LIBRO PIÙ
il libro si estende sul web

LEZIONE 2

L'organizzazione relative e l'organizzazione hash

L'organizzazione relative

Nell'**organizzazione relative** la chiave primaria di ogni record identifica univocamente sia il record stesso ,sia l'indirizzo logico in cui il record è registrato.

In altri termini, questo tipo di organizzazione è possibile solo quando la chiave corrisponde alla posizione del record all'interno dell'archivio.
Ad esempio, un foglio di calendario è strutturato in modo che a ogni giorno *k* corrisponda la *k*-esima riga dello stesso foglio. Quindi il giorno 5 si trova in quinta riga, il giorno 28 in ventottesima riga. Supponiamo di dover registrare un tabulato con le temperature di una località rilevate ogni ora. È chiaro che troveremo la temperatura delle ore 12 nella dodicesima riga, quella delle 20 nella ventesima riga e così via. Con tale organizzazione, l'archivio può essere paragonato a una specie di vettore memorizzato su dispositivi di memoria di massa.

TABULATO TEMPERATURE	
Ore 1	– 2
Ore 2	– 2
Ore 3	– 1
...	
Ore 23	1
Ore 24	0

Nell'organizzazione relative, la gestione degli indirizzi fisici è affidata al sistema operativo. Per determinare l'indirizzo del *k*-esimo record, il sistema operativo moltiplica la dimensione del record, ossia la sua lunghezza, per $(K-1)$ e, successivamente, somma tale valore a quello dell'indirizzo fisico del primo record registrato nell'archivio.

Un altro esempio di archivio a organizzazione relative può essere quello che contiene le informazioni inerenti i posti di un aereo, che sono numerati da 1 a *N*. Ciascun record dell'archivio, pertanto, rappresenta un posto che sarà così individuato tramite il suo numero.
Tra i principali vantaggi di questo tipo di organizzazione, ricordiamo che:

- a ogni record corrisponde una chiave numerica indicante la posizione occupata dal record all'interno dell'archivio (indirizzo logico);
- l'organizzazione relative può essere utilizzata solo su supporti di memorizzazione ausiliari ad accesso diretto;
- in tale tipo di organizzazione è consentito sia l'accesso sequenziale, sia quello diretto, quindi l'organizzazione relative può essere gestita indifferentemente come archivio sequenziale o come archivio non sequenziale;
- l'organizzazione relative risulta molto vantaggiosa per le operazioni interattive;
- utilizzando puntatori in questo tipo di organizzazione è possibile ottenere un ordine logico diverso da quello fisico;
- l'organizzazione relative offre la possibilità di accedere ai record con una chiave alfanumerica (ma ciò è a carico del programmatore).

Tra gli svantaggi, invece, ricordiamo che gli archivi relative presentano un inconveniente: spesso è molto difficile riuscire a impostare una relazione univoca tra la chiave primaria e la posizione nella quale memorizzare il record. Questo inconveniente obbliga il programmatore a utilizzare chiavi legate strettamente alla posizione relativa del singolo record all'interno dell'archivio, oppure a utilizzare particolari metodi di randomizzazione.

Ricordiamo, infine, che nell'organizzazione relative la cancellazione può avvenire solo logicamente, a eccezione dell'ultimo record.

