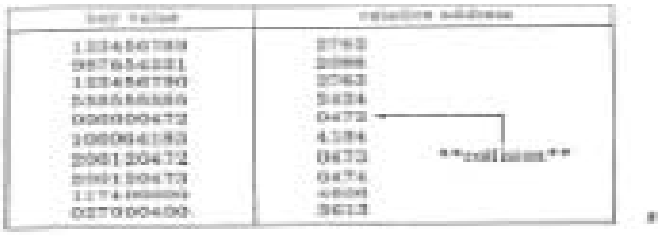
$$(1.25)\ n = (1.25) \times 4000$$
$$= 5000$$

Bilangan pembagi: 5003

$$\frac{123456789}{5003} = 24676 \text{ sisa } 2761 + 1 \quad \text{(alamat relatif)}$$

$$\frac{987654321}{5003} = 197412 \text{ sisa } 2085 + 1 \quad \text{(alamat relatif)}$$

Jadi alamat relatif didapat dari sisa pembagian + 1



Gambar 7. Contoh penggunaan pembagi 5003

MID SQUARE HASHING

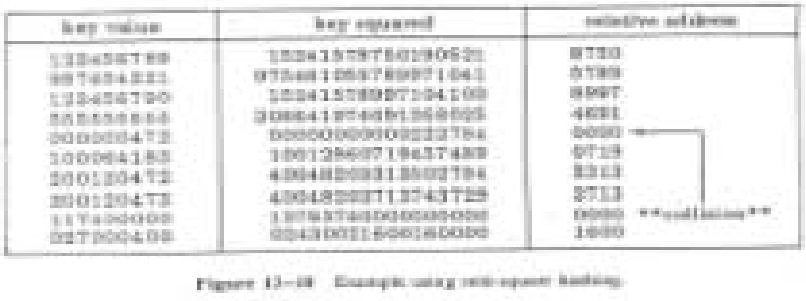
Untuk mendapatkan alamat relatif, nilai key dikuadratkan, kemudian beberapa digit diambil dari tengah.

Dari nilai key yang dikuadratkan kita cari tengah-tengahnya.
Jumlah nilai key yang dikuadratkan:

nilai key 123456789 = 17 digit.

Untuk alamat relatif = $\frac{17}{2} = 8.5$

Kita mulai dari digit ke 8 dihitung dari kiri, maka alamat relatif = 8750 (karena ditentukan 4 digit sebagai alamat relatif).



Gambar 8. Contoh Mid Square Hashing

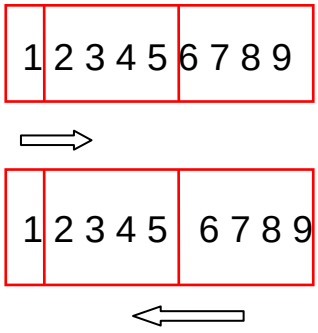
HASHING BY FOLDING

Untuk mendapatkan alamat relatif, nilai key dibagi menjadi beberapa bagian, setiap bagian (kecuali bagian terakhir) mempunyai jumlah digit yang sama dengan alamat relatif.

Bagian-bagian ini kemudian dilipat (seperti kertas) dan dijumlah.
Hasilnya, digit yang tertinggi dibuang (bila diperlukan).

Contoh:

Nilai key 123456789 dan alamat relatif sebanyak 4 digit. Nilai key dibagi menjadi bagian-bagian yang terdiri dari 4 digit, mulai dari sebelah kanan.



Menghasilkan:

```
  1
  2 3 4 5
  9 8 7 6   +
-----------
1 3 2 2 1
  (alamat relatif)
```

Gambar 9. Contoh Hashing by Folding

Perbandingan Fungsi Hash

- Teknik Division Remainder memberikan penampilan yang terbaik secara keseluruhan.
- Teknik Mid Square dapat dipakai untuk file dengan load factor cukup rendah akan memberikan penampilan baik, tetapi kadang-kadang dapat menghasilkan penampilan yang buruk dengan beberapa collision.
- Teknik Folding adalah teknik yang paling mudah dalam perhitungan, tetapi dapat memberikan hasil yang salah, kecuali panjang nilai key = panjang address.

Pendekatan terhadap masalah Collision

Ada 2 pendekatan dasar untuk menetapkan dimana K2 harus disimpan, yaitu:

- Open Addressing
- Separate Overflow

Open Addressing

Menemukan address yang bukan home address untuk K2 dalam berkas relatif.

Contoh:

K1 = 1 K2 = 1

R1 R2

K1	K2

Separate Overflow

Menemukan address untuk K2 diluar dari primary area dalam berkas relatif, yaitu di overflow area yang dipakai hanya untuk menyimpan record-record yang tak dapat disimpan di home address nya.

