Datenbanken: Lebenszyklus einer Query

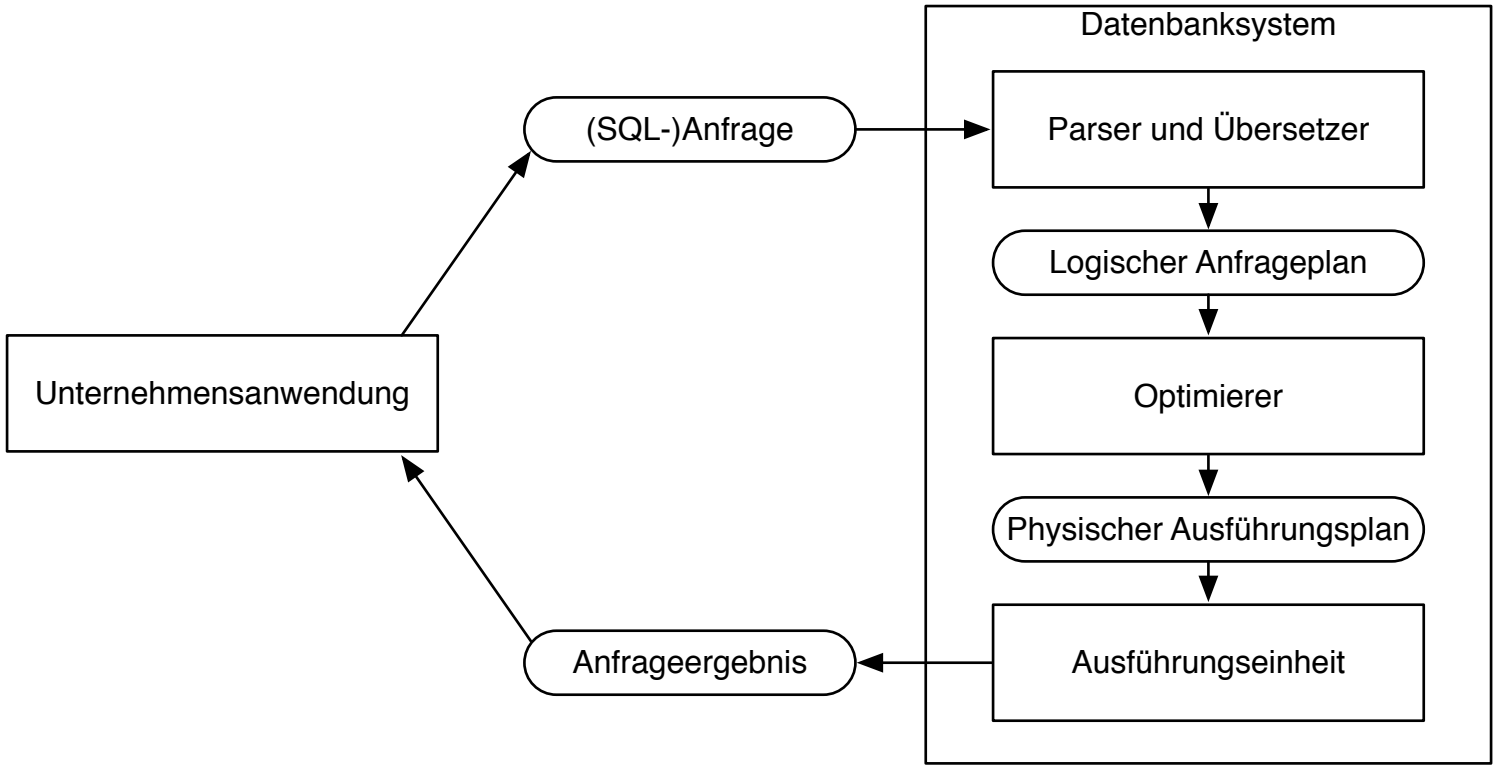
Stefan Halfpap, Ralf Teusner, Werner Sinzig