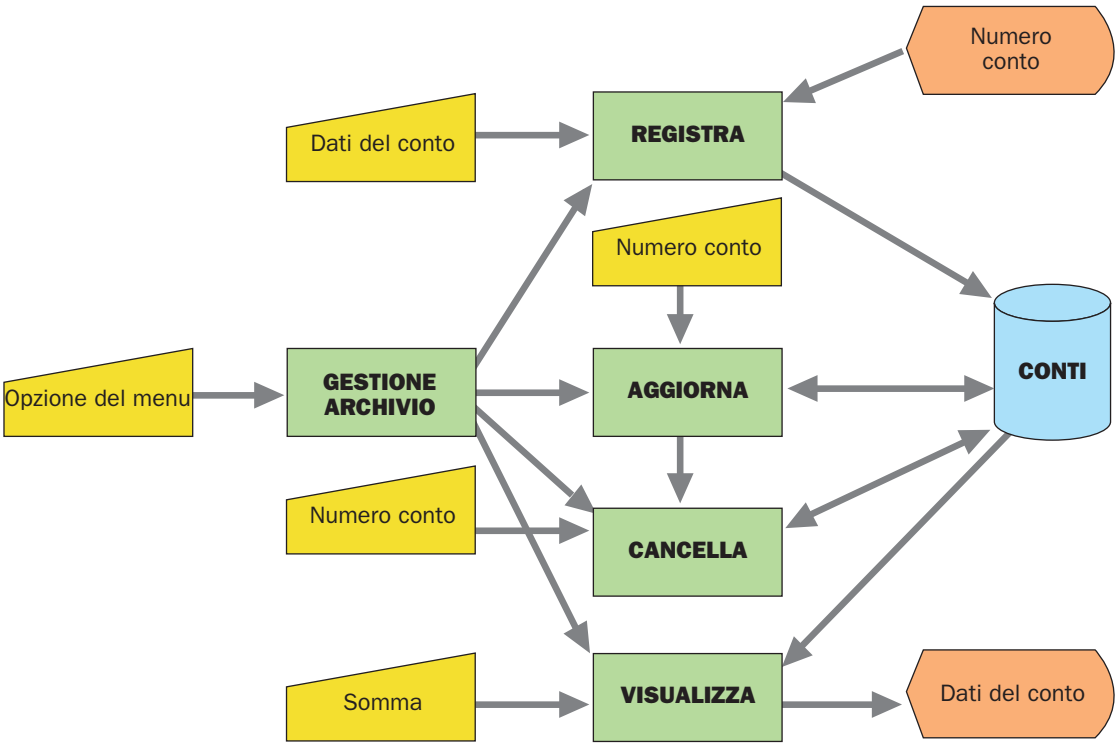
L'organizzazione hash

La **tecnica hash** si basa sull'utilizzo di funzioni che trasformano la chiave primaria di ciascun record in un numero intero, che rappresenta l'indirizzo logico, detto *indirizzo hash*, del record stesso.

Più in generale, applicare la tecnica hash significa definire una funzione di randomizzazione **H** (*funzione hash*) che associ a ogni record l'indirizzo di un record logico in cui è possibile memorizzarlo, attraverso la trasformazione della chiave **K** in un intero **X** compreso tra 1 e *N*, dove *N* è il numero di indirizzi logici che sono a disposizione.

$$H(K) = X$$

L'indirizzo X deve essere calcolato tutte le volte che occorre accedere al record di chiave K per effettuare operazioni di I/O.



LEZIONE 3

La funzione hash

La scelta della *funzione hash* assume un ruolo di fondamentale importanza, in quanto una funzione di trasformazione non idonea può rendere inefficiente o anche completamente invalida l'organizzazione stessa. Una funzione hash ottimale deve:

1. essere facilmente calcolabile, cioè composta da calcoli che siano i più semplici possibile, in modo da non appesantire il procedimento di conversione della chiave e diminuire, così, il tempo di accesso;
2. produrre sempre lo stesso indirizzo a partire dalla stessa chiave;
3. generare indirizzi uniformemente distribuiti nell'ambito dell'archivio;
4. generare indirizzi casualmente distribuiti nell'ambito dell'archivio, relativamente a chiavi simili (esempi di chiavi simili possono essere X1, X2, X3, oppure DARE, MARE, CARE);
5. coprire l'intero intervallo degli indirizzi evitando, se possibile, che ci siano indirizzi che non vengono mai generati;
6. generare indirizzi diversi per chiavi diverse.

