Kapitel 8: Physischer Datenbankentwurf

- Speicherung und Verwaltung der Relationen einer relationalen Datenbank so, dass eine möglichst große Effizienz der einzelnen Anwendungen auf der Datenbank ermöglicht wird.
- Vor dem Hintergrund eines logischen Schemas und den Anforderungen der geplanten Anwendungen wird ein geeignetes *physisches Schema* definiert.

8.1 Grundlagen

Begriffe

- physische Datenbank: Menge von *Dateien*,
- Datei: Menge von gleichartigen *Sätzen*,
- Satz: In ein oder mehrere *Felder* unterteilter Speicherbereich.
- relationale Datenbanken: Relationen, Tupel und Attribute.

Indexstrukturen

Hilfsstrukturen zur Gewährleistung eines effizienten Zugriffs zu den Daten einer physischen Datenbank.

Zugriffseinheiten

- *Seiten*, oder auch *Blöcke*,
- ein Block besteht aus einer Reihe aufeinander folgender Seiten,
- eine Seite enthält in der Regel mehrere Tupel.
- Typischerweise hat eine Seite eine Größe von 4KB bis 8KB und ein Block eine Größe von bis zu 32KB.

Speicherhierarchie

- *Primärspeicher* (*interner Speicher*) bestehend aus *Cache* und einem in Seitenrahmen unterteilten *Hauptspeicher*. Direkter Zugriff im Nanosekundenbereich.
- *Sekundärspeicher* (*externer Speicher*), typischerweise in Seiten unterteilter externer Plattenpeicher. Direkter Zugriff im Millisekundenbereich.
- *Tertiärspeicher* (*Archiv*): kein direkter Zugriff möglich.

Pufferverwaltung

- Anzahl benötigter Seiten einer Datenbank typischerweise erheblich größer, als die Anzahl zur Verfügung stehenden Seitenrahmen.
- *Datenbankpuffer*: im Hauptspeicher für die Datenbank verfügbaren Seitenrahmen.
- $\Longrightarrow$ *Pufferverwaltung*: für angeforderte Seiten der Datenbank muss ein Seitenrahmen im Puffer gefunden werden; sonst *Seitenfehler*.
- Wann kann der Inhalt eines Rahmens durch eine neue Seite ersetzt werden?

- Es ist aus Effizienzgründen häufig sinnvoll, geänderte Seiten nicht direkt in die Datenbank zurück zuschreiben, bzw. ein *Prefetching* der Seiten vorzunehmen.
- Eine Änderung heißt *materialisiert*, wenn die entsprechende Seite in den externen Speicher der Datenbank zurückgeschrieben ist.

Adressierung

- *Tupelidentifikatoren* der Form $Tid = (b, s)$, wobei b eine Seitenummer und s eine Tupelnummer innerhalb der Seite b.
- Mittels eines in der betreffenden Seite abgelegten *Adressbuchs* (*Directory*) kann in Abhängigkeit des Wertes s die physische Adresse des Tupels in der Seite gefunden werden.
- Tupel sind lediglich in ihrer Seite frei verschiebbar.
- Verschiebbarkeit über Seitengrenzen mittels *Verweisketten*.
- Tupel mit variabel langen Attributwerten verlangen *Längenzähler*, bzw. eine *Zeigerliste*.

8.2 Dateiorganisationsformen und Indexstrukturen

Zugriffsarten

- *Durchsuchen* (*scan*) einer gesamten Relation.
- *Suchen* (*search*). Lesen derjenigen Tupel, die eine über ihren Attributen formulierte Bedingung erfüllen.

  Bereichssuchen (*range search*): in den Bedingungen über den Attributen stehen nicht ausschließlich Gleichheitsoperatoren, sondern beliebige andere arithmetische Vergleichsoperatoren.
- *Einfügen* (*insert*) eines Tupels in eine Seite.
- *Löschen* (*delete*) eines Tupels in einer Seite.
- *Ändern* (*update*) der Inhalt eines Tupels einer Seite.

Haufenorganisation

- Zufällige Anordnung der Tupel in den Seiten.
- Suchen eines Tupels benötigt im Mittel $\frac{n}{2}$ Seitenzugriffe.
- Einfügen eines Tupels benötigt 2 Seitenzugriffe.

Was gilt für den Aufwand für das Löschen und Ändern eines Tupels?

