Datenbanken

Teil 2: Informationen

Kapitel 9: DB_Entwurf

Speicherstrukturen

Vergleich unterschiedlicher Speicherstrukturen

1. Malwettbewerb
2. HEAP-Datei
3. Hash-Verfahren
4. ISAM
5. B*-Baum
6. Sekundärindex

Beispiel: Der Malwettbewerb

Alle Kinder die am Malwettbewerb teilnehmen möchten, können ein Bild mit ihrem Namen und Ihrem Alter versehen und abgeben.

Wie kann nach dem Malwettbewerb die Rückgabe der Bilder am geschicktesten organisiert werden?

Für die Ausgabe der Bilder wurde eigens eine ***Raum*** zur Verfügung gestellt.
In diesem Raum befinden sich viele kleine ***Tische***, auf denen die Bilder ausgebreitet werden können. (Auf jedem Tisch können bis zu drei Bilder abgelegt werden.)

Teilnehmen

An dem Malwettbewerb haben die folgenden 10 Kinder teilgenommen:

Uwe	(5),	Ulrike	(12),	Anke	(7),
Sabine	(8),	Antonia	(6),	Marvin	(13),
Anton	(6),	Tobias	(11),	Udo	(10),
Torsten	(9).				

In Klammen ist das jeweilige Alter des Kindes angegeben.

HEAP-Datei

Heap-Datei

Bei einer Heap-Datei werden die Datensätze unsortiert hintereinander (sequentiell) geschrieben.

In dem Beispiel wurden die Tische einfach hintereinander aufgestellt und die Zeichnungen darauf unsortiert ausgelegt.

Vorteil:

Das Auslegen der Zeichnungen ist sehr einfach.

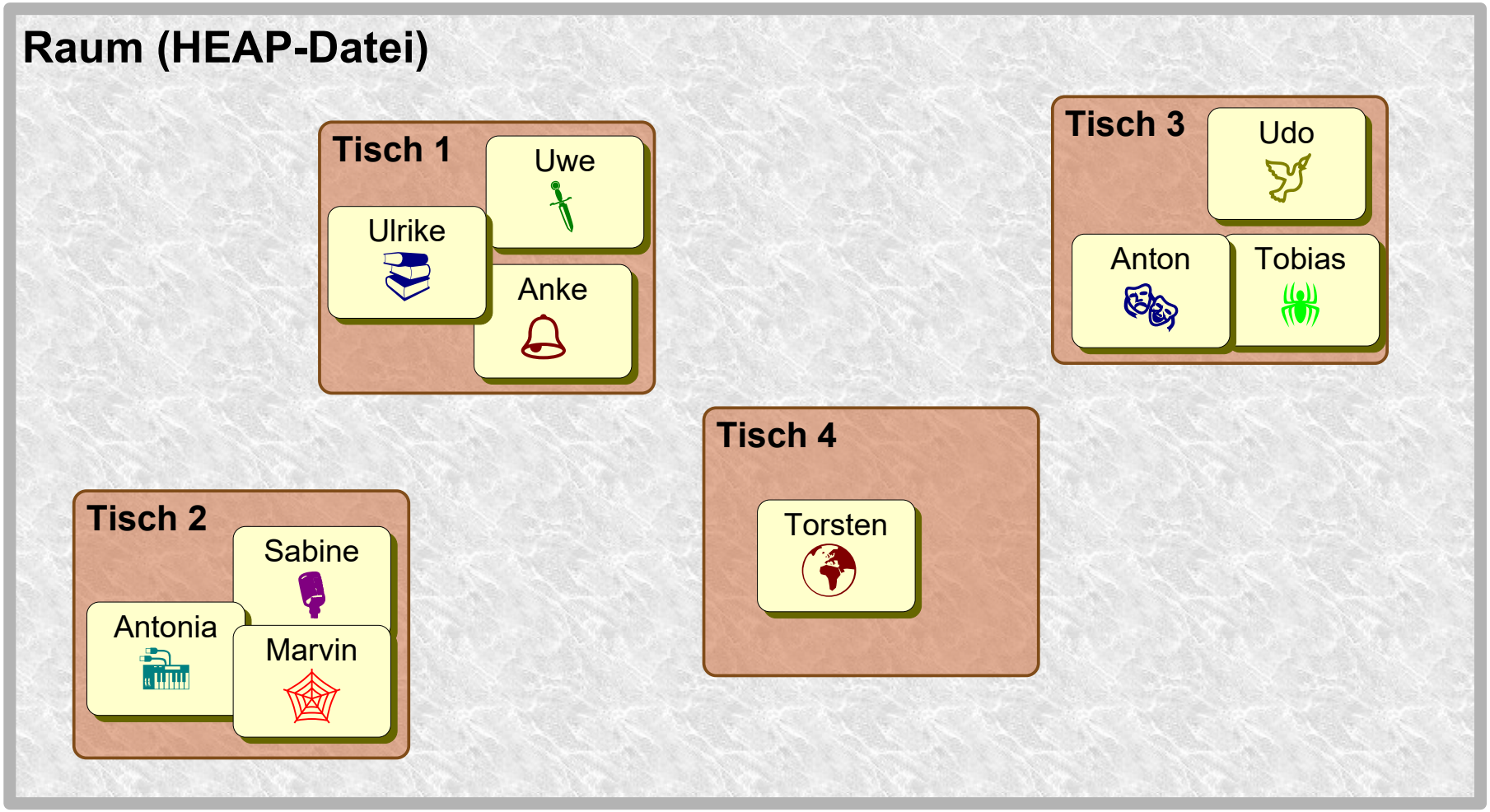
Nachteil:

