Basi di dati II, primo modulo
Esercizi di autovalutazione — 11 marzo 2011

Domanda 1 Si consideri una relazione *R(CodiceCliente,Cognome,Nome,Categoria)* con $N = 1.000.000$ ennuple. Con riferimento alla ricerca di tutti i clienti di una certa categoria, indicare il costo dell'accesso sequenziale e di quello diretto con indice su *Categoria* nei due casi seguenti (mostrare formule e valori numerici):

1. campo selettivo ($v_1 = 100.000$ valori diversi per *Categoria*)
2. campo poco selettivo ($v_2 = 20$ valori diversi per *Categoria*)

Supporre che l'indice abbia profondità $p = 4$ e che i fattori di blocco del file e dell'indice siano rispettivamente $f_R = 50$ e $f_C = 200$.

Domanda 2 Si consideri un B-tree con nodi intermedi che contengono due chiavi e tre puntatori e foglie con due chiavi, in cui vengano inserite chiavi (a partire dall'albero vuoto) nel seguente ordine: 12, 22, 32, 42, 52, 13, 14, 15, 16, 17, 18. Mostrare l'albero dopo l'inserimento di tre, cinque, sette chiavi e alla fine.

Domanda 3 Svolgere sul DBMS didattico SimpleDB (`http://www.cs.bc.edu/ sciore/simpledb/`) gli esercizi proposti in aula:

- Modificare la classe FileMgr in modo che supporti statistiche sul numero di blocchi letti e/o scritti (aggiungere uno o più metodi allo scopo)
- Modificare i metodi commit e rollback della classe RemoteConnectionImplementation (in simpledb.remote) in modo che stampino queste statistiche; debbono accedere al FileMgr che può essere ottenuto con il metodo SimpleDB.fileMgr (del package simpledb.server)
- Modificare BufferMgr per contare le richieste di pin()
- Modificare RecordMgr per contare i record che vengono letti

Domanda 4 Utilizzando SimpleDB (o meglio, i suoi moduli di livello più basso), scrivere una classe di test che esegue alcune operazioni su un file. Avvertenze:

- porre la classe nel package `simpledb.record`
- la classe deve disporre di alcuni degli oggetti del server; allo scopo, includere una chiamata al metodo init di SimpleDB:
  `SimpleDB.init("studentdb");`
- RecordFile fa riferimento ad una transazione, che deve essere creata allo scopo:
  `Transaction tx = new Transaction();`
- RecordFile fa riferimento allo schema logico e a quello fisico della tabella che viene implementata; vanno entrambi creati nella classe di test:
  `Schema sch = new Schema();`
  `sch.addIntField( ..... );` e/o `sch.addStringField( ..... );` (due o tre campi, non di più)
  ...
  `TableInfo ti = new TableInfo("prova",sch);`

Nella classe di test, effettuare

1. l'inserimento di una sequenza di 10000 record (con valori generati casualmente)
2. la stampa di tutti i record
3. l'eliminazione di una parte dei record (orientativamente il 50% dei record; ad esempio, quelli per cui il valore di un campo soddisfa una certa condizione)
4. la scansione di tutti i record (con la lettura di un campo numerico e un qualche calcolo, ad esempio il valore medio)
5. l'inserimento di altri 7000 record (sempre con valori generati casualmente)

Per ciascuna delle operazioni, stampare il numero di record letti e scritti, il numero di richieste di pin ricevute dal buffer e il numero di accessi a memoria secondaria.

Domanda 5 Come illustrato brevemente a lezione, esistono varie strategie utilizzabili da parte del gestore del buffer per scegliere la pagina da utilizzare (e quindi il blocco da rimpiazzare) per eseguire una *pin* (o *fix*):

- *naif*: se ne sceglie una qualunque
- *FIFO*: si sceglie quella caricata da più tempo
- *LRU (least recently used)*: si sceglie quella utilizzata meno di recente
- *clock*: si sceglie la prima libera successiva a quella utilizzata in occasione dell'ultimo rimpiazzo; questa strategia considera il buffer come circolare (la prima pagina è successiva all'ultima) e da ciò deriva il nome.

Il gestore del buffer di SimpleDB utilizza la strategia naif (il metodo findExistingBuffer nella classe BasicBufferMgr itera sul buffer partendo sempre dall'inizio). Implementare le altre tre strategie, modificando il metodo findExistingBuffer.