Suchbaum

- Die Tupel sind nach dem Suchschlüssel sortiert.
- Suchen eines Tupels benötigt eine logarithmische Anzahl Seitenzugriffe.
- Einfügen eines Tupels: s. später.

Hashverfahren

- Mittels einer Hashfunktion wird einem Suchschlüsselwert eine Seitennummer zugeordnet.
- Suchen eines Tupels benötigt in der Regel ein bis zwei Seitenzugriffe.
- Einfügen?

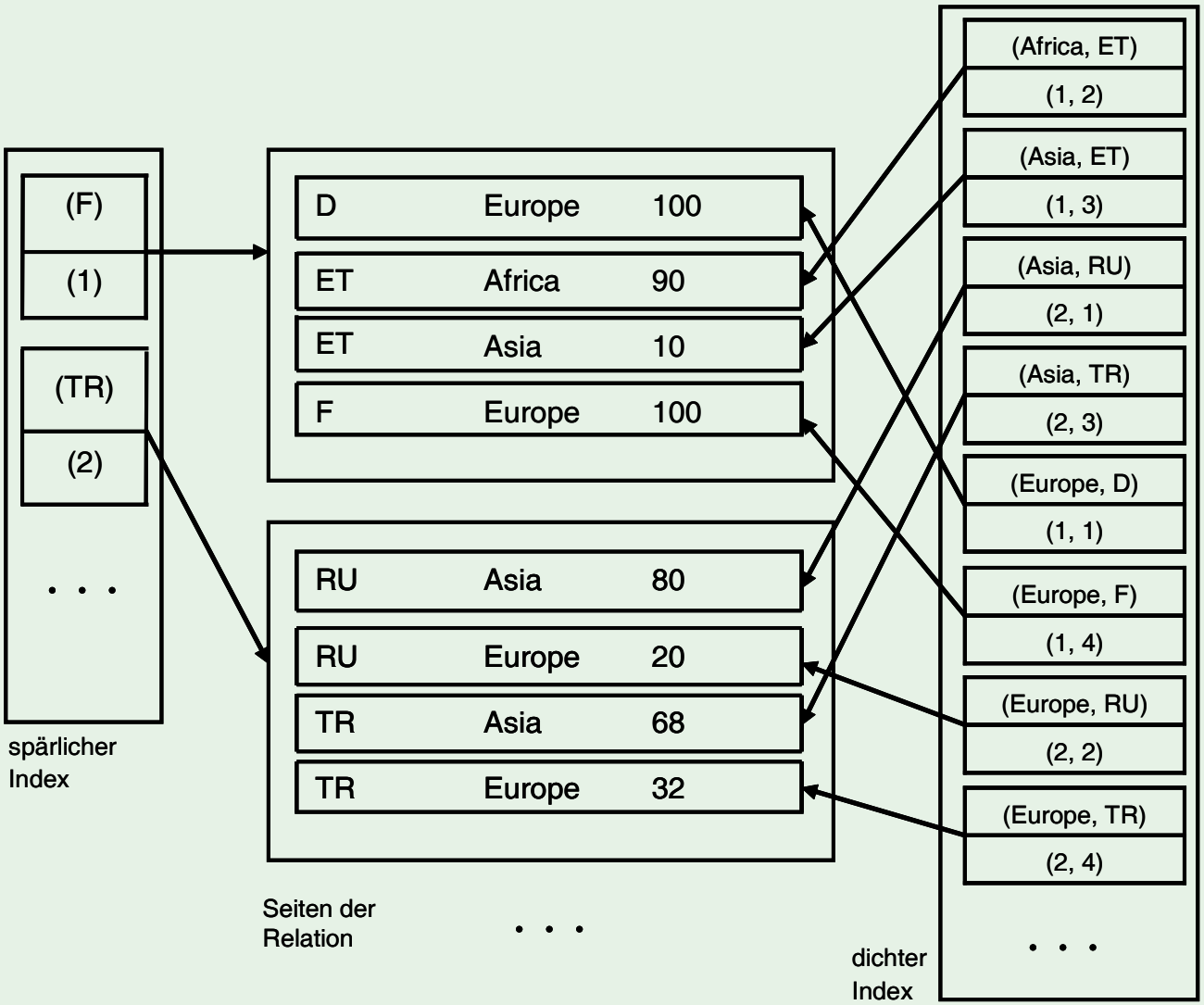
Primär- und Sekundärindex

- Ein *Suchschlüssel* ist die einem Index zugrunde liegende Attributkombination.
- Bei einem *Primärindex* ist der Primärschlüssel der Relation im Suchschlüssel enthalten.

