

Tuning delle strutture fisiche, esempi (Shasha)

Employee (SSN, Name, Dept, Manager, Salary)

Student(SSN, Name, Course, Grade, Stipend,WrittenEvaluation)

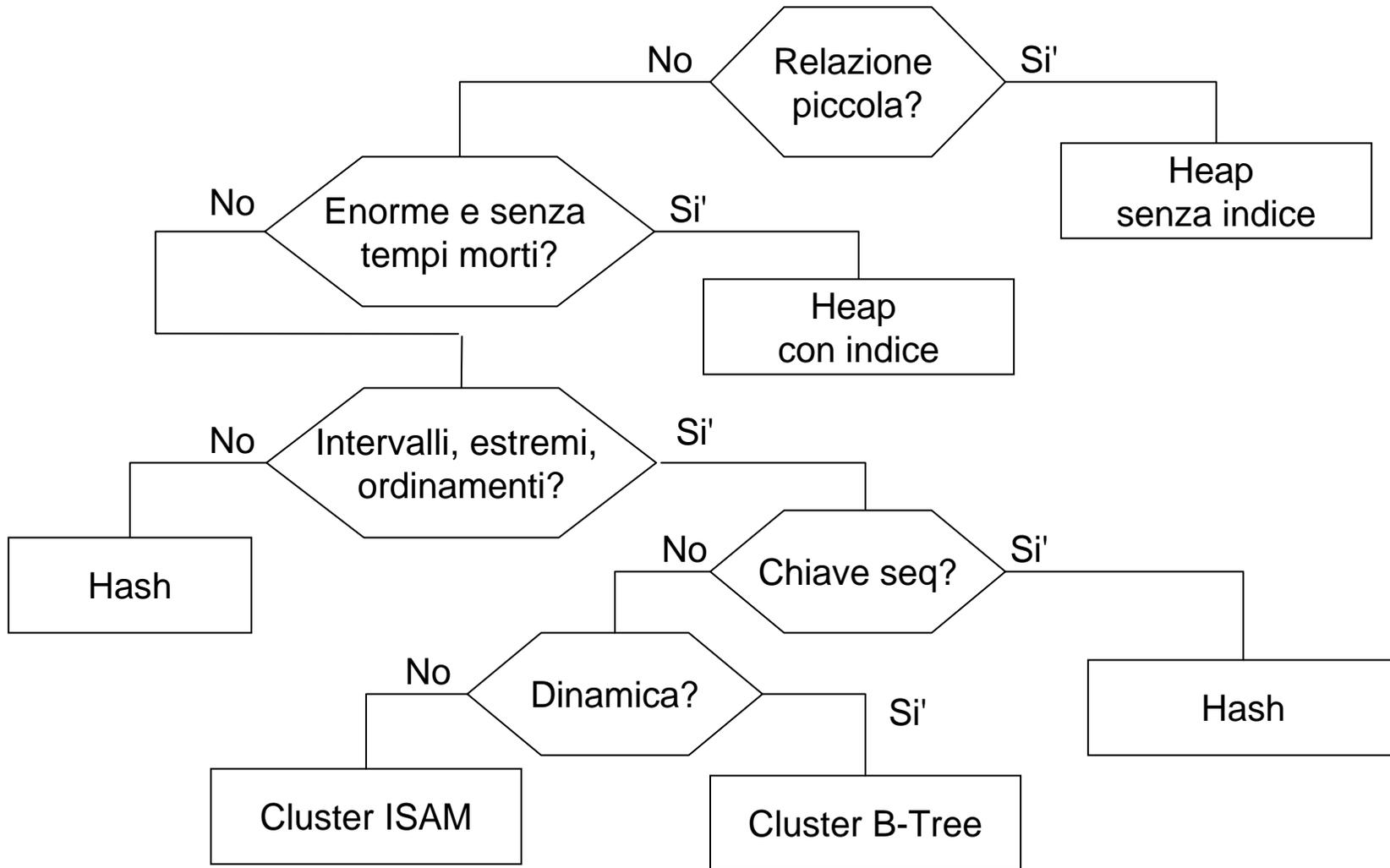
dal testo:

D. Shasha.

Database Tuning: a principled approach.