Wenn ein Kind seine Zeichnung abholen möchte, muss es alle Tische nacheinander absuchen bis es seine Zeichnung gefunden hat.

Fazit:

Die einfache Heap-Datei ist für Abfrageoperationen extrem schlecht geeignet, da sie völlig unsortiert ist.

HEAP-Datei


Raum (HEAP-Datei)
Tisch 1
Uwe
Ulrike
Anke
Tisch 3
Udo
Anton
Tobias
Tisch 4
Torsten
Tisch 2
Sabine
Antonia
Marvin

Hash-Verfahren

Hash-Verfahren

Bei dem Hash-Verfahren werden die Datensätze unterschiedlichen Bereichen zugeordnet. Innerhalb der Bereiche sind die Datensätze wieder ungeordnet. Die Zuordnung der Datensätze zu den Bereichen erfolgt über eine Funktion (Hashfunktion).

In dem Beispiel werden die Bilder anhand der Länge der Kindernamen mit Hilfe der Modulo 3 Hash-Funktion auf drei Bereiche aufgeteilt.

Vorteil:

Das Auslegen der Zeichnungen ist immer noch einfach. Das Durchsuchen kann bis zu einem der Bereichsanzahl entsprechenden Faktor verringert werden.

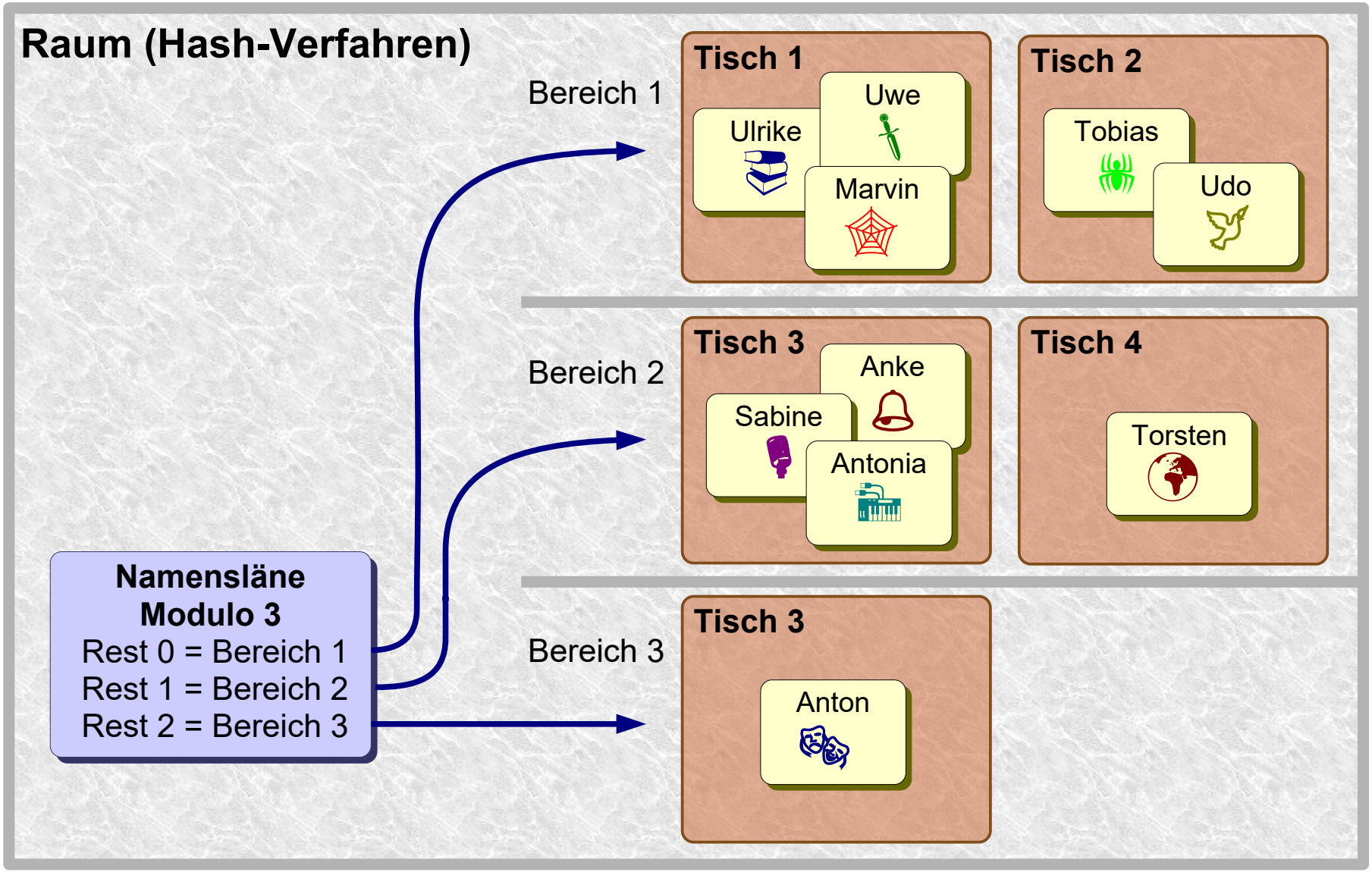
Nachteil:

Die einzelnen Bereiche müssen sequentiell durchsucht werden.

Fazit:

Das Hash-Verfahren ermöglicht ein einfaches Einfügen von Datensätzen. Auch das Löschen und Lesen von Daten ist im Vergleich zur Heap-Datei deutlich schneller. Nachteilig ist, dass eine sortierte Ausgabe der Daten durch dieses Verfahren nicht unterstützt wird.

Hash-Verfahren



ISAM-Verfahren

ISAM-Verfahren

Bei dem ISAM-Verfahren (**I**ndex **S**equential **A**ccess **M**ethod) wird neben der eigentlichen Datendatei, die die Datensätze in sortierter Reihenfolge enthält, eine weitere Datei, die so genannte **Index**-Datei gepflegt. Diese Index-Datei ermöglicht einen beschleunigten Zugriff auf die Daten.

In dem Beispiel werden die Zeichnungen alphabetisch nach Namen sortiert auf den Tischen ausgelegt. Über den alphabetisch ersten Name wird ein Index erstellt.