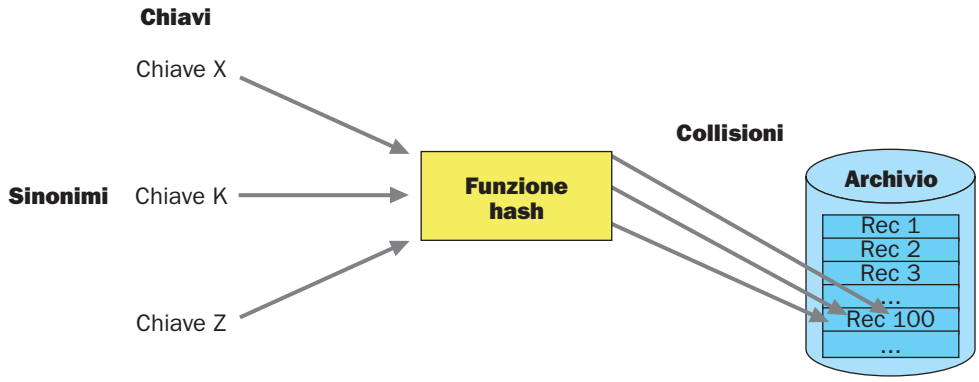
Relativamente all'ultimo punto, quindi, la funzione ideale è quella che a ogni valore della chiave primaria fa corrispondere sempre un indirizzo diverso, di modo che ogni record possa essere raggiunto tramite un solo accesso. Quando ciò accade si parla di **trasformazione perfetta**.

Supponiamo di dover memorizzare in un archivio i 1000 prodotti giacenti nel magazzino di un generico punto vendita, con un codice che varia tra 5000 e 5999. Supponiamo, inoltre, che il campo Codice Prodotto, al quale attribuiamo la funzione di chiave primaria, sia composto al massimo da quattro cifre. La funzione hash più semplice consiste nel sottrarre 5000 al valore di ogni codice e, successivamente, addizionare 1:

$$H(K) = H(\text{Codice Prodotto}) = (\text{Codice Prodotto}) - 5000) + 1$$

La funzione restituisce valori compresi tra 1 e 1000 facendo si che a ogni codice corrisponda un solo record.

La situazione presentata nell'esempio però, è ben lontana dalla realtà, in quanto è molto difficile realizzare funzioni hash che consentano una corrispondenza uno-a-uno tra chiave e indirizzo. Funzioni come quella dell'esempio, infatti, richiedono che il numero di record che compongono l'archivio fisico sia esattamente uguale al numero dei possibili valori che la chiave può assumere.
Nella realtà, il numero di chiavi effettive da gestire è sensibilmente inferiore a quello delle chiavi ipotetiche e ciò porterebbe a un notevole spreco di memoria e, di conseguenza, a un'inefficienza generale dell'intera organizzazione.
Per questi motivi si opta per altre funzioni hash che, pur non garantendo una corrispondenza uno-a-uno, cioè una trasformazione perfetta, non si discostano molto dal modello ideale. La funzione, in tal caso, produce una corrispondenza molti-a-uno, cioè accade che due o più chiavi, in fase di trasformazione, possano generare lo stesso indirizzo.



Quando ciò accade si produce una **collisione**, per risolvere la quale occorre disporre di particolari procedure che collochino le registrazioni in conflitto, dette **sinonimi**, in record di indirizzo diverso. I requisiti descritti precedentemente per la scelta della funzione hash al punto 4 e al punto 5 rappresentano la condizione necessaria per contenere il fenomeno della collisione.
A tal proposito, possiamo anche formulare un settimo requisito dicendo che scegliere una funzione hash ottimale significa scegliere una funzione che:

7. generi il minor numero possibile di collisioni e definisca apposite procedure per l'allocazione dei sinonimi (**gestione delle collisioni**).

Supponiamo di dover stabilire l'indirizzo logico corrispondente a record aventi le seguenti chiavi (BOLOGNA, LECCO, OTRANTO, BAGNOLO) per un archivio composto al massimo da 200 record. Per allocare le suddette chiavi ci serviamo del seguente metodo:

1. si associa a ogni carattere della chiave il corrispondente valore ASCII;
2. si sommano tutti i valori ottenuti;
3. si moltiplica tale somma per il numero di cifre che compongono il nome della chiave;
4. si calcola il resto della divisione del numero ottenuto per il numero di record inseribili, cioè 200, e si somma + 1, ottenendo il valore dell'indirizzo.

