TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Dr. Thomas Neumann

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS14/15

Harald Lang (harald.lang@in.tum.de)
http://www-db.in.tum.de/teaching/ws1415/grundlagen/

Blatt Nr. 10

Hausaufgabe 1

Gegeben sei eine erweiterbare Hashtabelle mit globaler Tiefe t. Wie viele Verweise zeigen vom Verzeichnis auf einen Behälter mit lokaler Tiefe t'?

In dem Verzeichnis einer Hashtabelle mit globaler Tiefe t werden t Bits eines Hashwerts für die Identifizierung eines Verzeichniseintrags verwendet. Für einen Behälter mit lokaler Tiefe t' sind hingegen nur die ersten t' Bits dieses Bitmusters relevant.

Mit anderen Worten bedeutet dies, dass alle Einträge, die einen Behälter mit lokaler Tiefe t' referenzieren, in den ersten t' Bits übereinstimmen. Da alle Bitmuster bis zur Länge t in dem Directory aufgeführt sind, unterscheiden sich diese Einträge in den letzten $t - t'$ Bits.

$\Rightarrow$ Es gibt somit $2^{t-t'}$ Einträge im Verzeichnis, die auf denselben Behälter mit lokaler Tiefe t' verweisen.

Hausaufgabe 2

Es sollen alle ca. 10 Milliarden Menschen in einer erweiterbaren Hashtabelle verwaltet werden. In jede Seite passen ca. 200 Einträge, durchschnittlich sind die Seiten halb voll. Je Verweis werden 4 Byte benötigt, da die Musterlösung aus einer Zeit stammt, in der es defakto nur Maschinen mit 32 bit CPU Architektur gab. Wie viel Speicherplatz verbraucht das Verzeichnis mindestens?

Das Verzeichnis enthält die Verweise auf alle Seiten (= Buckets), in dem die Einträge gehalten werden. Da pro Seite durchschnittlich 100 Einträge Platz haben, benötigen wir insgesamt $10^{10}/100 = 10^8$ Seiten. Um 10^8 Seiten zu referenzieren benötigen wir mindestens $\log_2 10^8$ Bits. Da dies eine positive ganze Zahl sein muss, ist die Anzahl der benötigten Bits $\lceil \log_2 10^8 \rceil$. Hiermit können $2^{\lceil \log_2 10^8 \rceil}$ Verweise im Verzeichnis abgelegt werden, da die Anzahl der Verweise in einem Verzeichnis immer einer 2er-Potenz entspricht. Pro Verweis werden 4 Byte benötigt, so dass das Verzeichnis eine Größe von $2^{\lceil \log_2 10^8 \rceil} \cdot 4$ Byte, also ungefähr 512 MB hat.

Hausaufgabe 3

Gegeben sei die folgende SQL-Anfrage:

```
select a.PersNr, a.Name
from Assistenten a, Studenten s, pruefen p
where s.MatrNr = p.MatrNr
  and a.Boss = p.PersNr
  and s.Name = 'Jonas';
```

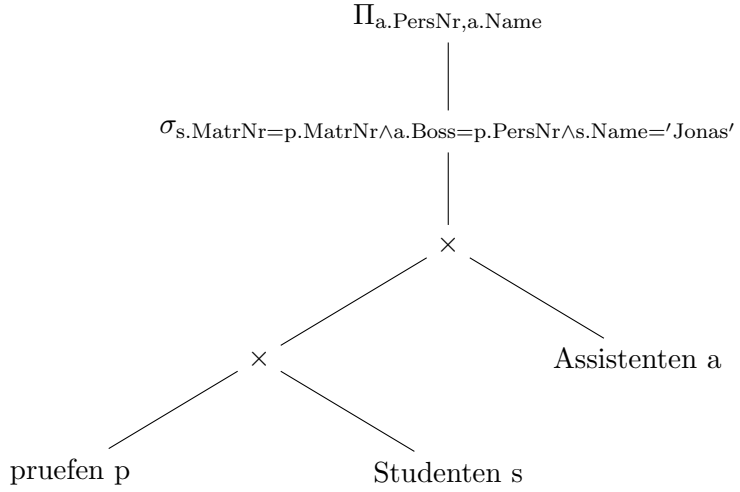
Geben Sie die kanonische Übersetzung dieser Anfrage in die relationale Algebra an. Verwenden Sie zur Darstellung des relationalen Algebraausdrucks die Baumdarstellung.

Optimieren Sie Ihren relationalen Algebraausdruck logisch. Gehen Sie dabei von **realistischen** Kardinalitäten für die relevanten Relationen aus.

