Tecnologia delle basi di dati (ex Basi di dati, primo modulo)

14 giugno 2006 — Cenni sulle soluzioni

Si fa riferimento al compito A, ma le osservazioni valgono anche per gli altri.

Domanda 2

Si richiamano le tre formulazioni

1. `CREATE INDEX contoCorrenteIX ON contoCorrente (numero) INCLUDE (saldo)`
2. `CREATE INDEX contoCorrenteIX ON contoCorrente (numero)`
3. `CREATE INDEX contoCorrenteIX ON contoCorrente (numero, saldo)`

La (1) favorisce le operazioni che prevedano ricerche su `numero` con accesso ai valori di `saldo`; se però sono necessari valori di altri attributi, non si ha nessun beneficio. Poiché `numero` è chiave, la soluzione (3) è sostanzialmente equivalente, da questo punto di vista, a parte piccole differenze sull'aggiornamento. Comunque, la soluzione (3) può essere vista come soluzione di "ripiego," nel caso in cui non sia disponibile la (1). La soluzione (2) favorisce l'accesso diretto su `numero` ma richiede l'accesso al file per il recupero dei valori sugli altri campi. Le soluzioni (1) e (3) penalizzano gli aggiornamenti (la (3) un po' di più, anche se non moltissimo, perché `numero` è chiave; se non lo fosse, la differenza sarebbe maggiore).

Domanda 3

Dimostrare che due schedule conflict-equivalenti hanno lo stesso grafo dei conflitti, mentre il viceversa non è necessariamente vero.

- dati due schedule S_1 ed S_2 conflict equivalenti, dimostriamo che hanno lo stesso grafo.
  Procediamo per "contrapposizione": supponiamo che non abbiano lo stesso grafo e mostriamo che non sono conflict equivalenti. Se non hanno lo stesso grafo allora c'è un arco, nel grafo di uno schedule non presente nell'altro; supponiamo che il grafo di S_1 abbia tale arco; esso è relativo a due azioni in conflitto; ma S_1 ed S_2 hanno le stesse azioni che, se sono in conflitto in S_1, allora sono in conflitto anche in S_2; se l'arco non compare nel grafo di S_2, allora le due azioni sono in S_2 in ordine diverso da quello presente in S_1 e quindi non è vero che S_1 ed S_2 sono conflict equivalenti.
- basta mostrare un controesempio, quale:
  - $S_1 : w_1(x)w_2(x)w_2(y)w_1(y)$
  - $S_2 : w_2(x)w_1(x)w_1(y)w_2(y)$

Domanda 4

Si ricorda che la vista

- `CREATE VIEW V AS SELECT * FROM (R1 LEFT JOIN R2 ON B=D) JOIN R3 ON C=G`

non viene necessariamente calcolata tutta: il processo di ottimizzazione viene applicato alla espressione ottenuta sostituendo la definizione della vista ad ogni sua occorrenza.

1. `SELECT A, L FROM V`
   - poiché il join su R_2 è esterno, esso non influisce sul risultato e viene omesso; quindi viene eseguito solo il join $R_1 \bowtie_{C=G} R_3$ (si può ragionare sulle proiezioni per ridurre i risultati intermedi, ma non è molto importante)
   - per quanto riguarda i costi, si può pensare ad un nested-loop con accesso diretto alla tabella interna e quindi, ignorando i buffer, si ha un numero di accessi a blocchi stimabile come segue (dove LIV_3 è il numero di livelli dell'indice di R_3 e s è la frazione di valori di C che compare in $R_3.G$, che approssimiamo a 1, in assenza di informazioni):

     $$S_1/(B/l_1) + S_1(\text{LIV}_3 + s) = \text{ ca } 1.000.000/(1.000/40) + 1.000.000(1+3) = \text{ ca } 4.000.000$$

2. `SELECT A FROM V`
   - per gli stessi motivi del caso precedente, il join su R_2 può essere omesso; invece quello su R_3 è necessario (per verificare l'appartenenza delle ennuple al risultato); non sarebbe stato necessario se ci fosse stato il vincolo di integrità referenziale fra C di R_1 ed R_3; (nota, per curiosità: non serve join, ma basta una scansione dell'indice di R_3, senza accesso ai blocchi di R_3 stessa);
3. `SELECT A, E FROM V`
   - servono entrambi i join.

(segue)

Domanda 5
Schema dimensionale

- FattiVendite(KCliente, KData, KNegozio, KArticolo, Quantità, Incasso)
- Articoli(KArticolo, CodArticolo, Descrizione, CodMarca, NomeMarca, CodNazioneMarca, NazioneMarca)
- Clienti(KCliente, CFCliente, Cognome, Nome, DataNascita, FasciaEtà, CodCategoriaCliente, DescrizioneCategoriaCliente, CittàResidenzaCliente, SiglaProvinciaResidenzaCliente, RegioneResidenzaCliente)
- Negozi(KNegozio, CodNegozio, Nome, Indirizzo, CittàNegozio, SiglaProvinciaNegozio, RegioneNegozio)
- Data(KData, ...)

Commenti:

- sono indicate chiavi ad hoc per le dimensioni
- il prezzo non deve comparire nella dimensione Prodotto, perché varia nel tempo, e i valori relativi contribuiscono alla misura Incasso
- la tabella FattiVendite viene calcolata con un join su Vendite e ElementiVendite, con raggruppamenti sulle chiavi delle quattro dimensioni (gestendo opportunamente le riconversioni delle chiavi stesse)
- le tabelle dimensioni sono ottenute attraverso join che denormalizzano (e omettono attributi non significativi)
- la variazione nel tempo di Residenza e FasciaEtà potrebbe essere gestita introducendo nuove ennuple nella dimensione Persone in occasione di ogni variazione (ovviamente questo rende indispensabile l'utilizzo di chiavi ad hoc); una soluzione alternativa, forse meno elegante, ma ragionevole, avrebbe potuto prevedere dimensioni separate per Residenza (con i dettagli) e FasciaEtà
- i dati personali dei clienti (Cognome, Nome e DataDiNascita) potrebbero essere omessi