Contoh:

K1 = 1 K2 = 1

R1

K1		

Overflow area

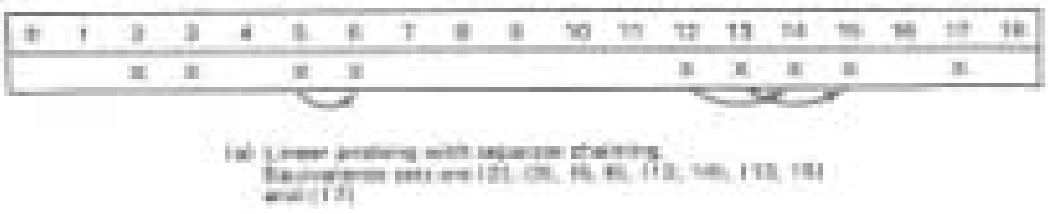
K2		

Ada 2 teknik untuk mengatasi Collision:

- Linier Probing, yang merupakan teknik open addresing.
- Double Hashing, yang dapat dipakai selain open addressing atau separate overflow.

Linear Probing

Salah satu cara menemukan lokasi record yang tak dapat disimpan di home address nya adalah dengan menggunakan Linear Probing, yang merupakan sebuah proses pencarian secara sequential/linear dari home address sampai lokasi yang kosong.



Gambar 10. Linear Probing

Double Hashing

Pendekatan lain dalam menemukan lokasi sebuah record pada waktu record tersebut tidak dapat disimpan dalam home address nya adalah dengan menggunakan Double Hashing, yang akan memakai fungsi hash kedua terhadap hasil dari fungsi hash pertama. Address dari record yang di hash kembali dapat terletak pada primary area atau di separate overflow area.

Keuntungan dari metode separate overflow adalah menghindari keadaan dimana dapat terjadi metode open addressing untuk sebuah record yang tak disimpan dalam home address nya menggantikan record lain yang terakhir di hash ke home address nya.

Masalah ini dapat dihindari dengan open addressing sederhana dengan memindahkan record yang sebelumnya ke lokasi lain (dengan probing atau hashing kembali) dan menyimpan record yang baru ketempat yang kosong.

Metode ini membutuhkan pengeluaran tambahan untuk pemeliharaan berkas. Berkas relatif dibagi menjadi 2 berkas , yaitu: *Primary Area dan Overflow Area.*

Gambar 11. Double Hashing

Perbandingan Linear Probing dan Double Hashing

Berkas dengan load factor kurang dari 0.5 pada linear probing akan menghasilkan synonim yang mengelompok, sedangkan double hashing synonimnya berpencar.

Load Factor < 0.5 : Double Hashing = Linear Probing.
Load Factor > 0.8 : Double Hashing > Linear Probing.

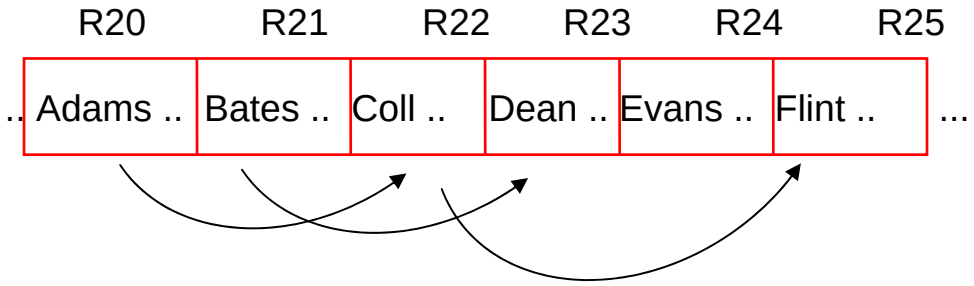
Synonim Chaining

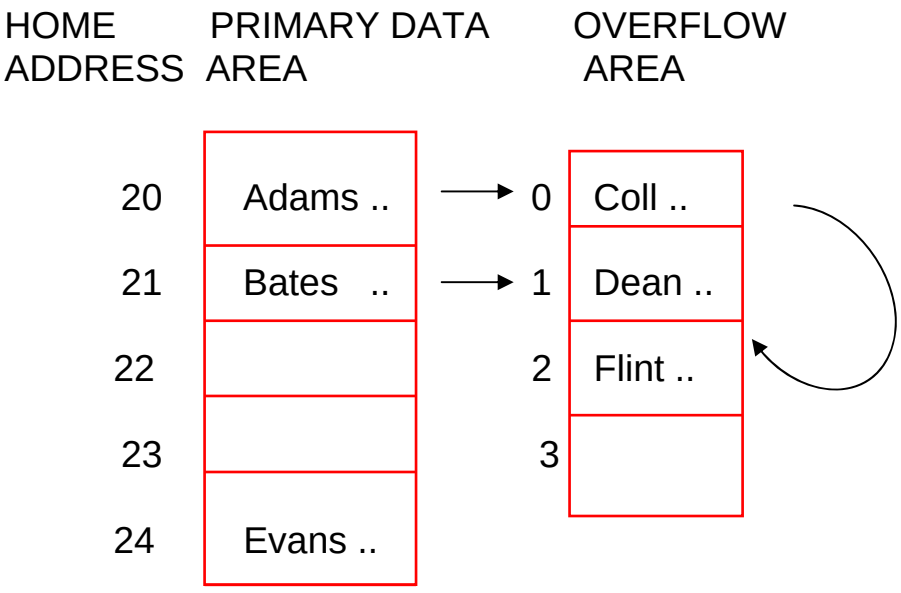
Pendekatan pemecahan collision yang mengakses synonim dengan fasilitas link-list untuk record-recordnya dalam kelas ekivalen. Adapun link list record-record dengan home address yang sama tak akan

mengurangi jumlah collision, tetapi akan mengurangi waktu akses untuk me-retrieve record-record yang tak ada di home address nya.

