Matricola: _____

Cognome:_____

Nome:_____

Basi di Dati
Prova intermedia del giugno 2015

Avvertenze: è severamente vietato consultare libri e appunti. Durata 2h

DOMANDE TEORIA (è obbligatorio rispondere ad almeno una domanda delle quattro di seguito elencate)

a) (3) Illustrare l'architettura di un DMBS descrivendo in particolare il modulo di gestione dei buffer; si indichi inoltre, per ogni modulo dell'architettura, quali sono le proprietà delle transazioni che contribuisce a garantire.

b) (2) Si presenti in dettaglio la definizione di Conflict-Serializzabilità (CSR).

c) (2) Lo studente illustri la struttura di accesso ai dati denominata indice primario denso: caratteristiche della struttura, ricerca, inserimento e cancellazione di entry dall'indice.

~~d) (3) Lo studente illustri l'algoritmo di codifica di Huffmann e mostri un esempio di codifica di un messaggio di lunghezza 4 sull'alfabeto {Z,W,Y,X}.~~

d) (3) Domanda aggiuntiva di teoria (Transazioni, DBMS, Affidabilità, Concorrenza, Indici, Ottimizzazione, XML)

ESERCIZI

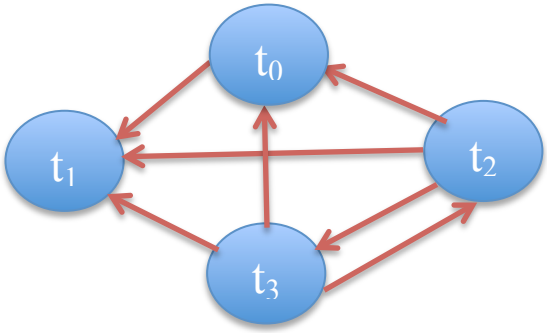
Esecuzione concorrente

Dato il seguente schedule S:

- (2) indicare se è conflict-SR oppure no (calcolare l'insieme dei conflitti)

  S: $r_0(t), r_2(z), r_3(z), w_1(z), r_3(x), r_2(x), w_3(x), w_3(y), w_2(y), w_0(y), w_1(t)$

  conflitti(S) = { $(r_0(t), w_1(t)), (r_2(z), w_1(z)), (r_3(z), w_1(z)), (r_2(x), w_3(x)), (w_3(y), w_2(y)),$ $(w_3(y), w_0(y)), (w_2(y), w_0(y))$ }



  Poiché il grafo non è ACICLICO lo schedule non è CSR.

- (3) se non è CSR verificare se è view-SR oppure non-SR (giustificare dettagliatamente la risposta).

  S: $r_0(t), r_2(z), r_3(z), w_1(z), r_3(x), r_2(x), w_3(x), w_3(y), w_2(y), w_0(y), w_1(t)$

  $t_0 = r_0(t), w_0(y)$

  $t_1 = w_1(z), w_1(t)$

  $t_2 = r_2(z), r_2(x), w_2(y)$

  $t_3 = r_3(z), r_3(x), w_3(x), w_3(y)$

  LeggeDa(S) = Ø $\Rightarrow$ $t_0 < t_1$, $t_2 < t_1$, $t_3 < t_1$, $t_2 < t_3$

  ScrittureFinali(S) = { $w_0(y), w_1(t), w_3(x), w_1(z)$ *} $\Rightarrow$ t2 < t0 , t3 < t0*

  Schedule seriali da considerare:

  S1: $r_2(z), r_2(x), w_2(y), r_3(z), r_3(x), w_3(x), w_3(y), r_0(t), w_0(y), w_1(z), w_1(t)$

  LeggeDa(S1) = Ø

  ScrittureFinali(S1) = { $w_0(y), w_1(z), w_1(t), w_3(x)$ *} quindi S è view-equivalente a S1 quindi è VSR.*

Ottimizzazione

Si consideri il seguente schema relazionale contenente le ricette di una catena di ristoranti:

INGREDIENTE(Codice, Nome, Calorie);
COMPOSIZIONE(Ricetta, Ingrediente, Quantità)
RICETTA(CodiceRicetta, Nome, Regione, TempoPreparazione)
Nota: la quantità nella tabella COMPOSIZIONE è espressa in grammi
Vincoli di integrità: COMPOSIZIONE.Ricetta → RICETTA, COMPOSIZIONE.Ingrediente → INGREDIENTE

Formulare in SQL la seguente interrogazione:
(1) Trovare gli ingredienti usati in ricette della Regione Veneto, riportando, il codice della ricetta e il nome e le calorie dell'ingrediente.

SELECT R.CodiceRicetta, I.Nome, I.Calorie

FROM RICETTA R JOIN COMPOSIZIONE C ON R.CodiceRicetta = C.Ricetta

JOIN INGREDIENTE I ON C.Ingrediente = I.Codice

WHERE R.Regione = 'Veneto'

(4) Calcolare il costo dell'interrogazione in termini di numero di accessi a memoria secondaria sotto le seguenti ipotesi:

- la selezione su ricetta richiede una scansione sequenziale della tabella RICETTA
- l'ordine di esecuzione del join è RICETTA ⋈ COMPOSIZIONE ⋈ INGREDIENTE
- le operazioni di join vengono eseguite con la tecnica "*Nested Loop Join*" con una pagina di buffer disponibile per ogni tabella
- NP(INGREDIENTE) = 40, NP(COMPOSIZIONE) = 200, NP(RICETTA) = 12
- NR(INGREDIENTE) = 1200, NR(COMPOSIZIONE) = 11000, NR(RICETTA) = 260
- VAL(Regione, RICETTA) = 20

Applicando la formula per il costo della tecnica "Nested Loop Join" al caso specifico e secondo l'ordine di join indicato risulta:

COSTO = NP(RICETTA) +

NR(RICETTA con selezione Regione='Veneto') * NP(COMPOSIZIONE)

*** NP(INGREDIENTE)**

COSTO = 12 +

260/20 * 200 * 40 = 104012

(2) Come cambia il costo se è disponibile un indice B+-tree sull'attributo Codice della tabella INGREDIENTE che occupa 2 pagine di memoria secondaria.

COSTO = NP(RICETTA) +

NR(RICETTA con selezione Regione='Veneto') * NP(COMPOSIZIONE)

*** (ProfIndice + selettività di Codice)**

COSTO = 12 +

260/20 * 200 * 3 = 7812

~~**SQL (è obbligatorio svolgere almeno due esercizi di questa sezione)**~~

~~Si consideri il seguente schema relazionale contenente i dati concernenti le ricette di un ristorante:~~

~~**INGREDIENTE**(Codice, Nome, Calorie, Grassi, Proteine, Carboidrati);~~
~~**COMPOSIZIONE**(Ricetta, Ingrediente, Quantità)~~
~~**RICETTA**(CodiceRicetta, Nome, Regione, TempoPreparazione)~~
~~Nota: la quantità di grassi, proteine e carboidrati e in grammi su 100 grammi di ingrediente; la quantità nella tabella COMPOSIZIONE è espressa in grammi~~
~~Vincoli di integrità: COMPOSIZIONE.Ricetta → RICETTA, COMPOSIZIONE.Ingrediente → INGREDIENTE~~

~~Formulare in SQL le seguenti interrogazioni (definire viste solo dove è necessario):~~

~~(3) SQL.1 Trovare il nome e il tempo di preparazione delle ricette della Regione Liguria che contengono almeno un ingrediente con più del 30% di grassi e almeno un ingrediente con meno del 10% di proteine.~~