18. Mai 2020

Vorlesungsinhalte/-aufbau

- Einführung zu Unternehmensanwendungen
- Grundlagen des IT-gestützten Rechnungswesens und der Planung
- **Einführung zu relationalen Datenbanken und Anfrageverarbeitung**
- Grundlagen von (spaltenorientierten) Hauptspeicherdatenbanken
- Trends in Hauptspeicherdatenbanken
- Klausur

Überblick


Datenbanksystem
(SQL-)Anfrage
Parser und Übersetzer
Logischer Anfrageplan
Optimierer
Physischer Ausführungsplan
Unternehmensanwendung
Anfrageergebnis
Ausführungseinheit

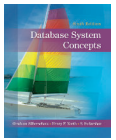
Überblick

- Client-Server-Kommunikation (Unternehmensanwendung-Datenbanksystem)
- Anfragebearbeitung
  - Parsing und Übersetzung
  - Optimierung
  - Ausführung
- Zusammenfassung

Literatur und Empfehlung

- Hector Garcia-Molina „Database Systems”

- Avi Silberschatz „Database System Concepts“



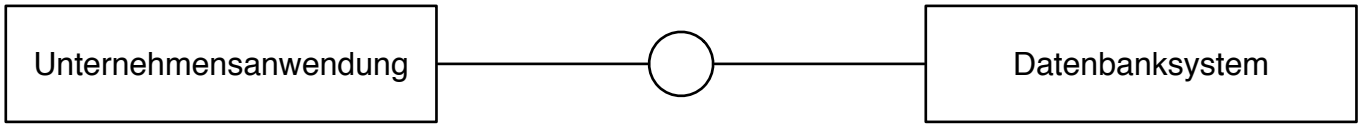
  http://db-book.com/
- Felix Naumann „Datenbanksysteme 1“
- Standford‘s CS 145 „Data Management and Data Systems“ https://cs145-fa18.github.io/
- CMU‘s 15-445 „Database Systems“ https://15445.courses.cs.cmu.edu/fall2018/
- PostgreSQL Dokumentation https://www.postgresql.org/docs/
- M. Dreseler et al. „Hyrise Re-engineered: An Extensible Database System for Research in Relational In-Memory Data Management“ (2019)
- J. Urbanski „Postgres on the wire“ https://wulczer.org/postgres-on-the-wire.pdf

Client-Server-Kommunikation

Kommunikationsschritte

- TCP-Verbindung aufbauen
- (optional) TLS-Setup
- DB-Verbindungsaufbau (DB, Protokoll-Version)
- (optional) Authentifizierung z.B. über Passwort
- → Anfrage senden
- ← Ergebnis erhalten
- DB-Verbindung beenden
- TCP-Verbindung beenden

Wire-Protokoll



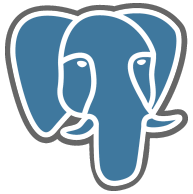
Client-Server-Kommunikation
Wire-Protokoll

- Alle großen Datenbanksysteme implementieren ihr eigenes, häufig geschütztes Wire-Protokoll
  - SAP HANA SQL Command Network Protocol Reference

- Die meisten neuen Systeme implementieren häufig Protokolle freier /Open-Source- Systeme
  - https://www.postgresql.org/docs/11/protocol.html
  - Wiederverwendung existierender Clients und Treiber
  - Bewährtes Design (eventuell Wiederverwendung der Server-Netzwerk-Schnittstelle)
  - Aber: gleiches Wire-Protokoll bedeutet nicht immer kompatible Systeme z.B. unterschiedliche SQL-Dialekte und SQL-Funktionalität