Cominciamo con la prima chiave: BOLOGNA

66 + 79 + 76 + 79 + 71 + 78 + 65 = 514

514 x 7 = 3598

(3598 MOD 200) + 1 = 198 + 1

B	O	L	O	G	N	A
66	79	76	79	71	78	65

Seconda chiave: LECCO

76 + 69 + 67 + 67 + 79 = 358

358 x 5 = 1790

1790 MOD 200 = 190 + 1

L	E	C	C	O
76	69	67	67	79

Terza chiave: OTRANTO

Collisione

79 + 84 + 82 + 65 + 78 + 84 + 79 = 551

551 x 7 = 3857

3857 MOD 200 = 57 + 1

O	T	R	A	N	T	O
79	84	82	65	78	84	79

Quarta chiave: BAGNOLO

66 + 65 + 71 + 78 + 79 + 76 + 79 = 514

514 x 7 = 3598

(3598 MOD 200) + 1 = 198 + 1

B	A	G	N	O	L	O
66	65	71	78	79	76	79

LEZIONE 4

L'organizzazione hash: calcolo degli indirizzi

Assodato che è quasi impossibile realizzare funzioni che consentano una trasformazione perfetta e che, di conseguenza, il problema delle collisioni sarà sempre presente (indipendentemente dalla funzione), occorre concentrare le nostre forze sullo sviluppo di funzioni efficienti, piuttosto che perfette. Esistono molti metodi per calcolare gli indirizzi; esaminiamone solo alcuni.

Il metodo del troncamento

Questo metodo viene utilizzato quando la chiave numerica è composta da un elevato numero di cifre e il numero di record che possono essere memorizzati è abbastanza limitato. Consiste nell'estrarre dalla chiave un sottoinsieme di cifre che rappresentano l'indirizzo.

Supponiamo di avere un archivio che contiene informazioni sui tesserati di una palestra e sul quale possono essere memorizzati 1000 record partendo dall'indirizzo 1. Il ruolo di chiave primaria viene attribuito al numero di tessera di ogni socio, che è composto da un numero di sette cifre. Dobbiamo costruire una corrispondenza che a ogni valore della chiave di sette cifre associ un indirizzo di tre cifre, ossia da 000 a 999. La funzione hash, pertanto, sarà ottenuta eliminando dalla chiave le prime quattro cifre e sommando 1:

H(1548968) + 1 = 969
H(0003254) + 1 = 255
H(3293001) + 1 = 002
H(6331254) + 1 = 255

Sinonimi

Questo semplicissimo metodo è molto efficiente nel caso in cui le chiavi numeriche siano essenzialmente consecutive e con poche interruzioni. Gli svantaggi di tale funzione risiedono nel fatto che essa produce sinonimi ogni volta che si incontrano chiavi aventi le ultime tre cifre uguali e che il numero di indirizzi logici a disposizione deve essere sempre una potenza di 10. Ciò richiede (come spesso accade) che l'archivio debba essere composto da un numero di record superiore a quello effettivamente necessario. Riferiamoci all'esempio precedente: se la palestra ha solo 130 soci, deve prevedere sempre un archivio composto da 1000: è utile sovradimensionare l'archivio (magari prevederne 160), ma predisporre 1000 record è, ovviamente, un enorme spreco di memoria! Applichiamo questo metodo a una chiave alfanumerica. Supponiamo che la chiave da trasformare debba essere composta da quattro caratteri ottenuti estraendo gli ultimi quattro caratteri della chiave primaria. Per procedere alla trasformazione, si può utilizzare il semplicissimo metodo che pone in corrispondenza le 26 lettere dell'alfabeto con i primi 26 numeri naturali, proprio come riportato nella tabella riportata a lato.

