

[illegible]

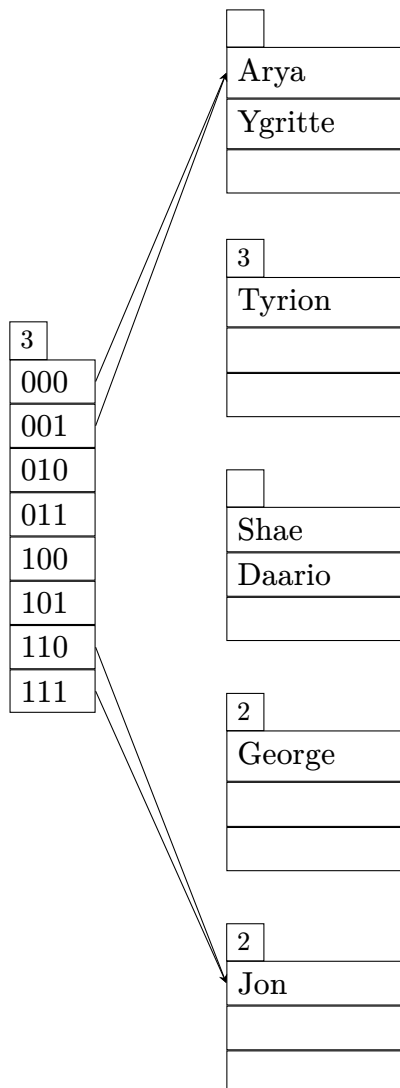
Aufgabe 2 - Erweiterbares Hashing.

1 Punkt

Die Hashfunktion h liefert die in der Tabelle angegebenen Binärwerte. **Ergänzen Sie die fehlenden Werte/Komponenten (globale/lokale Tiefen, Pointer).**

Hash-Tabelle:

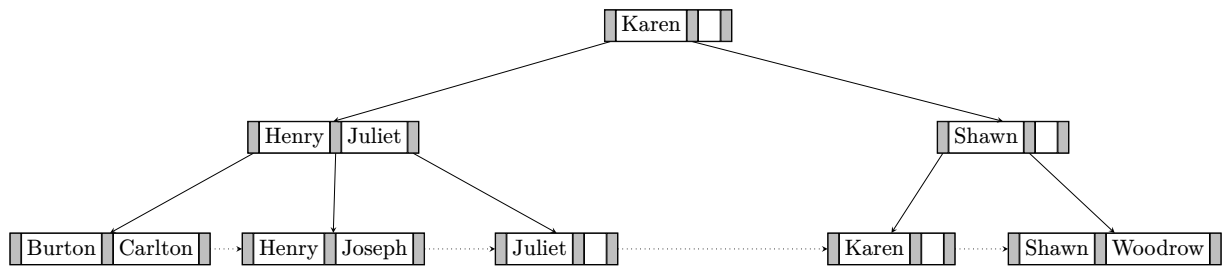
Name (x)	$h(x)$
Arya	0010
Cersei	0100
Daario	0110
George	1000
Jon	1100
Melisandre	1110
Robb	1011
Shae	0111
Tyrion	0101
Ygritte	0001



Aufgabe 3 - B^+ -Baum Einfügen.

1 Punkt

Gegeben ist ein B^+ -Baum mit $m = 3$. Zeichnen Sie den B^+ -Baum, der nach dem Einfügen von **Isaac** entsteht.



Aufgabe 4 - *Indexstrukturen mit kombiniertem Suchschlüssel.*1 Punkt

Gegeben ist ein geordneter Index mit kombiniertem Suchschlüssel auf die Attribute (*LanguageFamily*, *Language*, *NumSpeaker*) in dieser Reihenfolge. Geben Sie für die folgenden Anfrageprädikate an, ob eine effiziente Nutzung des Index zur Beantwortung der Anfrage möglich ist, und begründen Sie jeweils kurz Ihre Antwort.

