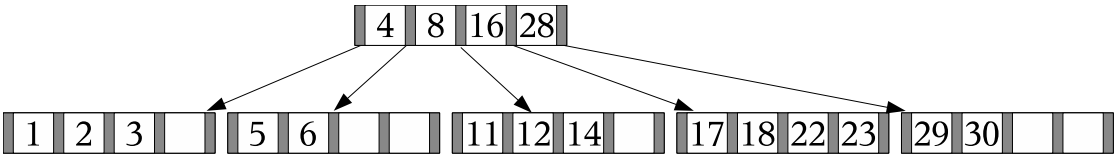
8. Übung zur Vorlesung "Datenbanken" im Sommersemester 2006
– mit Musterlösungen –

Prof. Dr. Gerd Stumme, Dr. Andreas Hotho, Dipl.-Inform. Christoph Schmitz

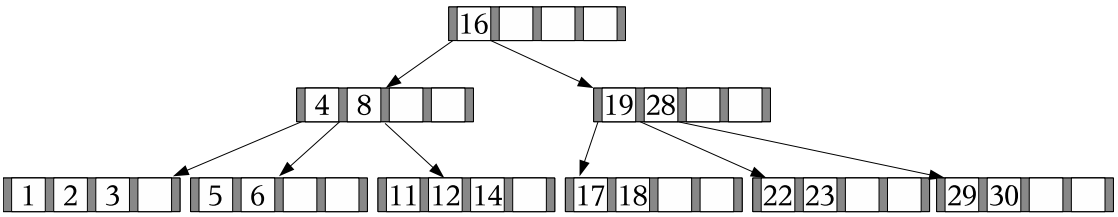
25. Juni 2006

Aufgabe 1 – Indexstrukturen

Zeichnen Sie diesen B-Baum, nachdem der Wert 19 eingefügt wurde.



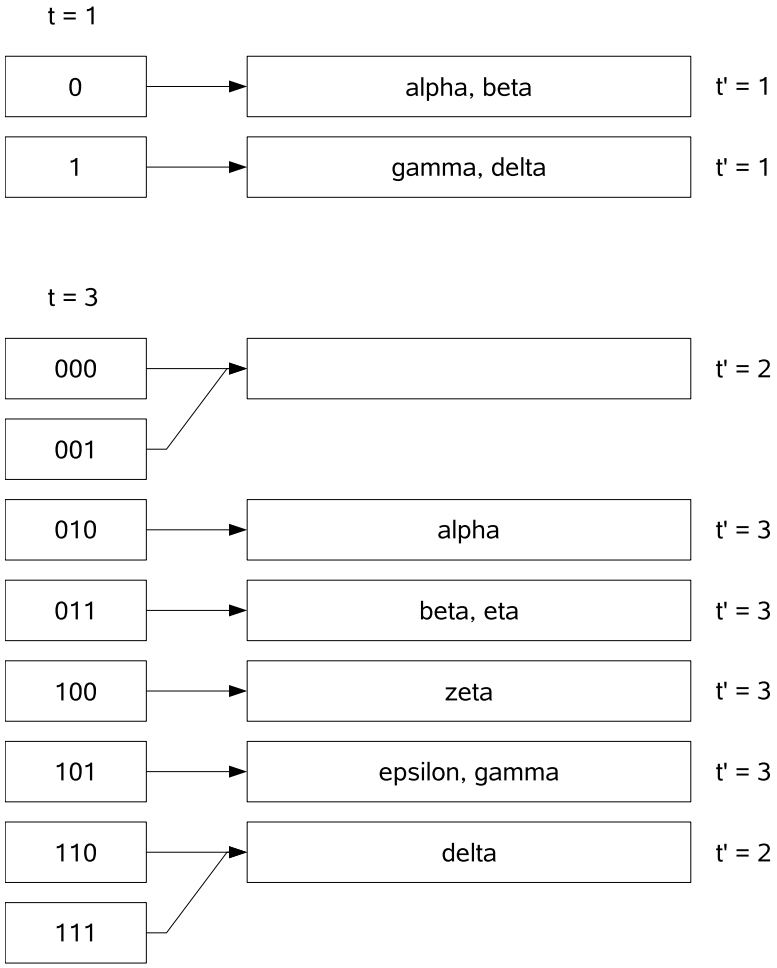
Einfügung von 19 verursacht zwei Überläufe und somit Einfügen einer neuen Ebene:



Aufgabe 2 – Indexstrukturen

Fügen Sie die folgenden Schlüssel in der angegebenen Reihenfolge in eine erweiterbare Hashtabelle mit Bucketgröße zwei ein. Zeichnen Sie den Zustand der Hashtabelle nach den Einfügungen von delta und eta.

x	$h(x)$
alpha	0101
beta	0110
gamma	1010
delta	1101
epsilon	1011
zeta	1001
eta	0111



Aufgabe 3 – RAID

Ein RAID-5-Array befindet sich in dem Zustand nach Tabelle 1. Dabei sei $B_{2,3}$ der dritte Block auf der zweiten Platte, $P_{124,3}$ der Parity-Block der dritten Blöcke von Platten 1, 2 und 4, usw.

Tabelle 1

Platte 1		Platte 2		Platte 3		Platte 4	
$P_{234,1}$	1111	$B_{2,1}$	0100	$B_{3,1}$	1101	$B_{4,1}$	0110
$B_{1,2}$	0110	$P_{134,2}$	0111	$B_{3,2}$	0110	$B_{4,2}$	0111
$B_{1,3}$	1011	$B_{2,3}$	1100	$P_{124,3}$	1110	$B_{4,3}$	1001
$B_{1,4}$	0010	$B_{2,4}$	1001	$B_{3,4}$	0110	$P_{123,4}$	1101

- Schreiben Sie die Werte

1000 in $B_{1,2}$
0100 in $B_{3,4}$

Wie sieht das RAID-Array nachher aus? Auf welche Platten wurde zugegriffen?

Platte 1		Platte 2		Platte 3		Platte 4	
$P_{234,1}$	1111	$B_{2,1}$	0100	$B_{3,1}$	1101	$B_{4,1}$	0110
$B_{1,2}$	**1000**	$P_{134,2}$	**1001**	$B_{3,2}$	0110	$B_{4,2}$	0111
$B_{1,3}$	1011	$B_{2,3}$	1100	$P_{124,3}$	1110	$B_{4,3}$	1001
$B_{1,4}$	0010	$B_{2,4}$	1001	$B_{3,4}$	**0100**	$P_{123,4}$	**1111**

Zugriffe:

- für $B_{1,2}$: Platte 1 und Platte 2 für die Parity-Information
- für $B_{3,4}$: Platte 3 und Platte 4 für die Parity-Information

- In dem RAID 5 ist Platte 3 ausgefallen. Der Zustand nach Einbau der neuen Platte ist:

Tabelle 2

Platte 1		Platte 2		Platte 3		Platte 4	
$P_{234,1}$	1111	$B_{2,1}$	0100	$B_{3,1}$	xxxx	$B_{4,1}$	0110
$B_{1,2}$	0110	$P_{134,2}$	1000	$B_{3,2}$	xxxx	$B_{4,2}$	0111
$B_{1,3}$	1011	$B_{2,3}$	1100	$P_{124,3}$	xxxx	$B_{4,3}$	1001
$B_{1,4}$	0010	$B_{2,4}$	1001	$B_{3,4}$	xxxx	$P_{123,4}$	1101

  - Rekonstruieren Sie den Inhalt von Platte 3.
  - Welche Operation hat zwischen dem Zustand von Tabelle 1 und dem von Tabelle 2 stattgefunden?

  Nur der Parityblock $P_{134,2}$ hat sich geändert, also gab es nur eine Änderung in Block $B_{3,2}$. Das Datum war

  $B_{3,2} = B_{1,2} \oplus P_{134,2} \oplus B_{4,2} = 0110 \oplus 1000 \oplus 0111 = 1001$

  Der Rest von Platte 3 bleibt wie in Tabelle 1.

Aufgabe 4 – Indexstrukturen

In der Vorlesung haben Sie als Indexstrukturen B-Bäume und Hashes kennengelernt.

- Nennen Sie die wesentlichen Vorteile der beiden Indexstrukturen gegenüber der jeweils anderen!

