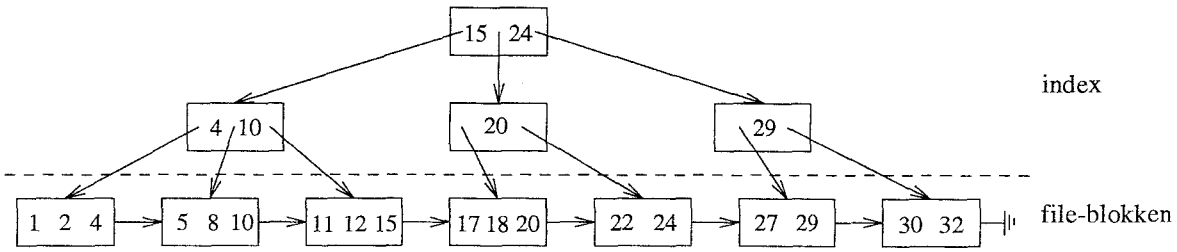
Tentamen Databases II (9 januari 2003)

1. Een B-tree van de orde n bevat p pointers per node waarbij $\lceil n/2 \rceil \leq p \leq n$ (behalve voor de root, daarvoor geldt: $p \leq n$).

   a) Laat in een plaatje zien hoe onderstaande B^+-tree (bestaande uit een B-tree van de orde 3 als index set en een file met maximaal 3 records per block als sequence set) eruit ziet, nadat een record met key-waarde 9 is toegevoegd.



   b) Stel dat we een **on**gesorteerde verzameling van 10 miljoen records hebben. Elk record heeft een primaire sleutel van 10 bytes en een totale grootte van 100 bytes. We willen deze gegevens gaan opslaan in een file en een non-dense index creëren op de primaire sleutel. We besluiten voor deze combinatie een B^+-tree te gebruiken, waarbij alle blokken een grootte hebben van 1K bytes.

      1) Beschrijf kort en bondig (dus in 1 of 2 zinnen) hoe de bedoelde B^+-tree op een efficiente (!) manier gemaakt kan worden.

      2) Wat valt er te zeggen over de (gemiddelde) vullingsgraad van de blokken van de op deze wijze gemaakte boom?

2. In een database zijn in een file alle gegevens opgeslagen van 100.000 studenten. Verder is gegeven dat:

   - de file bestaat uit een enkel-gelinkte lijst van blokken
   - de blokken van deze file zo vol mogelijk zitten
   - er alleen sprake is van unspanned records
   - de file ongeordend is (dus de records staan **niet** op volgorde van een of ander veld)
   - er precies één record is voor elke student
   - elk studenten-record 50 bytes beslaat en zo'n record behalve de velden stad en college ook nog velden zoals b.v. naam, collegekaartnummer en studiebeurs bevat.
   - er 2000 verschillende steden en 100 verschillende colleges voorkomen in de file en dat studenten uniform (i.e. zo gelijkmatig mogelijk) verdeeld zijn over steden en colleges
   - de blockgrootte 1K bytes is, een pointer 4 bytes kost, een plaatsnaam 16 bytes en een collegenaam 12 bytes in beslag neemt
   - er (alleen) twee indices bestaan op resp. de velden stad en college, waarbij deze beide indices geimplementeerd zijn m.b.v. accession-lists (met daarin fysieke recordpointers) en een B^+-tree (met **alleen** de root **continu** aanwezig in primair geheugen!) met een enkel-gelinkte sequence-set

   Bereken nu hoeveel disk-blocks er gemiddeld van schijf gelezen moeten worden (i.e., hoeveel disk-accessen nodig zijn) voor het beantwoorden van de volgende vier vragen:

   a) Heeft de student met collegekaartnummer 135246 een studiebeurs?

   b) Wat zijn de namen van alle studenten uit Haarlem?

   c) Wat zijn de namen van alle studenten die het college Netwerken volgen?

   d) Wat zijn de namen van alle studenten uit Amsterdam die het college Gegevensbanken volgen?

3. Bij deze opgave gaat onderdeel a over transacties en betreffen de onderdelen b en c de serializability van transacties.

   a) Het begrip transactie is nauw verbonden met 4 aspecten. Om deze 4 aspecten makkelijk te kunnen onthouden is er een ezelsbruggetje. Geef de ezelsbrug en de 4 termen die erbij horen.

   b) Hieronder wordt een schedule gegeven voor een aantal transacties. De subscript bij elke operatie geeft aan tot welke transactie die operatie behoort. De letters r en w staan voor read resp. write en de hoofdletters A t/m E representeren data-items. De tijdsvolgorde is de normale leesvolgorde (i.e. van links naar rechts en daarna, zo nodig, verder op de volgende regel).

      Bepaal m.b.v. het in het boek gegeven algorithme of dit schedule conflict serializable is.
      Geef niet alleen je uiteindelijke conclusie (wel of niet conflict serializable), maar ook een plaatje van het resultaat van het algorithme waar je conclusie op gebaseerd is. Laat ook d.m.v. labels in het plaatje zien welke data-items betrokken waren bij welke "conflicten".

      (Je hoeft dus niet de volledige uitvoering van het algorithme op te schrijven, maar alleen het bijpassende plaatje met daarin alle juiste labels. Vergeet niet de uit het plaatje te trekken conclusie te geven.)

      $r_2(C); r_2(B); w_2(B); r_3(B); r_3(C); r_1(A); w_1(A); w_3(B); w_3(C); w_2(A); r_1(B); w_1(B); w_2(A);$

   c) Doe nu voor onderstaand schedule hetzelfde als gevraagd bij onderdeel b.

      $r_1(A); r_3(B); r_2(C); r_4(B); r_4(C); r_2(E); w_2(E); r_2(D); r_1(E); w_2(D); w_4(C); r_4(D); w_4(D); r_1(C); r_3(E); r_3(D); w_3(B); r_1(D); w_1(D); r_1(B); w_1(A);$

4. Gegeven de volgende 5 schedules:

   $S_1 : r_1(X); w_1(X); w_2(X); c_2; r_3(X); w_3(X); c_3; c_1;$
   $S_2 : r_1(X); w_1(X); r_3(X); w_2(X); c_2; w_3(X); c_1; c_3;$
   $S_3 : r_1(X); w_1(X); c_1; w_2(X); c_2; r_3(X); w_3(X); c_3;$
   $S_4 : r_1(X); w_1(X); r_2(X); w_2(X); r_1(Y); w_1(Y); c_1; c_2;$
   $S_5 : r_1(X); w_1(X); r_2(X); w_2(X); c_2; r_1(Y); c_1;$

   Geef voor elk van de 5 schedules S_i (i=1 t/m 5) antwoord op achtereenvolgens de volgende 3 vragen:

   Is S_i recoverable? cascading rollback avoiding? strict?

   N.B. Geef bij elk van de bovenstaande 5 onderdelen je antwoord in de vorm van 3 ja of nee antwoorden. (Natuurlijk steeds in de volgorde van de 3 deelvragen.)

Puntenverdeling:

opgave 1: 20 punten
opgave 2: 30 punten
opgave 3: 25 punten
opgave 4: 25 punten

Cijfer := (aantal punten) / 10