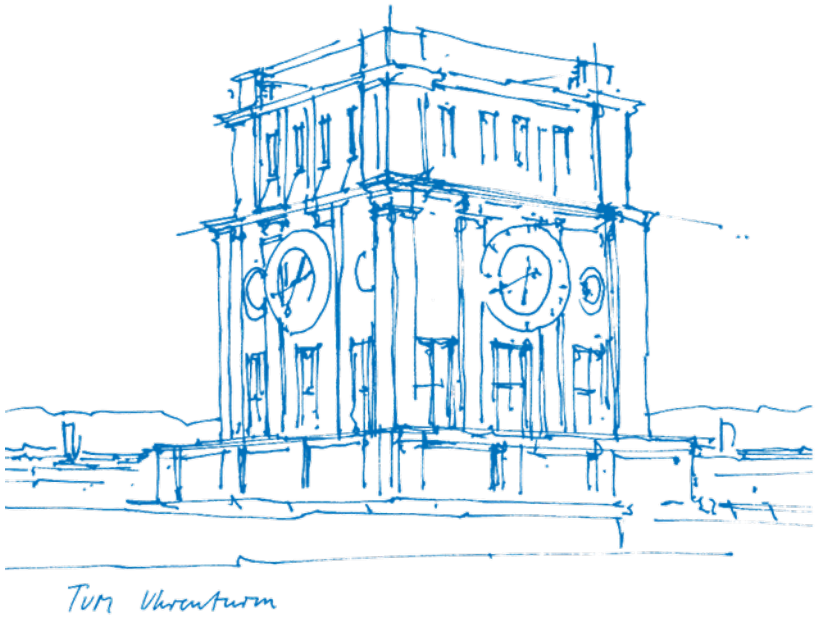
Grundlagen Datenbanken

Benjamin Wagner

15. Januar 2019



Allgemeines

- Folien von mir sollen unterstützend dienen. Sie sind nicht von der Übungsleitung abgesegnet und haben keinen Anspruch auf Vollständigkeit (oder Richtigkeit).
- Bei Fragen oder Korrekturvorschlägen: wagnerbe@in.tum.de
- Vorlesungsbegleitendes Buch von Professor Kemper (Chemiebib)
- Mein Foliensatz ist online: `https://github.com/wagjamin/GDB2018`

Anfrageoptimierung

- Wenn wir eine Anfrage an die Datenbank stellen, muss diese abgearbeitet werden
- Die Datenbank muss einen Plan erstellen, wie sie die Anfrage abarbeiten soll
- Dies wird in relationaler Algebra ausgedrückt
- **Aber:** verschiedene Algebrabäume können logisch äquivalent sein

$\Rightarrow$ Die Datenbank muss einen möglichst effizienten Plan finden

Anfrageoptimierung - Erste Idee

- Beginn: übersetzen die Anfrage in einen beliebigen relationalen Plan
- Diesen Plan können wir dann mit Umformungsregeln modifizieren
- **Frage:** was sind Beispiele für solche Regeln?
- Suche Heuristiken, welche Regeln zu "billigeren"Plänen führen
- Wunsch: Zwischenergebnisse sollen klein sein

Anfrageoptimierung - Erste Idee

- Beginn: übersetzen die Anfrage in einen beliebigen relationalen Plan
- Diesen Plan können wir dann mit Umformungsregeln modifizieren
- **Frage:** was sind Beispiele für solche Regeln?
- Suche Heuristiken, welche Regeln zu "billigeren"Plänen führen
- Wunsch: Zwischenergebnisse sollen klein sein

1. Selektionen aufbrechen
2. Selektionen nach unten schieben
3. Kreuzprodukte & Selektionen zu Joins verschmelzen
4. Joinreihenfolge bestimmen (klein: links, groß: rechts)

Korrelierte Unteranfragen

- Warum ist folgende Anfrage bei der Ausführung potentiell langsam?

```
1 SELECT s.Name, p.VorlNr
2 FROM Studenten s , prüfen p
3 WHERE s.MatrNr = p.MatrNr AND p.Note = (
4         SELECT MIN(p2.Note)
5         FROM prüfen p2
6         WHERE s.MatrNr=p2.MatrNr )
```

- **Frage:** wie lösen wir dieses Problem?

Ausführung der Anfrage

- Wir haben nun einen (hoffentlich) guten Query-Plan
- Dieser muss von der Datenbank ausgeführt werden
- Klassisch: jeder Operator wird implementiert
- Interface eines Operators: *open, close, next*
- Die Tupel werden dann bottom-up durch den Iteratorbaum geschleust
- **Ausblick:** das ist leider sehr ineffizient, Hyper kompiliert stattdessen den Operatorbaum in Maschinencode
- Wir müssen nun noch die einzelnen Operatoren implementieren

Joins - Algorithmen

- Der Join ist einer der wichtigsten Operatoren

$\Rightarrow$ Wir brauchen eine effiziente Implementierung!

- **Frage:** welche Join-Algorithmen gibt es?

Joins - Algorithmen

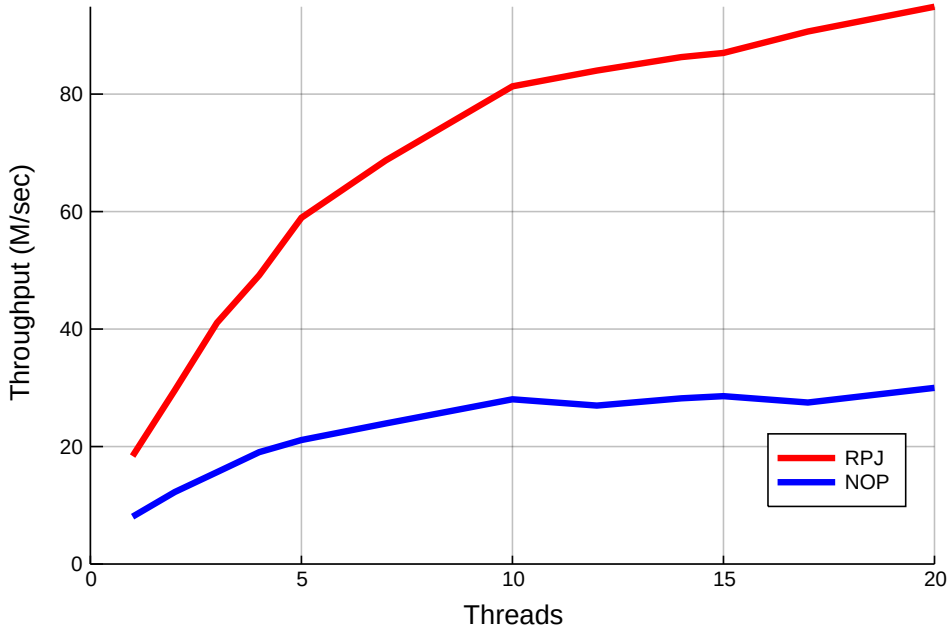
- Der Join ist einer der wichtigsten Operatoren

$\Rightarrow$ Wir brauchen eine effiziente Implementierung!

- **Frage:** welche Join-Algorithmen gibt es?
- Nested-Loop-Join, Hash-Join, Sort-Merge-Join
- In Hauptspeicherdatenbanken sind i.d.R. Hash-Joins am schnellsten
- **Aber:** selbst für Hash-Joins gibt es viele Implementierungen

Haupspeicher Hash Joins - Algorithmen

- Zwei konkurrierende Hash Join Implementierungen
- Gleiche Daten, alles im Hauptspeicher



S uniformly distributed, $|R| = |S| \approx$16.8M. Quelle.