Prentice-Hall, 1992

Scelta della struttura secondo Shasha



Progettazione fisica: euristiche suggerite da Informix

- Non creare indici su relazioni piccole (<200 ennuple)
- non creare indici su campi con pochi valori (se proprio servono, che siano primari)
- creare indici su campi con selezioni
- per i join: creare indici sulla relazione più grande

Caso 1

- Sulla relazione Student viene definito un indice, ma l'interrogazione seguente non lo usa:

```
SELECT *  
FROM Student  
WHERE Name = 'Bayer'
```

Possibile motivazione

- Le statistiche potrebbero non essere aggiornate, e quindi l'ottimizzatore sceglie di non usare l'indice, ritenendo la relazione piccola

Caso 2

- Sull'attributo Salary della relazione Employee viene definito un indice, ma l'interrogazione seguente non lo usa, anche dopo l'aggiornamento delle statistiche:

```
SELECT *  
FROM Employee  
WHERE Salary/12 = 4000
```

Possibile motivazione

- L'indice non viene utilizzato perché c'è l'espressione aritmetica

Caso 3

- Sull'attributo Dept della relazione Employee viene definito un indice, l'interrogazione seguente lo usa, ma senza beneficio:

```
SELECT *  
FROM Employee  
WHERE Dept = 12
```

Possibile motivazione

- Probabilmente ci sono molti impiegati nel dipartimento 12 e l'indice è secondario. L'interrogazione deve quindi accedere a molte pagine (magari quasi tutte).

Caso 4

- La relazione Studente ha una struttura ordinata con indice su SSN e quando si modifica il valore di "WrittenEvaluation"
 - si generano liste di overflow, oppure
 - la risposta è molto lenta

Possibile motivazione

- WrittenEvaluation ha probabilmente lunghezza variabile. La struttura ordinata è poco flessibile in questo caso. Probabilmente conviene una struttura disordinata con indice secondario su SSN

Casi 5 e 6

- La relazione impiegato ha un fattore di blocco (numero di ennuple per blocco) pari a 30. Ogni impiegato (si vede dallo schema) afferisce ad un dipartimento.
- Caso 5: Ci sono 20 dipartimenti diversi.
- Caso 6: Ci sono 2000 dipartimenti diversi.
- Conviene avere un indice secondario su Dept?

Risposta

- Nel caso 5 no, perché ogni pagina contiene mediamente impiegati di tutti o quasi i dipartimenti, e quindi un accesso con indice non è più efficiente di una scansione sequenziale, anzi.
- Nel caso 6 sì, perché ogni pagina contiene solo record di al più 30 dipartimenti su 2000

Caso 7

- Viene effettuata una copia della relazione Employee, per eseguire su di essa interrogazioni fuori linea (e nessun aggiornamento), in particolare:
 - 1 Contare gli impiegati con un certo stipendio (frequente)
 - 2 Trovare gli impiegati con lo stip max in un dip (frequente)
 - 3 Trovare un impiegato dato il SSN (un po' meno frequente)
- Quali strutture fisiche?

Possibile soluzione

- un indice secondario potrebbe essere più efficiente per (1) perché denso
- per (2) serve un indice su Dept, Salary (secondario va bene)
- per (3) un indice primario può essere meglio di un secondario

Caso 8

- Stipend (in Student) è retribuzione mensile, mentre Salary (in Employee) è annuale. Vogliamo trovare impiegati e studenti con la stessa retribuzione. Che differenza c'è (quale conviene?) fra:

```
SELECT *  
FROM Employee, Student  
WHERE Salary =12*Stipend
```

```
SELECT *  
FROM Employee, Student  
WHERE Salary/12 = Stipend
```