Client-Server-Kommunikation
Existierende Protokolle

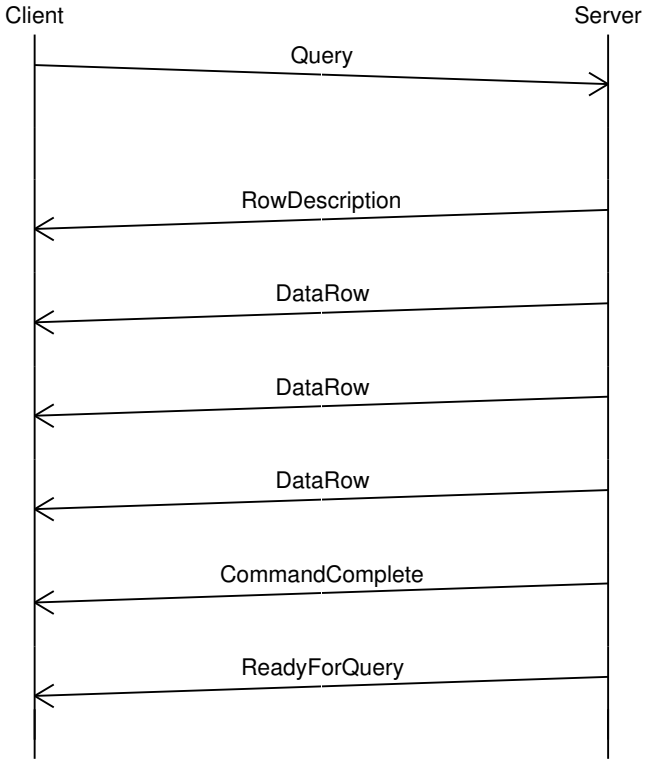
https://15721.courses.cs.cmu.edu/spring2019/slides/13-networking.pdf



VERTICA

Client-Server-Kommunikation
PostgreSQL Wire-Protokoll

- Nachrichtenbasiertes Protokoll
- Jede Nachricht (außer die Initiale) hat folgendes Format:
  - Kennung/Typ, Länge, Daten
  - Daten hängen von der Kennung ab



J. Urbanski „Postgres on the wire“ https://wulczer.org/postgres-on-the-wire.pdf

Client-Server-Kommunikation
PostgreSQL Wire-Protokoll - Nachrichtentypen

Query

'Q'	int32 len	str query

RowDescription

'T'	int32 len	int16 numfields

+

str col	int32 tableoid	int16 colno	int32 typeoid	int16 typelen	int32 typmod	int16 format

. . .

DataRow

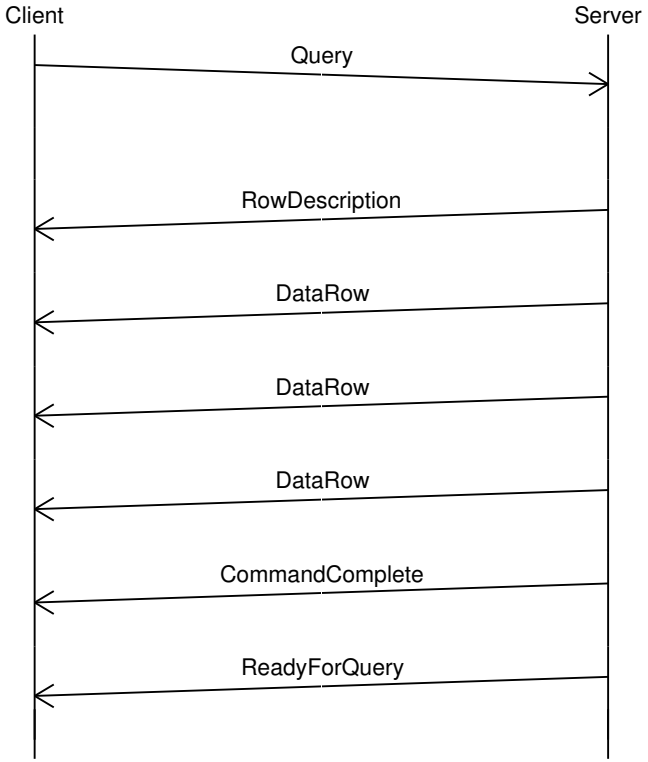
'D'	int32 len	int16 numfields	int32 fieldlen	char[fieldlen] data	...