A	0
B	1
C	2
...	...
...	...
Z	25

A questo punto, proprio come nei sistemi di numerazione pesati, si procede alla costruzione di una funzione che calcoli la sommatoria dei prodotti di ogni carattere per il suo peso, ossia per 26 elevato alla posizione occupata all'interno della parola. Proviamo a calcolare l'indirizzo, o meglio, il numero corrispondente alla chiave HELLOCIAO.
Estraiamo gli ultimi quattro caratteri dalla chiave HELLOCIAO:

$$\begin{aligned} CIAO &= C \times 26^3 + 1 \times 26^2 + A \times 26^1 + O \times 26^0 = \\ &= 2 \times 26^3 + 8 \times 26^2 + 0 \times 26^1 + 14 \times 26^0 = \\ &= 2 \times 17576 + 8 \times 676 + 0 + 14 \times 1 = \\ &= 35152 + 5408 + 14 = \\ &= 40574 \end{aligned}$$

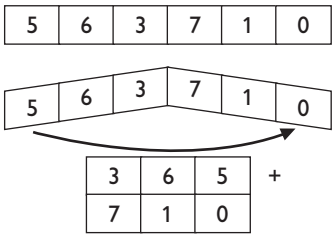
A questo punto, per calcolare il reale indirizzo, si può applicare il metodo del troncamento, ma, per farlo, il programmatore deve aver stabilito a priori qual è il numero massimo di record che può contenere l'archivio, in modo da poterlo approssimare in modo migliore. Come per il caso di una chiave numerica, anche in

questo caso lo svantaggio principale deriva da un notevole spreco di memoria e dalla possibilità di generare diverse collisioni, visto che esistono diverse parole che possono iniziare o terminare con le stesse lettere.

Il metodo folding

Questo metodo consiste nel dividere la chiave numerica in due o più parti. Successivamente, tali parti vengono sommate tra loro e il valore ottenuto (o una sua parte), viene utilizzato come indirizzo. Il nome **folding** (termine inglese che significa "piegare") si spiega se si immagina di piegare la chiave congiungendo le sue estremità, proprio come se fosse scritta su un foglio di carta, e di sommare tra loro le cifre che ricadono nella stessa posizione. Se la somma ottenuta supera il numero massimo di cifre di cui deve essere composto l'indirizzo, si procede al troncamento delle prime cifre.
Supponiamo che la chiave sia 563710:

$$H(563710) = 365 + 710 = \mathbf{1075}$$



Graficamente abbiamo quanto è riportato a lato.
Il valore dell'indirizzo è, quindi, **1075**. Anche questo metodo genera diverse collisioni, poiché, ad esempio, H(563710) = H(349132). Inoltre, presenta gli stessi svantaggi del metodo di troncamento poiché, anche in questo caso, all'indirizzo viene riservato uno spazio pari a una potenza di 10. Ribadiamo, infine, che la scelta di suddividere la chiave in due o più parti è in relazione alla specifica applicazione. Tale suddivisione deve essere attentamente valutata, al fine di consentire una distribuzione uniforme degli indirizzi, per ridurre il più possibile il numero delle collisioni.

Il metodo della divisione modulo N

Con questo metodo l'indirizzo del record si ottiene dal resto della divisione della chiave numerica per il numero massimo di record memorizzabili nell'archivio. In generale, se indichiamo con k la chiave e con N il numero massimo di record, l'indirizzo del record è dato da:

$$H(K) = K \text{ MOD } N$$

Sappiamo che l'operatore modulo N fornisce sempre valori compresi tra 0 ed N – 1. Se vogliamo che la funzione restituisca indirizzi Hash compresi tra 1 ed *N* dobbiamo apportare alla funzione la seguente modifica:

$$H(K) = (K \text{ MOD } N) + 1$$

Anche questo metodo ha generato dei sinonimi, ma questo metodo presenta il vantaggio di poter essere applicato anche in presenza di un numero massimo di record che non è multiplo di 10. Gli indirizzi generati, proprio per la definizione di modulo, non necessitano di troncamento. Per quanto riguarda i sinonimi, la scelta del valore N del modulo è il punto fondamentale per ridurre al minimo la loro presenza.
Supponiamo di dover memorizzare le seguenti dieci chiavi in un archivio composto, al massimo, da 20 record:

92 64 21 80 12 13 9 88 11 58

Chiave K	H(K) = (K mod 20) + 1
92	12 + 1 = 13
64	4 + 1 = 5
21	1 + 1 = 2
80	0 + 1 = 1
12	12 + 1 = 13
13	13 + 1 = 14
9	9 + 1 = 10
88	8 + 1 = 9
11	11 + 1 = 12
58	18 + 1 = 19

LEZIONE 5

L'organizzazione hash: gestione delle collisioni

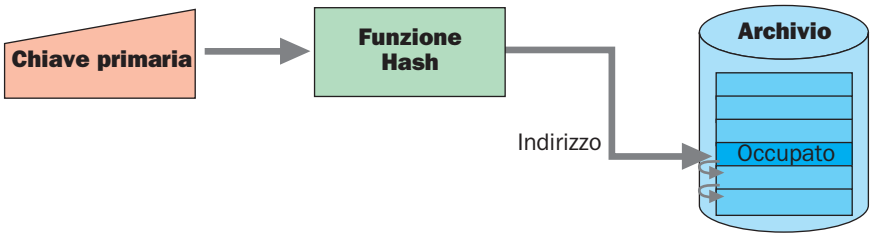
Anche ponendo il massimo scrupolo nella scelta della funzione hash, è praticamente impossibile evitare il verificarsi delle collisioni.

Le **tecniche di gestione delle collisioni** sono metodi che consentono di individuare un indirizzo libero per la chiave K nel caso in cui la funzione hash H(K) restituisca un valore già ottenuto per un'altra chiave, cioè l'indirizzo logico di un record già occupato.

La gestione corretta di questa specifica situazione riveste particolare importanza sia nella fase di inserimento dei record, sia in quella di ricerca di una chiave. Nel caso dell'inserimento, si applica a ogni chiave da inserire nell'archivio uno dei metodi descritti in precedenza (o altri ancora). Ottenuto il valore dell'indirizzo, si effettua un test nella posizione indicata per controllare se è occupata (collisione) o meno. Se non è occupata, il record può essere inserito, mentre se il posto non è libero va ricercata una nuova posizione, sino a quando non se ne incontra una libera. Per la ricerca il discorso è analogo: si applica la funzione alla chiave da ricercare e si accede al record indicato dal valore restituito dalla funzione. Se la chiave in esso contenuta coincide con quella cercata, il problema è risolto, altrimenti si esegue la scansione dell'archivio andando su altre posizioni, sino a quando non si trova la chiave cercata. Come per la funzione hash, esistono vari metodi anche per gestire le collisioni: ne esaminiamo alcuni.

Scansione lineare o indirizzamento aperto

Quando la funzione hash produce un indirizzo già occupato da un altro record, occorre effettuare la scansione dell'archivio in modo da ricercare il primo posto libero in cui sistemare la chiave che ha generato la collisione. La tecnica della **scansione lineare**, detta anche dell'**indirizzamento aperto**, consiste proprio in questo. Se, ad esempio, la collisione avviene sull'indirizzo 15 e poniamo come passo di scansione 3, occorre esaminare i record di indirizzo 18, 21, 24, 27 sino al reperimento di un posto vuoto. Se, in particolare, **p = 1** si parla di **scansione lineare con passo unitario**. La scelta del passo unitario assicura il controllo di tutti gli indirizzi, in quanto, se la chiave collide su un indirizzo, si va a controllare la posizione immediatamente successiva, quella successiva ancora e così via. Il processo termina quando si trova un posto libero, oppure quando è stata esaminata metà delle posizioni presenti all'interno dell'archivio (in tal caso viene comunicato un messaggio di archivio pieno). La scansione non si interrompe quando si incontra la fine dell'archivio, ma riprende dalla prima posizione.