Verwenden Sie hierfür die folgenden aus der Vorlesung bekannten Optimierungstechniken:

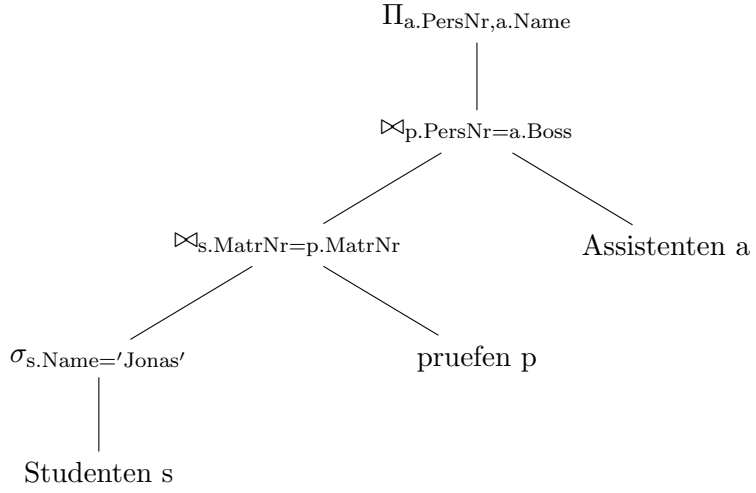
- Aufbrechen von Selektionen
- Verschieben von Selektionen nach "unten" im Plan
- Zusammenfassen von Selektionen und Kreuzprodukten zu Joins
- Bestimmung der Joinreihenfolge

Kanonische Übersetzung:



realistische Kardinalitäten: nur ein Student mit dem Namen 'Jonas', viel mehr Assistenten, deshalb Studenten zuerst, dann über pruefen mit Assistenten joinen

logische Optimierung: Selektionen ganz nach unten, Joins statt Kreuzprodukte & in richtiger Reihenfolge



Gruppenaufgabe 1 (muss nicht Zuhause vorbereitet werden)

Zum CAP-Theorem hieß es in der Vorlesung, dass in verteilten Systemen nur zwei der drei "Wünsche" (Konsistenz, Verfügbarkeit und Partitionstoleranz) gleichzeitig erfüllbar sind.

Welche der drei Kombinationen CA, CP, und AP sind jedoch sehr ähnlich?

Das CAP-Theorem beschreibt das Problem, dass bei einem verteilten System nur zwei der drei wichtigen Eigenschaften Konsistenz (**C**onsistency), Verfügbarkeit (**A**vailability),

und Partitionstoleranz (**P**artition tolerance) eingehalten werden können. Dementsprechend sollte es drei verschiedene Arten von Systemen geben:

CA: Konsistent und verfügbar, aber nicht partitionstolerant.

CP: Konsistent und partitionstolerant, aber nicht verfügbar.

AP: Verfügbar und partitionstolerant, aber nicht konsistent.

Wie Daniel Abadi in einem Blogartikel[1] beschreibt, gibt es jedoch keinen praktischen Unterschied zwischen CP- und CA-Systemen. Ein CP-System gibt die Verfügbarkeit auf, wenn das Netzwerk partitioniert ist (der Definition nach könnte es auch nie verfügbar sein; ein solches System macht jedoch wenig Sinn). CA-Systeme tolerieren per Definition Netzwerkpartitionen nicht. Aber was bedeutet das? In der Praxis ist es so, dass auch bei einem CA-System die Verfügbarkeit verloren geht, sobald das Netzwerk partitioniert ist. Auf diese Weise ist sichergestellt, dass bei Wiederherstellung des Netzwerkkonnektivität wieder CA erfüllt werden kann.

Es gibt daher nur zwei Arten von Systemen: CP/CA und AP. Es stellt sich nur die folgende Frage: Wie reagiert das System, wenn das Netzwerk partitioniert wird? Entweder das System gibt die Verfügbarkeit auf (CP/CA) oder die Konsistenz (AP).

Als Schlussbemerkung, sei noch darauf hingewiesen, dass CAP nichts über die Performanz des Systems aussagt. Es kann Sinn machen, sowohl die Verfügbarkeit als auch die Konsistenz einzuschränken, um eine höhere Leistungsfähigkeit zu erreichen (z.B. eine geringere Latenz). Mehr zu diesem Thema findet sich in dem oben erwähnten Blogeintrag von Daniel Abadi.

[1] `http://dbmsmusings.blogspot.de/2010/04/problems-with-cap-and-yahoos-little.html`