CommandComplete

'C'	int32 len	str tag

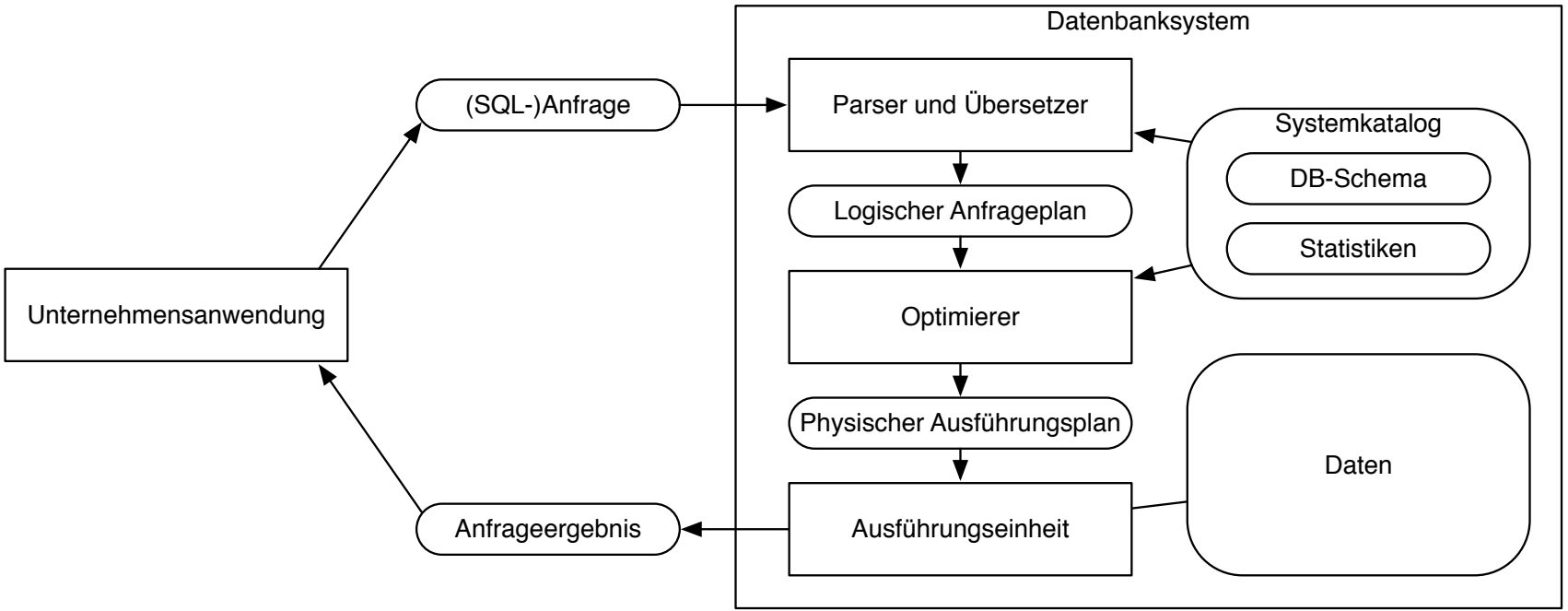
ReadyForQuery

'Z'	int32 len	'I' or 'T' or 'E'