  Sekundärindex anderenfalls. Ein Suchschlüsselwert identifiziert i.A. eine Menge von Tupeln.

Zu einer Relation existieren typischerweise mehrere Indexstrukturen über jeweils unterschiedlichen Suchschlüsseln.

Beispiel


(F)
(1)
(TR)
(2)
. . .
spärlicher Index
D Europe 100
ET Africa 90
ET Asia 10
F Europe 100
RU Asia 80
RU Europe 20
TR Asia 68
TR Europe 32
Seiten der Relation
. . .
(Africa, ET)
(1, 2)
(Asia, ET)
(1, 3)
(Asia, RU)
(2, 1)
(Asia, TR)
(2, 3)
(Europe, D)
(1, 1)
(Europe, F)
(1, 4)
(Europe, RU)
(2, 2)
(Europe, TR)
(2, 4)
. . .
dichter Index

weitere Indexformen

- *geballt*: logisch zusammengehörende Tupel sind auch physisch benachbart gespeichert.

  Vorteile?

- *dicht*: pro Suchschlüsselwert der Tupel der betreffenden Relation einen Verweis,
- *spärlich*: pro Intervall von Suchschlüsselwerten einen Verweis.

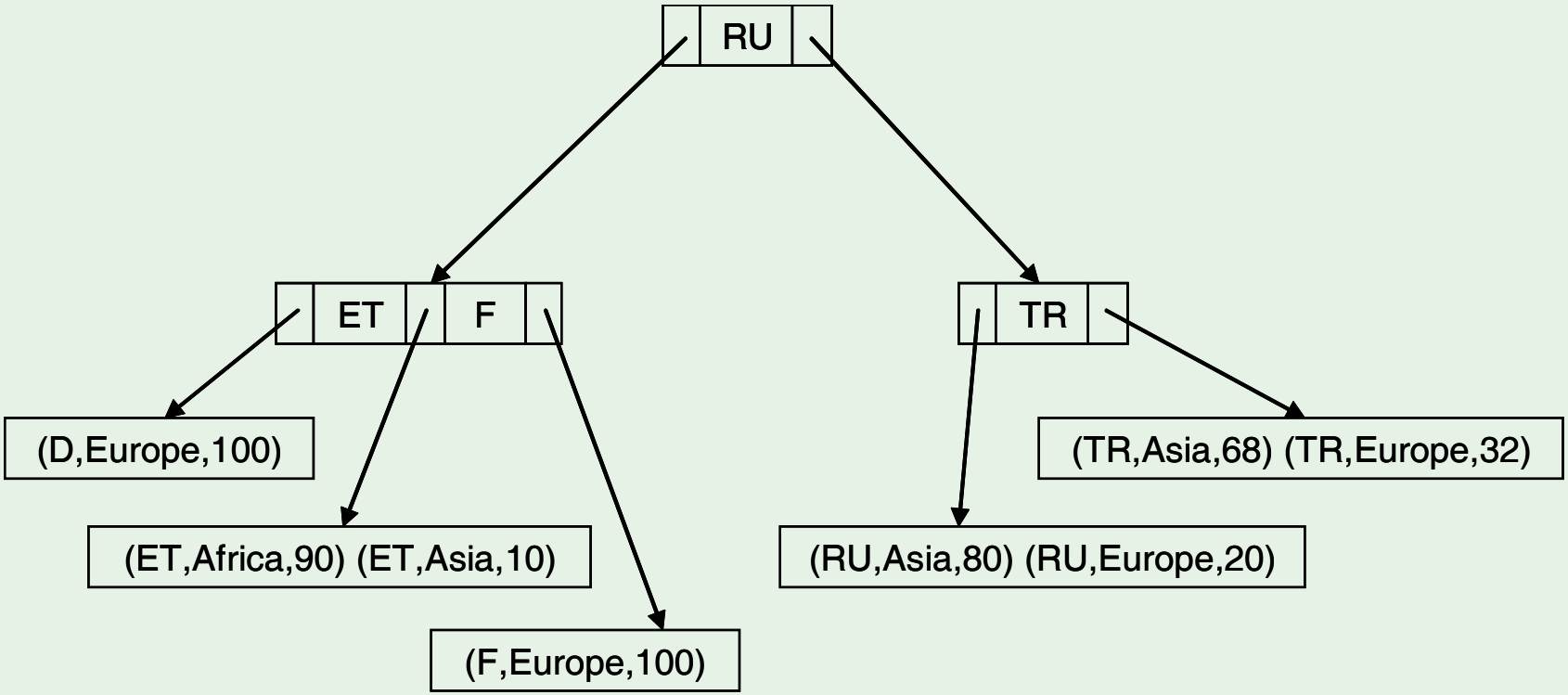
- Eine Relation heißt *invertiert* bzgl. einem Attribut, wenn ein dichter Sekundärindex zu diesem Attribut existiert,
- sie heißt *voll invertiert*, wenn zu jedem Attribut ein dichter Sekundärindex existiert.

