ARCHIVI HASH

L'organizzazione hash si utilizza quando occorre velocizzare la ricerca delle informazioni negli archivi di grandi dimensioni.

L'archivio è composto da M record, inizialmente vuoti, a cui si può accedere in modo diretto tramite l'indirizzo relativo di ciascun record.

L'indirizzo relativo del record contenente l'informazione da ricercare viene calcolato da una funzione matematica della chiave primaria (**funzione di trasformazione** o funzione hash):

$$P=h(K)$$

con P = indirizzo relativo, K = chiave, h() funzione hash.

La trasformazione è **perfetta** se ad ogni chiave corrisponde un indirizzo distinto (corrispondenza biunivoca tra l'insieme di tutte le possibili chiavi e l'insieme degli indirizzi 1..M).

Gli archivi informatici contengono un numero ridotto di chiavi rispetto a tutte le possibili chiavi teoriche: con una chiave alfanumerica di 5 caratteri si hanno 26^5=11.881.376 configurazioni differenti.

L'inaccettabile occupazione di memoria di massa richiesta nel caso di trasformazione perfetta, porta nella pratica a limitare il numero massimo dei record dell'archivio (**trasformazione non perfetta**).

Riducendo lo spazio di memoria e gli indirizzi è inevitabile il verificarsi di "**collisioni**" (sinonimi) : due o più chiavi producono il medesimo indirizzo.

Si definisce **rapporto di occupazione** o **fattore di carico** ==> n° chiavi presenti/n° record totali.

Per non occupare troppo spazio nelle memorie di massa il fattore di carico deve essere maggiore di 0.5.

Requisiti della funzione hash:

- **distribuzione uniforme** delle chiavi nello spazio degli indirizzi (uniforme possibilità di collisione);

- **distribuzione casuale** delle chiavi: chiavi simili non devono produrre indirizzi identici;

- **velocità di calcolo** della funzione per non ritardare le operazione.

Esempi di funzioni di trasformazione (con chiavi numeriche) :

Es.1 - K=76497632987 M=10.000 si preleva una parte della chiave P=2987, parte terminale, o P=7649, parte iniziale, o P=9763, parte centrale.

Es.2 - K=6345635 M=10.000 si calcola il quadrato di K e si preleva una parte del quadrato, si ha una maggiore casualità negli indirizzi.

Es.3 - P=K mod M con M=n.di record (M dovrebbe essere un numero primo per garantire una migliore distribuzione).

Es.4 - Si divide la chiave in due o più parti e si sommano: K=76497632987 M=5.000 K'=764+9763+2987=13514 P=13514 mod 5000=3514 , si ha una maggiore casualità poiché nel calcolo vengono usate tutte le cifre.

Con chiavi alfanumeriche occorre prima trasformare l'informazione alfanumerica in numerica, ad es. utilizzando il codice Ascii di ciascun carattere, e quindi procedere al calcolo della funzione hash.

RISOLUZIONE DELLE COLLISIONI

a) Metodi ad **indirizzamento aperto.**

Si genera una sequenza di indirizzi hash (sequenza di scansione).

La legge di scansione dovrebbe generare tutti gli indirizzi hash prima di riproporre l'indirizzo iniziale h(k).

- "**lineare con passo unitario**" : in caso di collisione gli indirizzi successivi al primo sono quelle:

$$P[i+1]=(P[i]+1) \bmod M \quad \text{con} \quad P[1]=h(K)$$

oppure $P[i]=(h(K)+i) \bmod M$

Tecnica dell'overflow progressivo.

Produce il fenomeno dell'agglomerazione (**clustering**).

- "**doppia trasformazione o rehashing**" la scansione è ancora lineare ma con passo dipendente dalla chiave:

$$P[i+1]=(P[i]+h'(k)) \bmod M \quad \text{con} \quad P[1]=h(K)$$

oppure $P[i]=(h(K)+i*h'(K)) \bmod M$

La funzione h'(K) determina il passo di scansione, dipendente dalla chiave.

Ad esempio : h'(K)=h(K) mod (M-n).

In entrambi i casi non si possono cancellare i record fisicamente poiché si interromperebbe la scansione nella ricerca delle chiavi successive a quella cancellata.

Occorre quindi cancellare i record solo logicamente utilizzando dei flag (campi booleani) appositamente predisposti nella struttura del record.

b) Metodi di **concatenazione**.

"**liste confluenti**": i record dei sinonimi vengono collegati in una catena attraverso un campo di link, mentre gli indirizzi hash sono calcolati come in precedenza con la funzione di scansione.

- Si risolvono così i problemi della cancellazione e dell'agglomerazione delle chiavi.
- Può però accadere che in una stessa catena confluiscano delle chiavi aventi differenti indirizzi hash iniziali; in questo caso possono esserci problemi nella cancellazione delle chiavi.

"**liste distinte**": l'archivio è diviso in due parti: "address region", intervallo dei possibili indirizzi hash, e "cellar" (cantina), area di heap per le liste di collisione. Si ha in pratica una specie di area di overflow concentrata in cui vengono inserite le liste delle chiavi che hanno prodotto collisioni; le liste sono distinte per ogni indirizzo hash di partenza.

Si ottengono i migliori risultati con il metodo delle liste distinte dimensionando la "cellar" al un valore pari al 14% dell'archivio (86% per l'address region), valori maggiori o minori produrrebbero in un caso liste troppo lunghe e nell'altro il possibile riempimento della "cellar".

Vantaggi :

Il numero medio di accessi all'archivio nella ricerca di una chiave è indipendente dal numero delle chiavi totali e risulta minore di 1.5 con un fattore di carico di 0.5.

Tale organizzazione risulta in assoluto **la più veloce nella ricerca delle chiavi**.

Svantaggi :

- **Impossibilità ad effettuare una scansione sequenziale ordinata** dal momento che le chiavi hanno una distribuzione casuale e quindi non ordinata.

Le liste dei sinonimi possono avere lunghezze notevoli con tempi di accesso per alcune chiavi troppo lunghi in applicazioni particolari (real time).

Occupazione non ottimale della memoria, il fattore di carico non deve essere molto elevato al fine di limitare il numero di collisioni.

E' necessaria una stima preventiva delle dimensioni dell'archivio; in caso di sovrastima si occuperrebbe troppo spazio, mentre in caso di sottostima si rischierebbe il riempimento dell'archivio e la necessità di procedere ad una riorganizzazione ed al ricalcolo degli indirizzi di tutte le chiavi (rehashing).

Per risolvere l'ultimo problema sono state sviluppate delle tecniche di **trasformazione dinamica** che consentono di modificare gradualmente la grandezza dell'archivio; ad esempio le tecniche di hashing virtuale e di hashing lineare consentono di iniziare con un archivio di piccole dimensioni allargandolo successivamente all'aumentare delle chiavi inserite.

Conclusioni :

Le prestazioni di un archivio con organizzazione hash dipendono da:

- qualità della funzione hash - uniformità, casualità e velocità di calcolo;
- metodo di risoluzione delle collisioni;
- rapporto di occupazione o fattore di carico.

L'organizzazione hash viene utilizzata solamente in archivi di grandi dimensioni quando occorre velocizzare al massimo la ricerca e l'accesso all'archivio viene effettuato esclusivamente su una singola chiave alla volta.