J. Urbanski „Postgres on the wire“ https://wulczer.org/postgres-on-the-wire.pdf

Anfragebearbeitung
Von der SQL-Anfrage zum Ergebnis

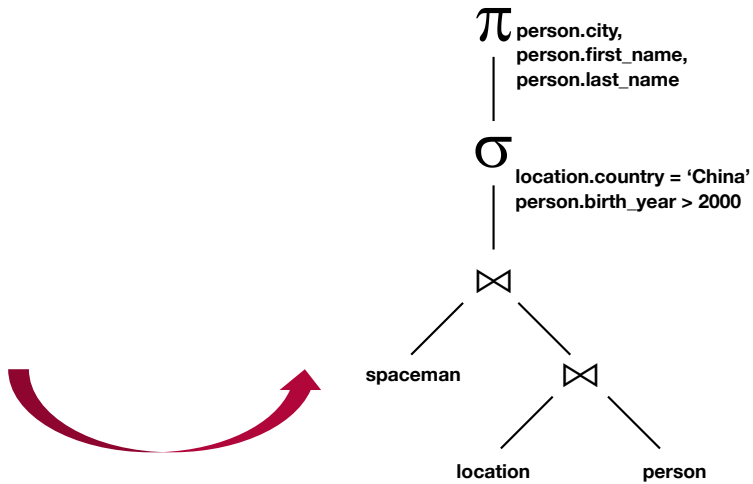


Anfragebearbeitung
Query-Parsing und Übersetzung

- Query-Parsing: Syntaxüberprüfung
- Übersetzung: inklusive Semantiküberprüfung anhand des Schemas

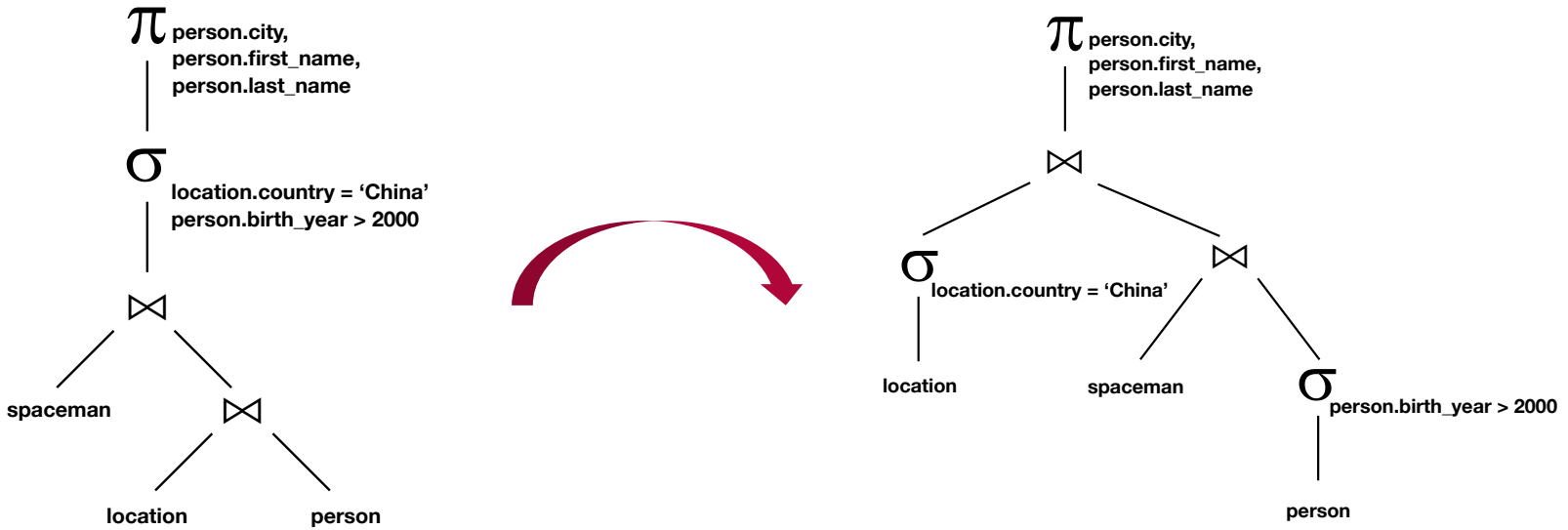
- Der SQL-Parser erstellt einen logischen Anfrageplan, das ist ein Ausdruck/Baum mit relationalen Operatoren

```
1  SELECT p.city , p.first_name, p.last_name
2  FROM person AS p
3  INNER JOIN location ON p.city = location.city
4  INNER JOIN spaceman ON spaceman.first_name = p.first_name
5  AND spaceman.last_name = p.last_name
6  WHERE location.country = 'China' AND p.birth_year > 2000
```