Vorteil:

Der richtige Tisch wird über den Index bestimmt. Die alphabetische Sortierung erleichtert die Suche auf dem Tisch.

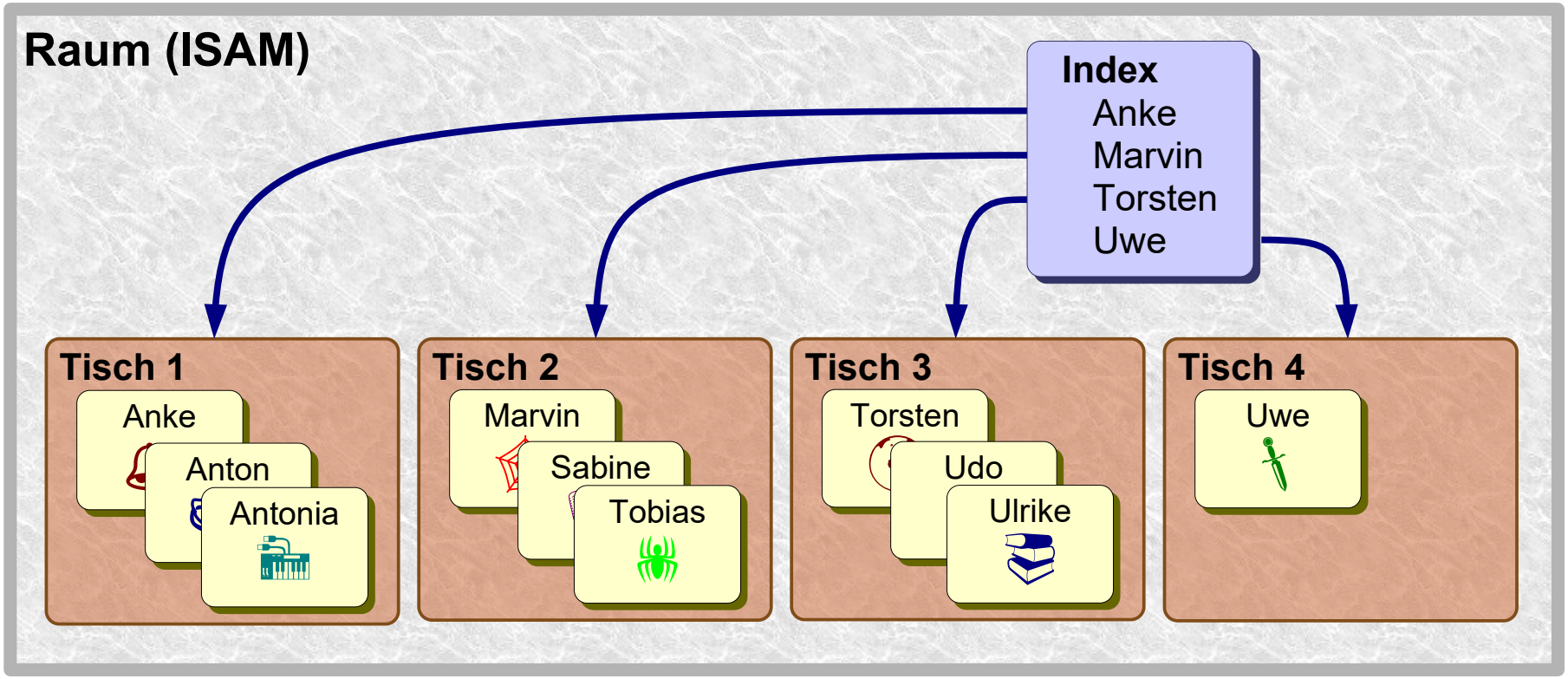
Nachteil:

Das Auslegen der Bilder ist wegen der Sortierung sehr aufwändig.
Die Indexdatei muss angepasst werden.