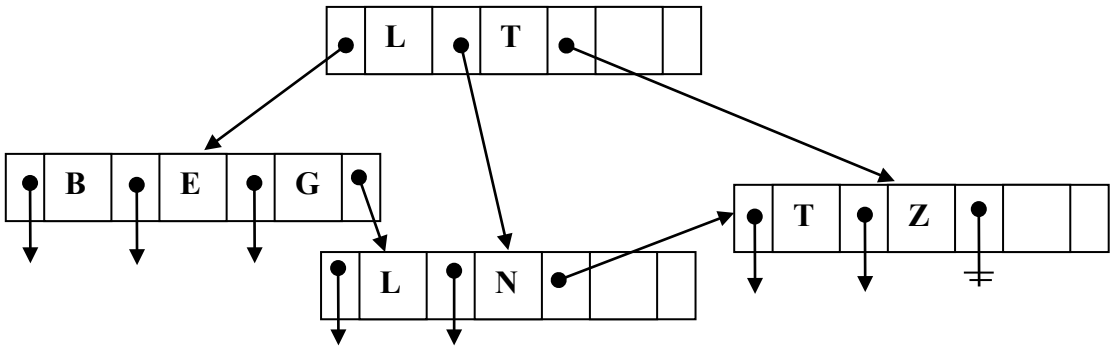
~~(3) SQL.2 Trovare per ogni ricetta la quantità totale (in grammi) di proteine e carboidrati, riportando oltre alle quantità richieste anche il nome della ricetta.~~

~~(3) SQL.3 Trovare gli ingredienti usati solo in ricette della Regione Veneto, riportando il nome e le calorie dell'ingrediente.~~

~~(2) SQL.4 Trovare per ogni ricetta con tempo di preparazione maggiore di 60 minuti l'elenco degli ingredienti ordinati in senso crescente per nome della ricetta e decrescente per quantità; nel risultato riportare il nome della ricetta e tutti gli attributi della tabella ingrediente.~~

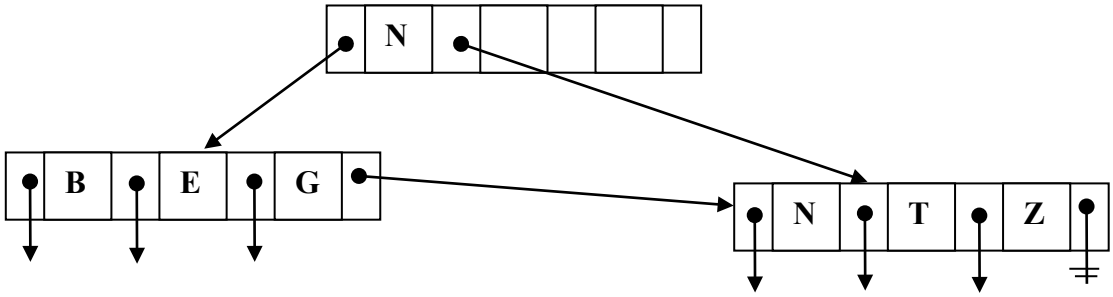
B$^+$-tree

Dato il seguente B$^+$-tree (fan-out=4), mostrare lo stato dell'albero:



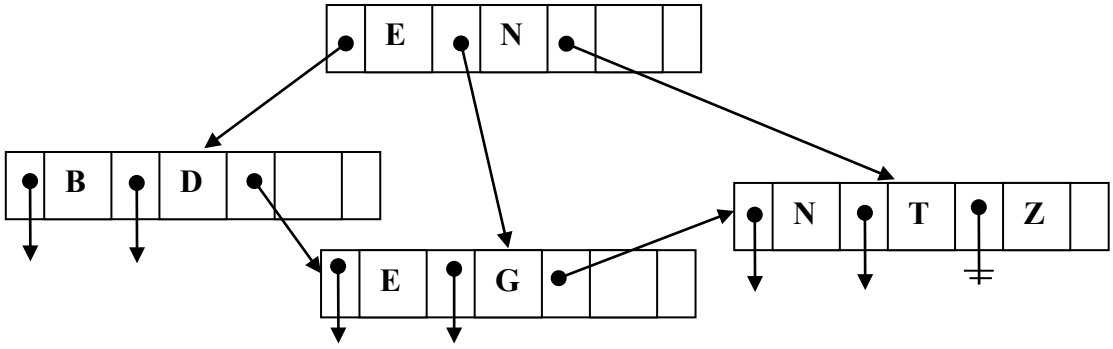
- (2) dopo la cancellazione del valore chiave L e

