Basi di dati II
Esercizi di autovalutazione — 30 aprile 2013
Possibili soluzioni

Domanda 1 Il *semijoin* è un'operazione simile al join, in cui del secondo operando interessano solo gli attributi di join. In concreto, se il join su $A_1 = A_2$ di $r_1(X_1)$ e $r_2(X_2)$ (con $A_1 \in X_1$, $A_2 \in X_2$ e $X_1 \cap X_2 = \emptyset$) è definito come

$$r_1 \text{ JOIN}_{A_1=A_2} r_2 = \{ t \text{ su } X_1X_2 \mid \text{esistono } t_1 \in r_1 \text{ e } t_2 \in r_2 \text{ con } t[X_1] = t_1,\ t[X_2] = t_2 \text{ e } t_1[A_1] = t_2[A_2] \}$$

il semijoin corrispondente è definito come

$$r_1 \text{ SEMIJOIN}_{A_1=A_2} r_2 = \{ t_1 \mid t_1 \in r_1 \text{ ed esiste } t_2 \in r_2 \text{ con } t_1[A_1] = t_2[A_2] \}$$

Ad esempio:

R_1

A	B	C	D
1	1	1	1
2	2	2	1
3	2	2	2

R_2

G	E
1	1
3	2

$R_1 \text{ SEMIJOIN}_{D=G} R_2$

A	B	C	D
1	1	1	1
2	2	2	1

Mostrare algoritmi per l'esecuzione del semijoin, come modifica degli algoritmi noti per il join, indicandone il costo (e sottolineando i casi in cui tale costo è diverso da quello del join).
Inoltre, specificare quali possono essere i vantaggi dell'uso del semijoin in contesti distribuiti (con le due relazioni memorizzate in nodi diversi), anche come passo preliminare per l'esecuzione di join.

Risposta (cenni)
Gli algoritmi possono fermarsi alla verifica della presenza della ennupla del secondo argomento, senza accedere alle relative ennuple. Ciò diventa significativo quando si utilizza un indice (ad esempio in un nested loop con R_2 interna e indice su A_2 oppure in un merge scan che utilizzi un indice A_2 su R_2).
Con riferimento all'utilizzo dei buffer, il beneficio può risultare ancora maggiore, perché si possono costruire strutture temporanee che contengono solo il campo di join di R_2 e quindi sono molto più compatte, al punto che il join può essere svolto anche con una sola scansione.
In un contesto distribuito, il semijoin può essere utile per evitare di trasferire un'intera relazione; in pratica si trasferiscono solo i valori degli attributi di join, realizzando quindi un semijoin, per poi trasferire solo le ennuple che certamente contribuiscono al join.

Domanda 2 Spiegare perché, scrivendo un programma che accede a due basi di dati diverse utilizzando JDBC, non è possibile garantire il commit a due fasi. Indicare quali funzionalità aggiuntive sono necessarie per i server locali che vogliano realizzare tale servizio e come si può procedere con un sistema che non disponga di tali funzionalità.

Risposta (cenni)
Con JDBC non è possibile implementare sui partecipanti lo stato di ready, che affidi la responsabilità della decisione ad un coordinatore. L'unica soluzione alternativa prevede la disponibilità di "transazioni compensative" che annullino l'effetto delle transazioni effettuate sui singoli nodi.

Domanda 3 Si consideri il protocollo di commit a a due fasi (2PC).

1. Spiegare perché
   (a) un guasto del coordinatore (TM) può avere conseguenze molto pesanti anche sulle prestazioni dei partecipanti (RM);
   (b) un guasto di un partecipante non ha conseguenze particolari sulle prestazioni degli altri partecipanti (a parte l'eventuale abort di transazioni).

Per ovviare alle conseguenze negative di un guasto del coordinatore, alcune implementazioni del 2PC prevedono la possibilità di comunicazione fra i partecipanti (mentre la versione base prevede solo comunicazione fra il coordinatore e ciascuno dei partecipanti). In particolare, un partecipante che abbia una transazione in stato di "ready" può chiedere agli altri partecipanti informazioni sullo stato di tale transazione (che può essere "prima-del-ready", "ready", "commit" o "abort") presso di loro.

2. In tale contesto indicare
   (a) quali insiemi di risposte il partecipante può ricevere e quali invece no;
   (b) come (e in quali casi) il partecipante può trarre profitto dalle risposte ottenute.