Anfragebearbeitung
Optimierung

- Abhängig von der Reihenfolge der Operationen kann die Ausführungszeit stark variieren
- Daher versucht das DBMS einen effizienten Ausführungsplan zu finden



Anfragebearbeitung
Optimierung

Often, the impact of the query optimizer is much larger than the impact of the runtime system [..] Changes to an already tuned runtime system might bring another 10% improvement, but changes to the query optimizer can often bring a factor 10.

T. Neumann. Engineering high-performance database engines. PVLDB, 2014

Anfragebearbeitung
Optimierung

- Für eine gegebene Anfrage (SQL ist deklarativ) gibt es **sehr viele logisch äquivalente Ausführungspläne** (um das gleiches Anfrageergebnis prozedural zu erzeugen)
- Der Query-Optimizer wählt einen Ausführungsplan mit niedrig geschätzten Ausführungskosten

Kostenarten

- Algorithmisch: z.B. Komplexität der Algorithmen für Operator-Implementierungen
- Logisch: geschätzte Ergebnisgröße des Operators
  (z.B. Abnahme für Selektion, Abnahme oder Zunahme für Join)
  ← unser Fokus
- Physisch: Hardware-abhängige Berechnungskosten (z.B. I/O-Bandbreite, Cache-Miss)

Anfragebearbeitung
Optimierung – Ansätze

- Heuristiken/Regeln
  - Umformulierungen um offensichtlich ineffiziente Ausführungspläne zu vermeiden
  - Benötigen kein Kostenmodell
- Kostenbasierte Ansätze
  - Nutzen Kostenmodelle um Ausführungspläne zu schätzen
  - Wählen Ausführungsplan mit den am niedrigsten geschätzten Ausführungskosten

- **Ziel: kleine Zwischenergebnisse**

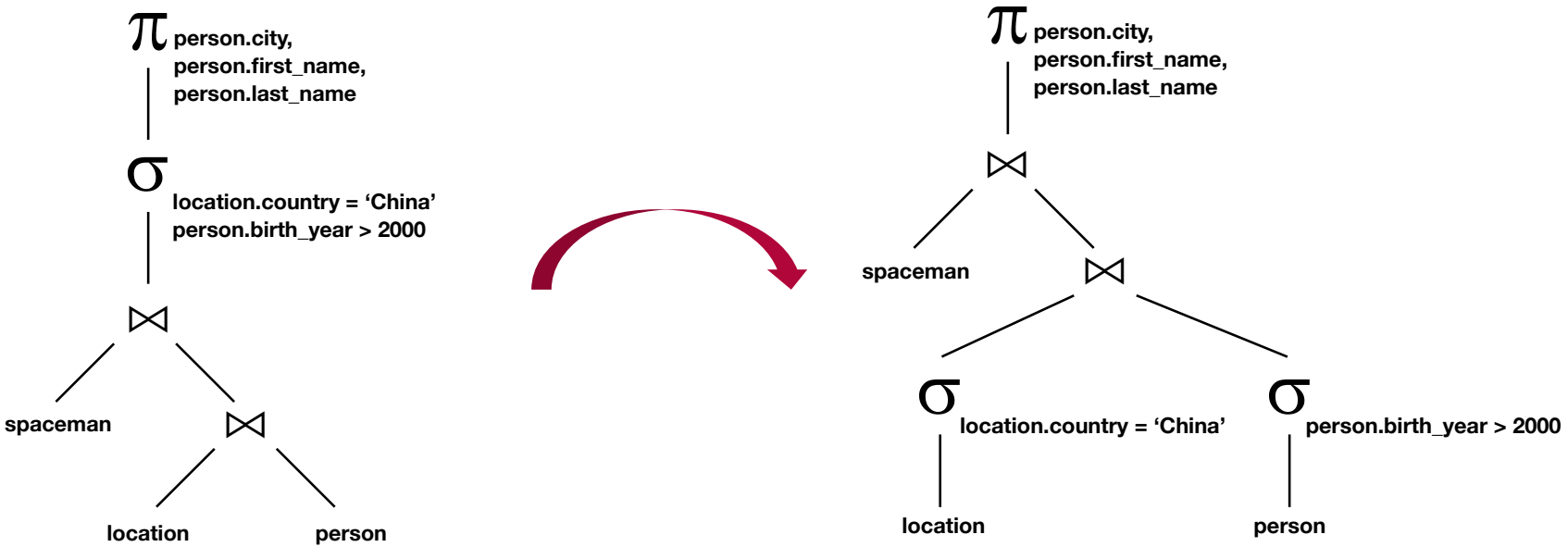
Anfragebearbeitung
Optimierung – Äquivalenzen der Relationalen Algebra