Caso 8, risposta

```
SELECT *  
FROM Employee, Student  
WHERE Salary =12*Stipend
```

```
SELECT *  
FROM Employee, Student  
WHERE Salary/12 = Stipend
```

Indici e espressioni, vedi caso (2):

- un indice su Stipend potrebbe non essere usato nella prima interrogazione (su Salary sì)
- un indice su Salary potrebbe non essere usato nella seconda interrogazione (su Stipend sì)
- Quindi se c'è un indice solo, abbiamo una risposta
- Se ci sono entrambi, se uno è primario, conviene farlo usare
- Se sono entrambi secondari, conviene far usare quello sulla relazione più grande (salvo in caso in cui esso sia poco selettivo)

Caso 9

Onorder(Supplier,Part,Quantity,Price)

Operazioni

- 1 Inserimento (molto frequente)
 - 2 Eliminazione, dati Supplier e Part (molto frequente)
 - 3 Trovare la quantità totale degli ordini di una Part (frequente)
 - 4 Trovare l'importo totale degli ordini di un fornitore (rara)
- Quali strutture fisiche?

Possibile soluzione

- Se ci sono tempi morti:
 - indice primario su Part e Supplier (hash non va bene per 3)
- altrimenti, secondario

Caso 10

Onorder(Supplier,Part,Quantity,Price)

Operazioni sola lettura

- 3 Trovare la quantità totale degli ordini di una Part (frequente)
- 4 Trovare l'importo totale degli ordini di un fornitore (meno frequente)

- Quali strutture fisiche?

Possibile soluzione

- Indice primario su Part e Supplier
- Indice secondario su Supplier

Caso 11

Archivio clienti di una carta di credito. Operazioni:

- 1 Inserimento nuovo cliente (frequente)
 - 2 Trovare un cliente dato il codice fiscale (frequente)
 - 3 Trovare un cliente dato il "ClientNumber" (meno frequente)
 - 4 Scandire l'archivio per intero (rara)
- C'è una struttura primaria hash sul codice fiscale e un indice secondario B-tree sul "ClientNumber" (che viene generato sfruttando le primitive sui contatori). Le transazioni però sono molto lente.

Possibile motivazione

- l'ultima pagina del B-tree è un collo di bottiglia; se il sistema la prevede su può usare struttura hash secondaria (cioè struttura hash con un livello aggiuntivo di puntatori)

Caso 12

- Una relazione ha un indice B-tree sul codice fiscale e le operazioni principali sono ricerche e aggiornamenti su ennuple individuate sulla base del codice fiscale.
- Se le prestazioni non sono soddisfacenti, come provare a migliorarle?

Possibili soluzioni

- B-tree ordinato (in alcuni casi)
- hash

Caso 13

- Una relazione disordinata risulta inefficiente in caso di molti inserimenti concorrenti
- Se le prestazioni non sono soddisfacenti, come provare a migliorarle?

Possibili soluzioni

- Essendo la relazione disordinata, gli inserimenti sono sempre nell'ultimo blocco, su cui i lock generano ritardi. Quindi:
 - si può usare una struttura hash che "sparpagli"
 - usare lock a livello di record
 - riorganizzare le applicazioni, per ridurre i picchi di inserimenti (può non essere possibile)
 - raggruppare più inserimenti in ciascuna transazione

Caso 14

- Ellis Island gestisce l'archivio dei milioni di immigrati negli USA dell'800 e primo 900.
- Per ogni immigrato ci sono molti campi, e viene offerto un servizio che dato cognome (talvolta anche nome) e anno di arrivo fornisce le altre info.
- Che struttura fisica?

Possibili soluzioni

- Abbiamo solo lettura, quindi soluzioni statiche ok
- Indice primario (anche ISAM) su cognome e nome
- forse un indice secondario su cognome e anno
- altri indici probabilmente poco selettivi (ma comunque valutabili, perché senza costi di aggiornamento)