Risposta (cenni)

1.
   (a) perché il partecipante rimane in attesa di decisione (con le risorse bloccate)
   (b) perché nessuno è costretto a rimanre in attesa

2.
   (a) se c'è almeno un commit, non ci possono essere abort né "prima-del-ready"; se c'è almeno un abort o un "prima-del-ready", non ci possono essere commit
   (b) con un abort o un commit si può prendere una decisione

Domanda 4 Si supponga che Napoleone abbia fatto utilizzare il commit a due fasi per organizzare le attività in battaglia. Si consideri il seguente scenario. Esistono:

(a) generali che possono coordinare azioni
(b) reparti di riserva disponibili per azioni
(c) messaggeri utilizzati dai generali e dai reparti di riserva per comunicare gli uni con gli altri

In particolare, il 2PC viene utilizzato dai generali per organizzare azioni che coinvolgano (*contemporaneamente, ad un certo orario*) due o più reparti di riserva (ma ciascun reparto potrebbe ricevere richieste da più generali). Ad esempio, il generale Murat, alle ore 10, decide di voler organizzare un attacco alle ore 12 con il coinvolgimento del quarto squadrone di cavalleria e della prima batteria di artiglieria pesante.
Descrivere brevemente il protocollo in questo contesto, sottolineando le criticità, dovute al fatto che si tratta di una battaglia e quindi tutti i soggetti coinvolti (generali, comandanti dei reparti di riserva e messaggeri) possono essere colpiti. In particolare, individuare quali ipotesi sarebbero necessarie (anche se non sempre soddisfatte in una battaglia) per permettere l'utilizzo del protocollo.

Risposta (cenni)

- protocollo: un generale invia richieste a due o più reparti, se tutti rispondono affermativamente (perché hanno ricevuto il messaggio, sono liberi e riescono a far arrivare il messaggio di risposta) il generale comunica l'effettivo svolgimento dell'attacco, chiedendo conferma e continua a richiedere conferma finché non la riceve; ciascun reparto, se dichiara la propria disponibilità, non può poi ritirarla
- criticità:
  - ogni reparto deve avere la possibilità di ricordare messaggi inviati e ricevuti e quindi (ipotesi fondamentale) deve esistere un "registro" dei messaggi che possa essere consultato da un comandante che subentri qualora quello in carica venga colpito; l'ipotesi fondamentale è che il registro rimanga sempre integro (cosa ovviamente non scontata nel contesto in questione);
  - il requisito del rispetto dei tempi (che nel 2PC non c'è) non può essere garantito: la ripetizione dei messaggi nella seconda fase potrebbe non essere sufficiente (i reparti potrebbero ricevere la conferma in ritardo)

Domanda 5 Considerare un sistema distribuito su cui viene eseguita una transazione che coinvolge tre nodi, un coordinatore A e due partecipanti B e C. Dopo la richiesta di `prepare` da parte del coordinatore, il partecipante C va in crash senza ricevere il messaggio, mentre il partecipante B fa in tempo a ricevere il messaggio e a rispondere positivamente ma va in crash poco dopo, non ricevendo la prima notifica della decisione. Indicare, nello schema sottostante, una possibile sequenza di scritture sui log e invio di messaggi, supponendo che entrambi i nodi siano ripristinati abbastanza presto. Per semplicità, si fa riferimento ad una sola transazione e quindi non c'è bisogno di indicarla. Per i messaggi si usi la notazione *tipo*→*destinatari* (come nell'esempio: `prepare`→`B,C`). Supporre che timeout per le varie fasi scattino all'incirca negli istanti indicati a sinistra della tabella.

	Nodo A		Nodo B		Nodo C	
	Log	Messaggi	Log	Messaggi	Log	Messaggi
	`prepare(B,C)`	`prepare`→`B,C`				
			`ready`			
				`ready`→`A`		
					crash	
			crash			
t_1	`abort`					
		`abort`→`B,C`				
t_2		`abort`→`B,C`				
			restart			
			`abort`			
				`ack`→`A`		
t_3		`abort`→`C`				
					restart	
					`abort`	
						`ack`→`A`
	`complete`					

Eventuali commenti:
A deve ricevere un `ack` anche da C, prima di concludere con `complete`, perché C potrebbe essere in stato di `ready`