Fazit:

Durch die Sortierung der Datensätze und der Indexdatei
wird das Lesen von Daten deutlich effizienter.
Das Einfügen und Löschen von Daten wird deutlich aufwendiger.

ISAM


Raum (ISAM)
Index
Anke
Marvin
Torsten
Uwe
Tisch 1
Anke
Anton
Antonia
Tisch 2
Marvin
Sabine
Tobias
Tisch 3
Torsten
Udo
Ulrike
Tisch 4
Uwe

B*-Baum

Baumartige Struktur

Betrachtet man die Indexdateien des ISAM-Verfahrens wiederum als Datensätze, so kann zu diesen Datensätzen wiederum eine Indexdatei generiert werden. Durch diese Verschachtelung von Indexdateien kommt man zu einer baumartigen Struktur.

Merkmale eines B-Baumes:*

- Jeder Knoten des Baumes kann mehrere Kindknoten haben (Mehrwegbaum).
- Der Baum ist ausgeglichen (balancierter Baum).
- Die Knoten enthalten nur Verweise und nicht die eigentlichen Daten.

In dem Beispiel werden die Zeichnungen wieder sortiert auf den Tischen ausgelegt. Dann werden **mehrere** Teilindizes erzeugt, die jeweils mehrere benachbarte Tische umfassen. Über diese Teilindizes wird eine übergeordnete Indexdatei angelegt.

Vorteil:

Es entstehen keine „übergroßen“ Indexdateien.

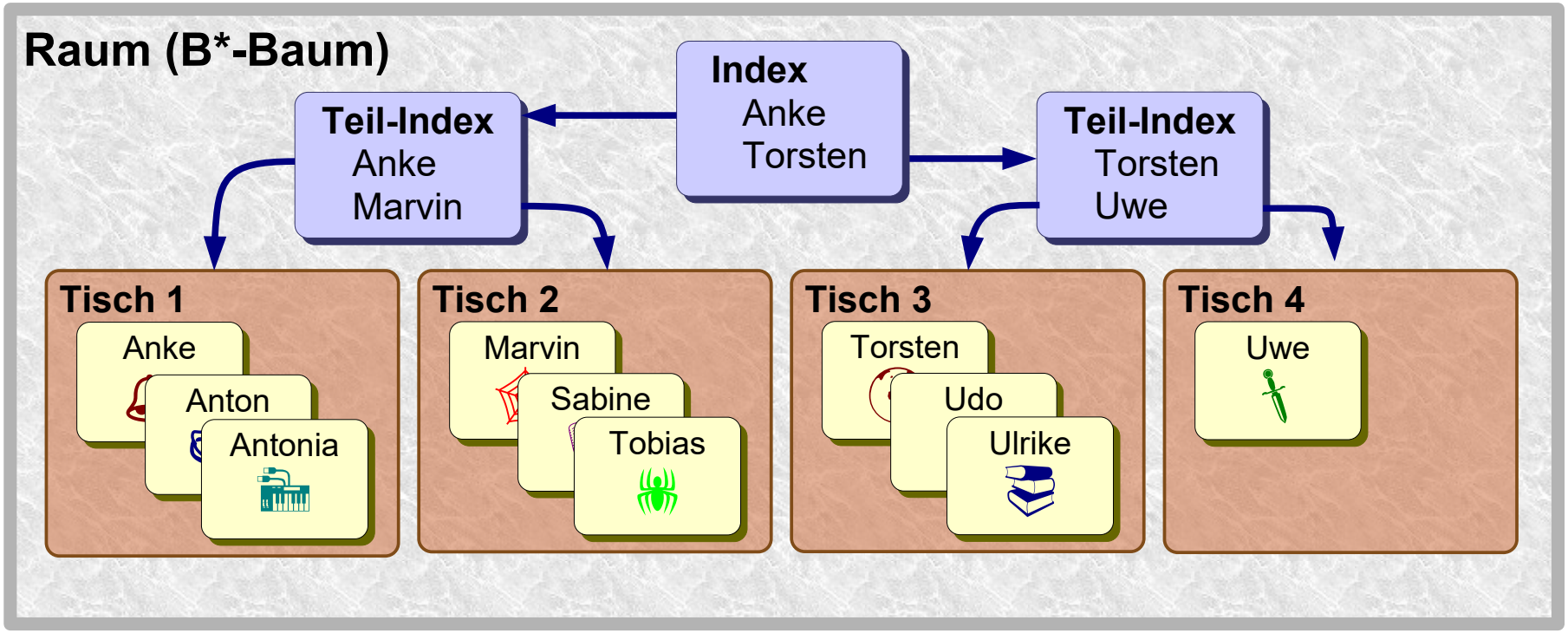
Nachteil:

Komplizierte Einfüge- und Löschoperationen.

Fazit:

Optimale Lösung für Anwendungen mit überwiegend lesendem Zugriff.

B*-Baum


Raum (B*-Baum)
Index
Anke
Torsten
Teil-Index
Anke
Marvin
Teil-Index
Torsten
Uwe
Tisch 1
Anke
Anton
Antonia
Tisch 2
Marvin
Sabine
Tobias
Tisch 3
Torsten
Udo
Ulrike
Tisch 4
Uwe

Sekundärindex

Sekundärindex

Ein Sekundärindex ist ein Index (Beispielsweise ein B*-Baum) über ein zusätzliches Merkmal.

In dem Beispiel wird ein Sekundärindex über das Alter der Kinder erzeugt. Dabei haben die Kinder folgende Alter:

Uwe 5,	Ulrike 12,	Anke 7,	Sabine 8,	Antonia 7,
Marvin 13,	Anton 6,	Tobias 11,	Udo 10,	Torsten 9

Vorteil:

Die Effizienz von lesenden Zugriffen bezüglich des zusätzlichen Merkmals können erheblich gesteigert werden. Der Primärindex bleibt unverändert.

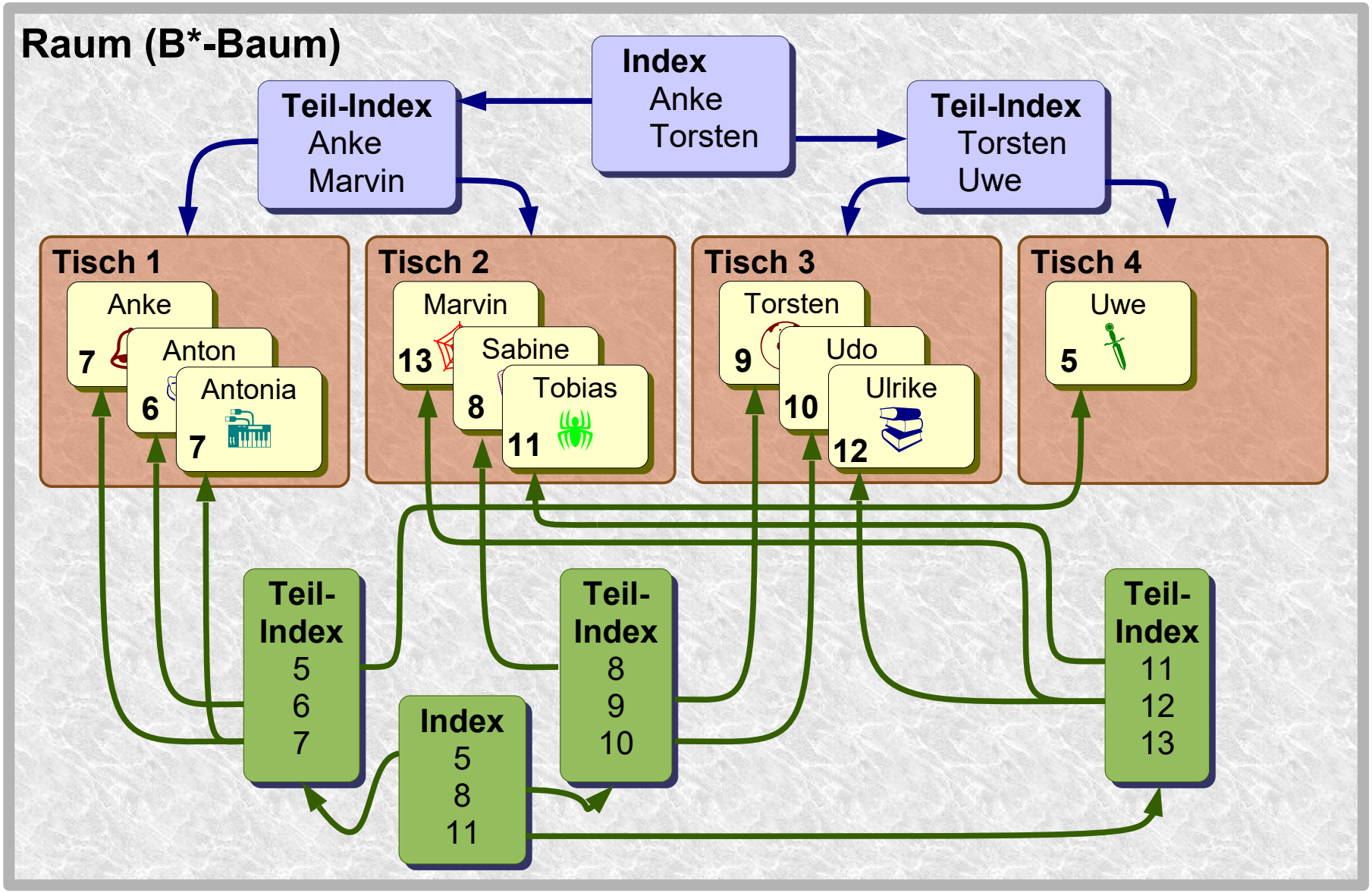
Nachteil:

Einfüge- und Löschoperationen werden nochmals komplizierter, da mehrere Indizes angepasst werden müssen.

Fazit:

Optimale Lösung zur Beschleunigung von lesenden Zugriffen auf beliebige weitere Merkmale. Der Primärindex bleibt davon unbeeinflusst.

Sekundärindex



Vergleich Speicherstrukturen

Eignung der Speicherstrukturen für Lese- und Einfügeoperationen.

LESE-Operation	EINFÜGE-Operation	Geeignete Datenorganisation
viel  wenig	wenig  viel	

Vergleich Speicherstrukturen

Eignung der Speicherstrukturen für Lese- und Einfügeoperationen.

LESE-Operation	EINFÜGE-Operation	Geeignete Datenorganisation
viel  wenig	wenig  viel	***B*-Baum*** Indexdateien auf Indexdateien

Vergleich Speicherstrukturen

Eignung der Speicherstrukturen für Lese- und Einfügeoperationen.

LESE-Operation	EINFÜGE-Operation	Geeignete Datenorganisation
viel  wenig	wenig  viel	***B*-Baum*** Indexdateien auf Indexdateien
		ISAM Sortierte Datei mit Indextabelle

Vergleich Speicherstrukturen

Eignung der Speicherstrukturen für Lese- und Einfügeoperationen.

LESE-Operation	EINFÜGE-Operation	Geeignete Datenorganisation
viel  wenig	wenig  viel	***B*-Baum*** Indexdateien auf Indexdateien
		ISAM Sortierte Datei mit Indextabelle
		Hashing Aufteilung in unsortierte Bereiche

Vergleich Speicherstrukturen

Eignung der Speicherstrukturen für Lese- und Einfügeoperationen.

LESE-Operation	EINFÜGE-Operation	Geeignete Datenorganisation
viel  wenig	wenig  viel	***B*-Baum*** Indexdateien auf Indexdateien
		ISAM Sortierte Datei mit Indextabelle
		Hashing Aufteilung in unsortierte Bereiche
		Heap-Datei Einfache sequentielle Datei