Optimierungen für Selektion

- Prädikat-Pushdown
  - Filter so früh wie möglich ausführen
  - Den selektivsten (der das (Zwischen-)Ergebnis am stärksten verkleinert) Filter zuerst ausführen

- Prädikate vereinfachen (auch unmögliche oder unnötige Prädikate)

Anfragebearbeitung
Optimierung – Prädikat-Pushdown



Anfragebearbeitung
Optimierung – Äquivalenzen der Relationalen Algebra

Hasso Plattner Institut

Optimierung für Projektion

- Projektion-Pushdown
  - Früh ausführen um kleinere/kürzere Tupel zu erzeugen
  - Idee: nur angefragte und benötigte (Join-Bedingung, Gruppierung, Sortierung) Attribute im Zwischenergebnis zu halten

(für spaltenbasierte Datenbanken nicht wichtig)

Anfragebearbeitung
Optimierung – Äquivalenzen der Relationalen Algebra

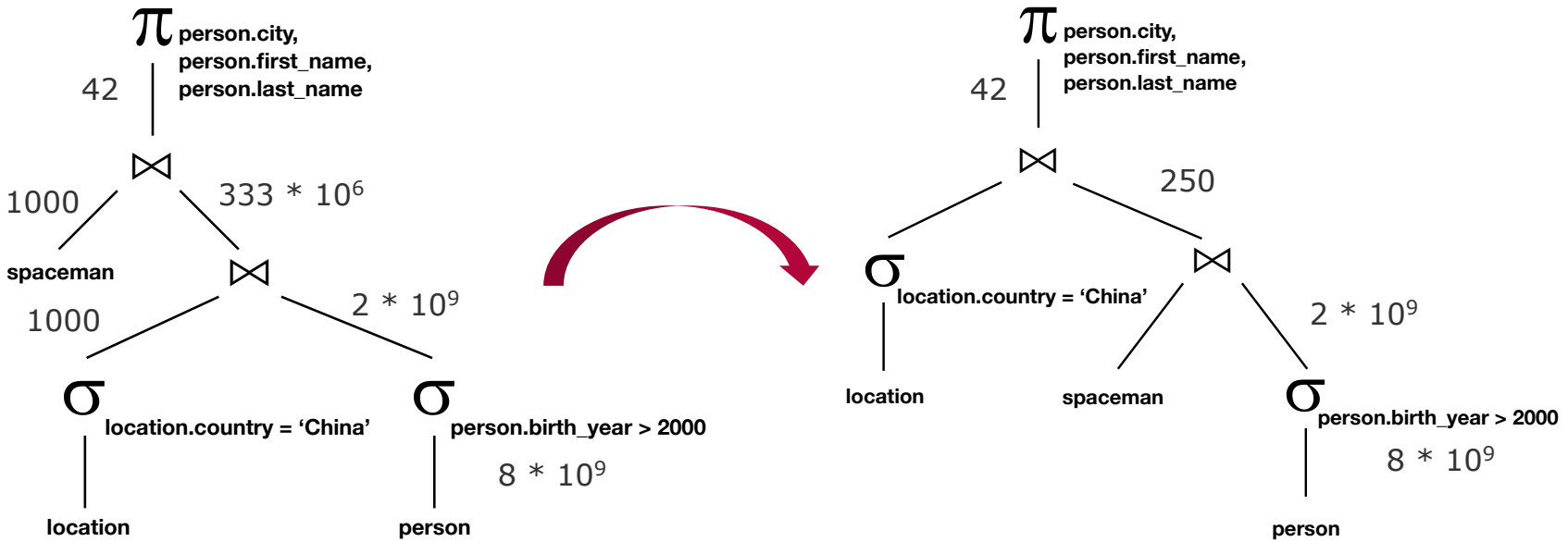
Optimierungen für Join

- Join-Reihenfolge mit niedrigsten Kosten (kleinsten Zwischenergebnissen) finden;
  möglich da Join **kommutativ** und **assoziativ** ist

$$R \bowtie S = S \bowtie R \qquad (R \bowtie S) \bowtie T = S \bowtie (R \bowtie T)$$

- Kreuzprodukt vermeiden

Anfragebearbeitung
Optimierung – Joinreihenfolge anpassen



Annahmen:

- Es gibt ca. 1000 Raumfahrer
- 1 / 6 der Menschen sind in China geboren
- Es gibt 1000 Orte in China (< 700 Städte, die sind aber sehr groß)
- 1 / 4 der Menschen sind nach 2000 geboren
- Alle Bedingungen sind unabhängig

Anfragebearbeitung
Optimierung – Kostenschätzung

- Wie groß ist das Ergebnis einer Selektion?
- Wie groß ist das Ergebnis eines Joins (Join-Kardinalität)?

Optimierer verwenden **Statistiken** (inklusive Constraints) und/oder **Sampling**

