Basi di dati II, primo modulo — Tecnologia delle basi di dati

1 luglio 2009 — Compito A

Cenni sulle soluzioni

Tempo a disposizione: un'ora e trenta minuti. Rispondere in modo ordinato evidenziando bene la "bella copia" e, in essa, le risposte alle singole domande. Consegnare comunque tutti i fogli protocollo e il testo.

Domanda 1 (20%) Illustrare brevemente (ma con chiarezza) le motivazioni che portano, nei datawarehouse, a preferire l'utilizzo di identificatori ad-hoc, generati nella fase di caricamento, rispetto all'uso di chiavi provenienti dalle applicazioni.

Domanda 2 (20%) Si consideri una relazione `R(Matricola,Cognome,Nome,DataNascita)`. Spiegare perché una struttura hash sull'attributo `DataNascita` non è la più indicata se l'operazione più importante è la ricerca delle persone nate in un certo intervallo temporale.

Discussione

Perché sarebbe necessario effettuare una ricerca per ciascuna data contenuta nell'intervallo. Quindi la struttura sarebbe efficiente solo per intervalli molto piccoli.

Domanda 3 (20%) Si consideri una forma di equivalenza fra schedule denominata *final-state*-equivalenza, secondo la quale due schedule S_1 e S_2 sono equivalenti se, per ogni istanza d della base di dati, essi la trasformano nello stesso modo (e quindi $S_1(d) = S_2(d)$, se con $S(d)$ indichiamo l'istanza della base di dati ottenuta applicando a d le operazioni dello schedule S).

1. Formulare una definizione per questa proprietà, variante della definizione di view-equivalenza.
2. Spiegare, almeno intuitivamente, il rapporto che esiste fra final-state-equivalenza e view-equivalenza (sono equivalenti, incomparabili, oppure una implica l'altra?).

Due schedule sono view-equivalenti se trasformano la base di dati nello stesso modo e offrono all'esterno gli stessi valori. La final-state-equivalenza non tiene conto di questa seconda proprietà. Gli schedule $w_1(x), r_2(x), w_3(x)$ e $w_1(x), w_3(x), r_2(x)$ sono final-state-equivalenti ma non view-equivalenti (perché hanno diversa relazione "legge-da"). Una definizione formale di final-state-equivalenza potrebbe richiedere (i) stesse scritture finali; (ii) stessa relazione "legge-da" per le transazioni che scrivono dopo la lettura. Di conseguenza, la final-state-equivalenza è condizione necessaria, ma non sufficiente per la view-equivalenza.

Domanda 4 (20%) Si supponga di dover realizzare un data mart relativo ad esami universitari. Fra i vari dati di interesse vi sono i voti riportati negli esami. Spiegare quali differenze possono sussistere nel data mart fra il caso in cui interessa ragionare sul voto medio e quello in cui interessa ragionare sulle distribuzioni per fasce di voto (cioè, ad esempio, quanti voti fra 18 e 23, quanti fra 24 e 29 e così via). Spiegare anche come procedere se interessano entrambi gli aspetti. Spiegare anche come si potrebbe gestire una situazione in cui le fasce di interesse varino (ad esempio, una nuova rilevazione desidera conoscere quanti voti sono compresi fra 18 e 21, quanti fra 22 e 25 e così via).

Discussione

Per ragionare sul voto medio, il voto (singolo o medio, sulla base della grana scelta, che dipende da altri fattori, qui non citati) deve essere una misura. Per ragionare sulle fasce di voto, voto (o fascia) deve essere una dimensione. La modifica delle fasce di interesse è gestibile solo se, fin dall'inizio, gli elementi della dimensione corrispondono ai singoli voti; in tal caso, basta aggiungere un attributo alla dimensione; altrimenti, è possibile gestire la nuova situazione solo ricalcolando i dati nel data mart.

Tecnologia delle basi di dati — 1 luglio 2009 — Compito A

Domanda 5 (20%) Un *quad tree*, nella sua versione più semplice, è un indice definito con riferimento a due diversi attributi di una relazione. Siano A e B gli attributi in questione. La struttura è costituita da un albero in cui ogni nodo intermedio ha due etichette, una per ciascuno degli attributi, e quattro sottoalberi. Se i valori di un nodo sono (a, b), allora

- il primo sottoalbero contiene riferimenti ai record con valori di A minori di a e valori di B minori di b
- il secondo ai record con valori di A minori di a e valori di B maggiori o uguali a b
- il terzo ai record con valori di A maggiori o uguali ad a e valori di B minori di b
- il quarto ai record con valori di A maggiori o uguali ad a e valori di B maggiori o uguali a b

Intuitivamente, ogni nodo divide lo spazio in quattro parti e quindi una foglia di un quad tree contiene riferimenti corrispondenti ad un "quadratino" dello spazio bidimensionale, con valori di A compresi in un intervallo $[a_1, a_2)$ e quelli di B in un intervallo $[b_1, b_2)$.

Consideriamo ora una relazione di $N = 4.000.000$ di ennuple, con fattore di blocco $F = 10$, in cui i due attributi, insieme, formino una chiave e in cui vi siano circa $v = 2.000$ valori diversi per ciascun attributo (abbastanza densi, ad esempio valori numerici compresi fra 1 e 2.100 e distibuiti omogeneamente) e che ciascun valore di A sia associato a circa 2.000 valori diversi di B e viceversa, con una varianza abbastanza piccola. Supponiamo che le foglie contengano circa 145 riferimenti ciascuna (sia nel caso del quad tree sia in quello del B+-tree) e di avere un sistema che permetta di caricare nel buffer tutti i livelli intermedi (ma non le foglie) sia per i quad tree sia per i B+-tree. Calcolare in tale contesto il costo delle seguenti operazioni nel caso in cui sulla relazione (disordinata) è definito un quad tree e in quello in cui è definito un B+-tree su A, B:

1. conteggio delle ennuple con un dato valore di B
2. accesso alle ennuple con un dato valore di A
3. conteggio delle ennuple con un dato valore di A
4. conteggio delle ennuple con valore di A compreso fra 100 e 115 (si tratta di circa 30.000 ennuple)
5. conteggio delle ennuple con valore di B compreso fra 100 e 115 (si tratta di circa 30.000 ennuple)

Rispondere (sul foglio protocollo) riempiendo una tabella come la seguente, con qualche commento che spieghi la risposta:

	1	2	3	4	5
quad tree su A, B					
B+-tree su A, B					

Possibile soluzione
Tutto dipende dall'organizzazione ordinata su A nel B+-tree vs "quadratino" nel quad tree. Data l'ipotesi sui buffer, è sufficiente contare le foglie coinvolte in ciascuna operazione (per semplicità assumiamo un riempimento pieno delle foglie, per entrambi i tipi di albero)

	conteggio puntuale su B	accesso puntuale su A	conteggio puntuale su B	conteggio in intervallo su A	conteggio in intervallo su B
quad tree su A, B	ca. 2000/12=165	ca. 165+2.000	ca. 2000/12=165	ca. 30.000/12	ca. 30.000/12
B+-tree su A, B	ca. 2000	ca. 13+2000	ca. 2000/145=13	ca. 30.000/145	ca. 4.000.000/145