  Poiché il limite minimo di riempimento dei nodi foglia è pari a 2 valori della chiave, la cancellazione di L pone il secondo nodo foglia sotto il limite di riempimento minimo pertanto va eseguito un merge. Possiamo scegliere come nodo fratello a cui applicare il merge il nodo di destra (quello che contiene i valori T e Z). Applicando il merge si ottengono alla fine due nodi foglia e si elimina di conseguenza un puntatore dalla radice. La struttura dell'albero dopo la cancellazione risulta quindi la seguente:



- (3) dopo l'inserimento del valore chiave D supponendo di partire dall'albero ottenuto al punto a).

  Poiché il limite massimo di riempimento dei nodi foglia è pari a 3 valori della chiave, l'inserimento di D pone il primo nodo foglia sopra il limite di riempimento massimo pertanto va eseguito uno split. Dopo lo split va aggiunto un puntatore alla radice che contiene quindi alla fine tre puntatori (ancora entro il limite massimo). La struttura dell'albero dopo l'inserimento risulta quindi la seguente:



XML

(6) Dato il seguente file XML e i seguenti requisiti si produca il file XML schema (XSD) che ne descrive la sintassi.

```
<?xml version="1.0"?>
<agenzia  xmlns ="http://www.agenzia.org"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.agenzia.org
                                        agenzia.xsd">
   <edificio id=”E002” >
         <proprietario>
               <nome>Mario</nome>
               <cognome>Rossi</cognome>
         </proprietario>
         <tipo>A3</tipo>
         <prezzoOfferto timestamp=” 2014-11-30 T 11:45:22”>100000</PrezzoOfferto>
         <prezzoOfferto timestamp=” 2015-01-20 T 11:45:22”>110000</PrezzoOfferto>
   </edificio>
   <edificio id=”E0023” >
         <proprietario>
               <nome>Mario</nome>
               <cognome>Rossi</cognome>
         </proprietario>
         <proprietario>
               <nome>Giovanni</nome>
               <cognome>Bianchi</cognome>
         </proprietario>
         <tipo>A1</tipo>
         <descrizione>Descrizione edificio</descrizione>
   </edificio>
…
</agenzia>
```

Requisiti

L’attributo `id` e l’attributo `timestamp` sono obbligatori. L’elemento `tipo` può assumere solo i valori: A1, A2, A3.

XMLSchema agenzia.xsd da completare

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace=" http://www.agenzia.org"
            xmlns="http://www.agenzia.org">
<xsd:complexType name=”T_Proprietario”>
   <xsd:sequence>
         <xsd:element name="nome" type="xsd:string"/>
         <xsd:element name="cognome" type="xsd:string"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name=”T_TipoEdificio”>
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="A1"/>
      <xsd:enumeration value="A2"/>
      <xsd:enumeration value="A3"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name=”T_PrezzoOfferto”>
   <xsd:simpleContent>
      <xsd:extension base="xsd:unsignedInt">
         <xsd:attribute name="timestamp" type="xsd:dateTime" use="required"/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>

<xsd:element name=”edificio”>
   <xsd:complexType>
     <xsd:sequence>
         <xsd:element name="proprietario" type="T_Proprietario"
                      maxOccurs=”unbounded”/>
         <xsd:element name="tipo" type="T_TipoEdificio"/>
         <xsd:element name="descrizione" type="xsd:string" minOccurs=”0”/>
         <xsd:element name="prezzoOfferto" type="T_PrezzoOfferto" minOccurs=”0”
                      maxOccurs=”unbounded”/>
     </xsd:sequence>
     <xsd:attribute name="id" type="xsd:ID" use="required"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name=”agenzia”>
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref=”edificio” maxOccurs=”unbounded”/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>

</xsd:schema>
```