Anfragebearbeitung
Optimierung – Statistiken

Meta-Informationen über Tabellen:

- Tabellen-Kardinalität: Anzahl der Tupel
- Attribut-Kardinalität: Anzahl der verschiedenen Werte für ein festes Attribut
- Schlüssel und Fremdschlüssel

- Datenverteilung
  - uniform, exponentiell, ...
  - Minimum, Maximum
  - Top-n häufigsten Werte mit Häufigkeiten
  - Histogramm

- Informationen über Korrelationen (nicht in Histogrammen enthalten)

Example: https://postgrespro.com/docs/postgrespro/11/view-pg-stats

Anfragebearbeitung
Optimierung – Sampling

- Selektivitäten verknüpfter Prädikate und Join-Kardinalitäten mit Statistiken zu schätzen ist schwierig
- Datenbanken nutzen Sample von Tabellen um Ergebnisgrößen von Operatoren zu schätzen
- Sample – Tabelle mit zufällig gewählten Tupeln der Ursprungstabelle

Anfragebearbeitung
Optimierung – Ausführungsplan

Neben den Optimierung des logischen Ausführungsplans, müssen vor der Ausführung weitere Entscheidungen getroffen werden:

- Zugriff auf Tabellen: Index nutzen oder Tabelle scannen
- Implementierung der Operatoren

  z.B. Join-Algorithmus (Nested-Loop vs. Sort-Merge vs. Hash-basiert)
- Weitergabe von Zwischenergebnissen: Materialisieren oder Pipeline

Datenbanken: Lebenszyklus einer Query Zusammenfassung

- Datenbankanwendungen und -systeme kommunizieren über Nachrichten
- Die Anfragebearbeitung im Datenbanksystem besteht aus folgenden Schritten:
  - Deklarative Anfragen werden in logische Anfragepläne umgewandelt
  - Logische Anfragepläne werden in optimierte physische Ausführungspläne umgewandelt
  - Ausführungspläne werden ausgeführt