1. **WHERE** *Language*="Spanish" **AND** *LanguageFamily*="Romance"
2. **WHERE** *Language*="Mandarin" **AND** $10^7 < \text{NumSpeaker} < 10^9$
3. **WHERE** *LanguageFamily*="Germanic" **AND** *Language*="English"
AND $10^6 < \text{NumSpeaker} < 10^7$
4. **WHERE** *LanguageFamily*="Slavic"

Aufgabe 5 - *Bitmap Index Scan*.1 Punkt

Gegeben sei eine Relation $R[A, B]$ mit folgenden Eigenschaften:

- $|R| = 10^6$ Tupel gespeichert auf $b_R = 10^3$ Blöcken,
- Non-clustering B⁺-Baum-Index auf Attribut B , $m = 1024$, maximale Höhe (d.h. die Knoten sind minimal befüllt),
- Duplikate werden mittels Tuple Identifier (TID) aufgelöst,
- Attribut B hat insgesamt 10^3 mal den Wert 2500, verteilt auf 100 Datenblöcke.

Es soll folgende Anfrage beantwortet werden:

$$\sigma_{B=2500}(R)$$

Geben Sie die Anzahl an **Datenblockzugriffen (worst case)**

- a. **ohne Anwendung** eines Bitmap Index Scans (**0.5 Punkte**),
- b. **unter Anwendung** eines Bitmap Index Scans (**0.5 Punkte**)

an.

Aufgabe 6 - *Join-Algorithmen*.1 Punkt

Welcher Join Algorithmus (**Merge Join**, **Index Nested Loop Join**) generiert die minimalen Kosten für das folgende Szenario? Geben Sie in der Lösung die **Algorithmen** und die **dazugehörigen Kosten** an.

Berechnen Sie einen **natürlichen Join** zwischen zwei Relationen $R[A, B]$ und $S[A, C]$ mit $|R| = 1000$ Tupel und $|S| = 12000$ Tupel. Die Relationen sind auf $b_R = 250$ bzw. $b_S = 2000$ hintereinander liegenden Blöcken gespeichert. Es existiert ein **sparse** B⁺-Baum Index auf $S.A$, wobei jeder Knoten im B⁺-Baum 20 Schlüssel speichern kann. Auf $R.A$ existiert kein Index. Nehmen Sie an, dass das Sortieren von b Blöcken $\lceil b \cdot \log_2(b) \rceil$ Blockzugriffe kostet.

Aufgabe 7 - Effiziente Anfragebearbeitung.

1 Punkt

Gegeben ist die Relation $R[A]$. Auf $R.A$ existiert ein sparse B⁺-Baum Index. Was ist die **effizienteste Strategie** um Bereichsanfragen von folgendem Typ zu beantworten?

$$\sigma_{a < A < b}(R)$$

Geben Sie **alle notwendigen Schritte** an.

Aufgabe 8 - *Schätzung der Join-Kardinalität.*1 Punkt

Gegeben seien 3 Relationen $R[A, B, C]$, $S[A, D, E]$, $T[D, E, F]$ mit folgenden Eigenschaften:

- $|R[A, B, C]| = 1000$ Tupel, $V(R, A) = 100$, $V(R, B) = 200$, $V(R, C) = 300$
- $|S[A, D, E]| = 4000$ Tupel, $V(S, A) = 50$, $V(S, D) = 200$, $V(S, E) = 300$
- $|T[D, E, F]| = 2000$ Tupel, $V(T, D) = 200$, $V(T, E) = 400$, $V(T, F) = 600$

Die Werte in den Tupeln sind gleichverteilt und unabhängig. Schätzen Sie die Kardinalität der folgenden Abfrage ab. ($\sigma_{A=100}(R) \neq \emptyset$).

$$(\sigma_{A=100}(R)) \bowtie S \bowtie T$$

Aufgabe 9

1 Punkt

Betrachten Sie die folgende Historie (Schedule). Geben Sie für folgende Eigenschaften an, ob die Historie diese jeweils erfüllt:

1. konfliktserialisierbar,
2. rücksetzbar (recoverable),
3. cascadeless

Begründen Sie Ihre Antwort!

T1:	T2:	T3:
write(A)		
	read(D)	
		write(A)
		write(D)
write(D)		
commit		
	write(C)	
		read(A)
		commit
	commit	

Aufgabe 101 Punkt

Kann die folgende Historie (Schedule) Ausgabe eines Schedulers mit **Strict Two-Phase-Locking** sein? Falls ja, geben Sie alle notwendigen Lock/Unlock-Operationen an. Erklären Sie ansonsten, warum diese Historie unmöglich ist.

T1:

T2:

T3:

read(A)

read(B)

read(A)

write(A)

write(B)

COMMIT

read(A)

read(B)

read(B)

read(A)

COMMIT

COMMIT