Questo metodo presenta un inconveniente, detto **agglomerazione** o **addensamento primario delle chiavi** (***clustering***), cioè la presenza di aree di memoria (**agglomerati** o ***cluster***) nelle quali confluiscono varie sequenze di sinonimi, che presentano indirizzi coincidenti da un certo punto in poi (creando lunghe catene di record adiacenti, con conseguente significativo rallentamento del tempo di accesso ai record).
Supponiamo di memorizzare le seguenti chiavi in un archivio composto al massimo da 9 record:

9 28 29 35 24 48 71 64

Per generare gli indirizzi ci serviamo del metodo della divisione modulo 9. Abbiamo:

9 → (9 mod 9) + 1 = 0 + 1 = 1 la chiave 9 viene memorizzata in posizione 1
28 → (28 mod 9) + 1 = 1 + 1 = 2 la chiave 28 viene memorizzata in posizione 2
29 → (29 mod 9) + 1 = 2 + 1 = 3 la chiave 29 viene memorizzata in posizione 3
35 → (35 mod 9) + 1 = 8 + 1 = 9 la chiave 35 viene memorizzata in posizione 9
24 → (24 mod 9) + 1 = 6 + 1 = 7 la chiave 24 viene memorizzata in posizione 7
48 → (48 mod 9) + 1 = 3 + 1 = 4 la chiave 48 viene memorizzata in posizione 4
71 → (71 mod 9) + 1 = 8 + 1 = 9 collisione con la chiave 35.

A questo punto, in base alla tecnica della scansione lineare con passo unitario, dobbiamo spostarci sul record successivo per vedere se la posizione è libera: dobbiamo andare sulla posizione 10. Ciò, ovviamente, non è realizzabile in quanto abbiamo supposto che l'archivio sia composto al massimo da 9 record. Occorre, pertanto, applicare la divisione anche all'indirizzo prodotto. Abbiamo:

71 → (71 mod 9) + 1 = 8 + 1 = 9 collisione con la chiave 35
(9 mod 9) + 1 = 0 + 1 = 1 collisione con la chiave 9
(1 mod 9) + 1 = 1 + 1 = 2 collisione con la chiave 28
(2 mod 9) + 1 = 2 + 1 = 3 collisione con la chiave 29
(3 mod 9) + 1 = 3 + 1 = 4 collisione con la chiave 48
(4 mod 9) + 1 = 4 + 1 = 5 la chiave 71 viene memorizzata in posizione 5.

Ora analizziamo l'ultima chiave:

64 → (64 mod 9) + 1 = 1 + 1 = 2 collisione con la chiave 28.

Poiché l'indirizzo 2 è interno all'intervallo, si cominciano a esaminare una a una le successive posizioni, sino a incontrare un posto libero.
Vengono quindi scandite le posizioni 3, 4, 5 e, infine, la posizione 6, sulla quale la chiave 64 viene memorizzata. Come si può constatare, la chiave 64 ha dovuto attraversare una parte della catena precedente, anche se i record individuati da questi indirizzi erano occupati.

Scansione non lineare o metodo pseudo-random

Il metodo che stiamo per descrivere rappresenta una variante dell'indirizzamento aperto e, come tale, nasce anch'esso allo scopo di eliminare o di tamponare il problema dell'agglomerazione. Per farlo, questo metodo ricorre a una tecnica di memorizzazione che non prevede una scansione sequenziale dell'archivio, poiché, invece di utilizzare il passo di scansione, genera un numero casuale (o, meglio, pseudocasuale) ogniqualvolta si verifica una collisione. Tale metodo si definisce **scansione non lineare** o ***pseudo-random***. In altri termini, si calcola l'indirizzo per iniziare la scansione: se su tale indirizzo si verifica una collisione, non ci si sposta nella posizione immediatamente successiva bensì si ricalcola un nuovo indirizzo, detto indirizzo di rehash, servendosi di una nuova funzione, detta **funzione di rehash**. L'**indirizzo di rehash** può essere calcolato con una qualsiasi funzione che generi numeri pseudocasuali e che, per ogni passo, definisca un incremento differente per l'indirizzo.
Circa la casualità, ribadiamo che l'indirizzo di rehash deve essere necessariamente pseudocasuale e non puramente casuale, in quanto, in fase di ricerca di una chiave, devono essere ricalcolati gli stessi indirizzi prodotti per la memorizzazione, ossia deve essere ripetuta la stessa sequenza di indirizzi.