  B-Baum

– Sortierung der Daten wird erhalten: Sort-Merge-Join und Bereichsanfragen möglich
– keine Hashfunktion notwendig, nicht so abhängig von Schlüsselverteilung

Hash

– Zugriff in $O(1)$ statt $O(\log n)$, es sind genau 2 Zugriffe notwendig

- Welcher Index (Art, welche Attribute) würde die folgenden Anfragen jeweils am meisten beschleunigen?

  1. `SELECT * FROM Professor WHERE 50000 <= gehalt <= 60000`
     B-Baum, da dies eine Bereichsanfrage ist, die in einer Hashtabelle nicht effizient beantwortet werden kann.
  2. `SELECT * FROM Professor WHERE Standort = 'WA73' and Raum = '0441'`
     Ein Hash-Index auf (Standort, Raum) würde hier einen Zugriff in $O(1)$ erlauben.

Hausaufgabe: Normalisierung

Bringen Sie die folgende Relation in Bezug auf die gegebenen funktionalen Abhängigkeiten mit dem Dekompositionsalgorithmus in BCNF.

R1(A, B, C, D, E, F, G, H, I, J, K, L)

Funktionale Abhängigkeiten:

(i) A → B, C, D, E, F, G

(ii) B → C, D, E, F, G

(iii) A, H → I, J, K, L

(iv) I → J, L

(v) F → E

Bemerkung Im Skript sind die Relationen jeweils als R_i^1, R_i^2 usw. nummeriert. Hier wird der Übersicht halber einfach eine fortlaufende Nummerierung verwendet.

- Zerlege R1 nach (i) →
  R2(A, H, I, J, K, L)
  R3(A, B, C, D, E, F, G)

- Zerlege R3 nach (ii) →
  R2(A, H, I, J, K, L)
  R4(A, B)
  R5(B, C, D, E, F, G)
- Zerlege R2 nach (iv) →
  R6(A, H, I, K)
  R7(I, J, L)
  R4(A, B)
  R5(B, C, D, E, F, G)
- Zerlege R5 nach (v) →
  R6(A, H, I, K)
  R7(I, J, L)
  R4(A, B)
  R8(B, C, D, F, G)
  R9(F, E)

Bemerkung Die Relation R1 hieß ursprünglich

Bestellung(RechnungNr, KundenNr, Name, Adresse, Bankname, BLZ, KontoNr, PositionNr, ArtikelNr, ArtBez, Menge, Einzelpreis)

Die normalisierten Relationen sind also dementsprechend:

RechnungPositionen(RechnungNr, PositionNr, ArtikelNr, Einzelpreis)

Artikel(ArtikelNr, ArtBez, Einzelpreis)

RechnungKunde(RechnungNr, KundenNr)

Kunden(KundenNr, Name, Adresse, BLZ, KontoNr)

Banken(BLZ, Bankname)

Solch eine Relation (bzw. die normalisierten Relationen) könnte z. B. die Bestellungen eines Versandhändlers aufnehmen.

Hausaufgabe: SQL

1. Berechnen Sie die Gesamtzahl der Semesterwochenstunden, die die einzelnen Professoren erbringen. Dabei sollen auch die Professoren berücksichtigt werden, die keine Vorlesungen halten.

```
select persnr, name, sum(sws) as swssumme
from professoren left outer join vorlesungen
     on persnr=gelesenvon
group by persnr, name
```

2. Finden Sie die Namen der Studenten, die in keiner Prüfung eine bessere Note als 3.0 hatten.

   a) Berücksichtigen Sie dabei alle Studenten, auch die ohne Prüfung.

```
select name
from studenten
where matrnr not in (
  select matrnr
  from pruefen
  where note < 3.0
)
```

   b) Berücksichtigen Sie nur die Studenten, die schon eine Prüfung abgelegt haben.

```
select name
from studenten s join pruefen p on s.matrnr = p.matrnr
where s.matrnr not in (
  select matrnr
  from uni.pruefen
  where note < 3.0
)
```

Auflösung zur SQL-Aufgabe vom 6. Übungsblatt (11.6.2007)

```
create view notendurchschnitt(matrnr, note) as (
   select matrnr, avg(note)
   from pruefen
   group by matrnr)
```