Contoh:

KEY	HOME ADDRESS	ACTUAL ADDRESS
Adams	20	20
Bates	21	21
Coll	20	22
Dean	21	23
Evans	24	24
Flint	20	25





Gambar 12. Hashing dengan Synonim Chaining

Bucket Addressing

Pendekatan lain dalam mengatasi collision adalah hash ke dalam block atau bucket yang dapat memberikan tempat sejumlah record.

Contoh:

Sebuah berkas relatif mempunyai relatif address space dari 0 sampai M dan sebuah bucket berukuran B record, address space akan terdiri dari B(M+1) record. Jika file terdiri dari N record, maka:

$$\text{Factor Muat} = \frac{N}{B(M+1)}$$

B record dapat semuanya di hash kedalam relatif address yang sama tanpa menyebabkan collision.

Pada saat sebuah bucket penuh, beberapa tempat baru harus ditemukan untuk record tersebut. Pendekatan dari masalah bucket penuh pada dasarnya sama dengan pendekatan untuk mengatasi collision dengan record addressing.

Jika open addressing dipakai, space dicari untuk bucket berikutnya (misal dengan linear probing) atau dalam bucket lainnya (misal dengan double hashing).

Jika teknik separate overflow yang dipakai, record baru ditempatkan dalam suatu himpunan bucket yang dirancang khusus untuk tempat record yang tak dapat ditampung pada bucket primer.
Bucket ini disebut bucket overflow.

Record-record yang disimpan dalam sebuah bucket dapat dikelola dalam:

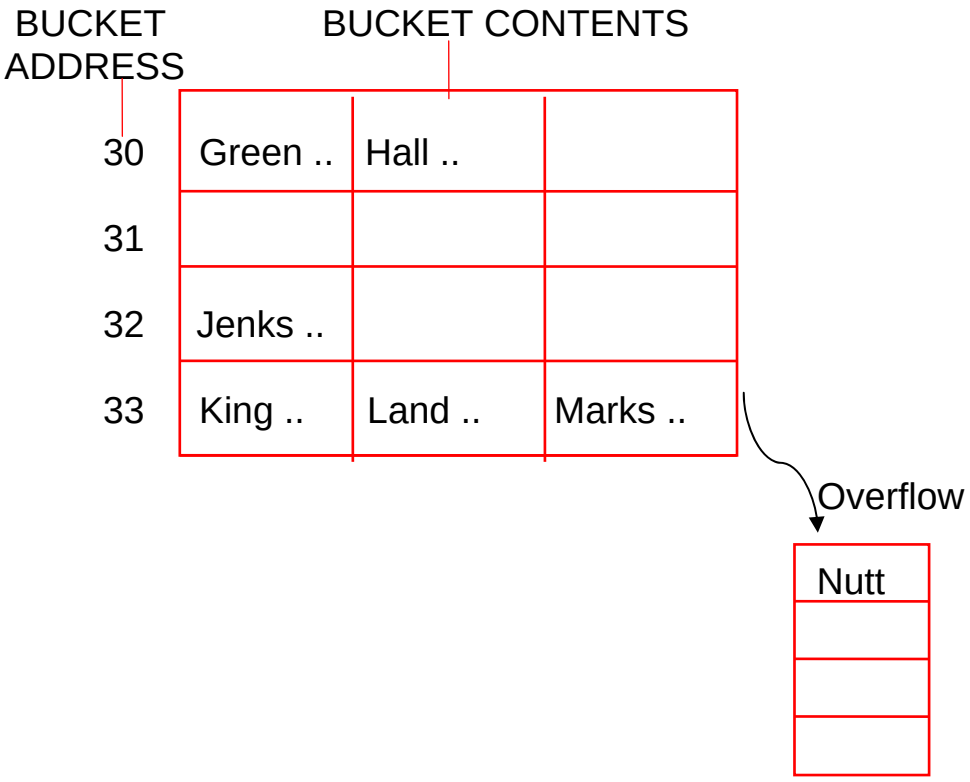
- Dapat disisipkan dalam urutan berdasarkan penempatannya di bucket.
- Dapat dipertahankan urutan nilai key-nya.

Bucket addressing ini umum dipakai.
Ukuran dari sebuah bucket dapat ditentukan oleh ukuran track atau sektor dalam DASD. Ukuran bucket umumnya sama dengan ukuran block untuk file.

Satu keuntungan penting dari penggunaan bucket yang dapat menampung banyak record ini adalah record dengan panjang yang berbeda dapat dipakai.

Contoh:

KEY	HOME ADDRESS
Green	30
Hall	30
Jenk	32
King	33
Land	33
Mark	33
Nutt	33



Gambar 13. Bucket Adrressing