8.3 Baum-Indexstrukturen

B-Baum der Ordnung (m, l); $m > 3$, $l \geq 1$ und m gerade.

- Die Wurzel ist entweder ein Blatt oder hat mindestens zwei direkte Nachfolger.
- Jeder innere Knoten außer der Wurzel hat mindestens $\frac{m}{2}$ und höchstens m direkte Nachfolger.
- Die Länge des Pfades von der Wurzel zu einem Blatt ist für alle Blätter gleich.
- Die inneren Knoten haben die Form $(p_0, k_1, p_1, k_2, p_2, \ldots, k_n, p_n)$, wobei $\frac{m}{2} \leq n \leq m - 1$. Es gilt:
  - p_i ist ein Zeiger auf den $i + 1$-ten direkten Nachfolger und jedes k_i ist ein Suchschlüsselwert, $0 \leq i \leq n$.
  - Die Suchschlüsselwerte sind geordnet, d.h. $k_i < k_j$, für $1 \leq i < j \leq n$.
  - Alle Suchschlüsselwerte im linken (rechten) Teilbaum von k_i sind kleiner (größer oder gleich) als der Wert von k_i, $1 \leq i \leq n$.
- Die Blätter haben die Form $(k_1^*, k_2^*, \ldots, k_g^*)$, wobei $g \leq l$ und k_i^* das Tupel mit Suchschlüsselwert k_i.

Beispiel



Eigenschaften

- $\lceil log_m N \rceil \leq h \leq \lfloor 1 + log_{\frac{m}{2}} \frac{N}{2} \rfloor$.
- Anzahl Externzugriffe für Suchen, Einfügen und Löschen: $O(h)$.

- Bereichsanfragen?
- Sekundärindex?

mehrattributiger Suchschlüssel

- Konkatenation der einzelnen Attributwerte. In welcher Reihenfolge?
- Unabhängige Indexstrukturen über den einzelnen Attributen. Nachbarschaftsbeziehungen?
- $\Longrightarrow$ mehrdimensionale Zugriffsstrukturen.

8.4 Hash-Indexstrukturen

- Eine gegebene Relation wird mittels einer Streuungsfunktion (*Hashfunktion*) h in Seiten aufgeteilt. Die Seiten seien numeriert von 0, 1, ..., $K-1$. Auf der Menge der Suchschlüsselwerte k sollte $h(k)$ möglichst gleichverteilt über 0, 1, ..., $K-1$ sein.
- Mittels h wird jedem Tupel in Abhängigkeit seines Suchschlüsselwertes k eine Seite $h(k) = k \ mod \ K$.

- Direkter Zugriff zu den Tupeln einer Relation mit einem einzigen Externzugriff, sofern hinreichend viele Seiten zur Verfügung gestellt werden und die Hashfunktion annähernd eine Gleichverteilung der Tupel in den Seiten bewirkt.
- Werden mehr Tupel einer Seite zugeordnet, als diese aufnehmen kann, so müssen der Seite *Überlaufseiten* zugeordnet werden.
- Typischerweise werden die Seiten im Mittel beim Aufbau eines Hashindex nur zu 80% gefüllt.
- Zugriff zu den Tupeln gemäß einer Sortierfolge wird nicht unterstützt.
- Bereichsanfragen?
- Sekundärindex?
- Behandlung mehrattributiger Schlüssel analog zu einem B-Baum.