Progettazione fisica / 12-05

L'obiettivo del dato un carico di lavoro progetta uno schema fisico (indici) che rende il più efficiente possibile l'esecuzione (su quali tabelle creare indici e di che tipo)

La progettazione diventa tuning quando si modifica il livello fisico di un DB già creato (monitoraggio del DB)

Scelte sbagliate di uno schema fisico può portare a esecuzioni lunghe e a sprechi di memoria

E' utile creare indici che vengano sfruttati per più query

Carico di lavoro (workload) $W = \{Q_1, \dots, Q_n\}$
insieme di operazioni
$= \{Q_1, \dots, Q_n\}$
consideriamo insieme di query

Per ogni query $Q_i \in W$ determinare lo schema SF_i fisico che riteniamo migliore per eseguire Q_i
→ ottenibile tramite euristiche
} approccio "CREATE INDEX" (scelgo lo schema fisico)

$S_0 = \{\}$ schema fisico iniziale vuoto

$S_i = S_{i-1} \oplus SF_i$ integro lo schema fisico di una certa query con gli schemi utilizzati in precedenza

Return S_n

Alla creazione di SF: devo specificare che indice voglio creare, su cosa lo creo e se è clusterizzato.

⊕ non è un'unione!

Se ho un indice clusterizzato su un attributo non posso crearne un'altro clusterizzato su un secondo attributo

Nota: Postgre gestisce automaticamente gli indici sulle query

Euristiche

- Evitare di creare indici su tabelle piccole (che sarebbero caricate interamente nel buffer)
- Per selezioni e join di uguaglianza un indice hash può garantire prestazioni migliori, tenendo conto che se c'è un overflow conviene quello ad albero
- Se selezioni e join su intervalli vale solo indice ad albero
- Se conviene creare un solo indice su tabella conviene clusterizzarlo.
- Al più un indice clusterizzato per tabella (file di dati)
  → conviene x diversi di ricerca dove la tante tuple restituite (selettività più bassa)
  → cercare di favorire più query

- Indici multi attributo pesano di più

Per equi-join
- se indici ordinati su attr. di join → index nested loop
- se indici clusterizzati → merge join
- in assenza di indice il sistema fa hash join

Non equi-join: solitamente index nested loop