Tutorium 6.5
SQL, Deanonymisierung, Anfrageoptimierung
Big Data Engineering

Prof. Dr. Jens Dittrich

bigdata.uni-saarland.de

20./21. Juni 2022

Das heutige Modell

[Personen] : {[PID:int, Name:varchar, Wohnort: varchar, Geburtsjahr: int]}

[Sänger*innen] : {[SID:(Personen→PID), Künstler*innenname:varchar, Genre:varchar]}

[Musiklabels] : {[MID:int, Name:varchar, Kapital: int, Gründungsjahr: int]}

[Songs] : {[SongID:int, Label:(Musiklabels→MID), Titel: varchar,
Veröffentlichungsdatum: date]}

[singen_live] : {[Song:(Songs→SongID), Sänger*in:(Sänger*innen→SID), Datum:date,
Arena:string]}

Wiederholung - Frage 1

Frage

In welche Schritte kann Anfrageoptimierung unterteilt werden?

Wiederholung - Frage 1

Frage

In welche Schritte kann Anfrageoptimierung unterteilt werden?

Lösung

1. SQL
   ↓ kanonische Übersetzung
2. annotierte relationale Algebra/logischer Plan
   ↓ heuristische/regelbasierte Optimierung
3. transformierter logischer Plan
   ↓ kostenbasierte Optimierung
4. physischer Plan
   ↓ Code-Erzeugung
5. ausführbarer Code

Streng genommen ist die Code-Erzeugung **kein** Teil der Anfrageoptimierung, sondern ein eigenständiges Problem.

Wiederholung - Frage 2

Frage

Was ist ein Join-Graph?

Wiederholung - Frage 2

Frage

Was ist ein Join-Graph?

Lösung

Mit einem Joingraphen können wir die Joins zwischen verschiedenen Relationen in einer Anfrage visualisieren. Ein Joingraph hat dabei einen Knoten für jede Eingaberelation und eine Kante für jedes Joinpädikat. Zusätzlich werden Knoten, auf denen ein Filterprädikat existiert, mit diesem annotiert und alle Kanten mit dem entsprechenden Joinprädikat annotiert.
Der Joingraph kann bereits mit der ursprünglichen SQL-Anfrage korrekt erstellt werden kann. Eine vorherige Übersetzung in kanonisch annotierte relationale Algebra ist hier nicht notwendig.

Wiederholung - Frage 3

Frage

Welche Vor- und Nachteile ergeben sich, wenn Sie anstatt einer Übersetzung in kanonisch annotierte Algebra gleich einen Joingraphen erstellen und damit weiterarbeiten?

Wiederholung - Frage 3

Lösung

Vorteile:

- Die Übersetzung in relationale Algebra wird überflüssig. Hierdurch erspart man sich auch etwaige Probleme bei der Übersetzung, wie die Umbenennung oder der Umgang mit Aggregatfunktionen.
- Gewisse Teile der heuristischen Optimierung sind bereits automatisch im Join-Graphen vorhanden, wie zum Beispiel der Predicate Pushdown und das Zusammenfassen von Prädikaten und Kreuzprodukten zu einem Join.

Nachteile:

- Gewisse Operatoren können in einem Join-Graphen nicht dargestellt werden, z.B. Projektionen oder Gruppierungen.
- Aufgrund des Fehlens von Projektionen kann auch der Projection Pushdown nicht durchgeführt werden

Aufgabe 1.1

Frage

Enthält unten stehende SQL-Anfrage einen Fehler? Wenn ja, geben Sie diesen an und korrigieren Sie ihn. Wenn nein, erklären Sie umgangssprachlich, was die Anfrage ausgibt.

```
SELECT    Wohnort, COUNT(*)
FROM      Personen
GROUP BY  Name;
```

Aufgabe 1.1

Lösung

Diese SQL-Anfrage ist nicht korrekt, da sie im SELECT auf “Wohnort” zugreift, obwohl dieses Attribut kein Gruppierungsschlüssel im GROUP BY ist. Um dies zu korrigieren, müsste man im GROUP BY ebenfalls auf “Wohnort” zugreifen.

Aufgabe 1.2

Frage

Enthält unten stehende SQL-Anfrage einen Fehler? Wenn ja, geben Sie diesen an und korrigieren Sie ihn. Wenn nein, erklären Sie umgangssprachlich, was die Anfrage ausgibt.

```
SELECT    MAX(Gruendungsjahr)
FROM      Songs
          JOIN Musiklabels
               ON Label = MID
HAVING    MIN(Veroeffentlichungsdatum)
          < '2000-01-01'
WHERE     Kapital < 2000;
```

Aufgabe 1.2

Lösung

Diese SQL-Anfrage ist syntaktisch nicht korrekt, da das WHERE nach dem HAVING steht. Dies ist falsch, da WHERE eine Bedingung auf Tupeln darstellt, während HAVING eine Bedingung an Gruppen ist. Um dieses Problem zu beheben, muss man das WHERE-Statement mit dem HAVING-Statement tauschen.
Beachten Sie hierbei, dass das HAVING auch ohne GROUP BY benutzt werden darf. In diesem Fall bezieht es sich auf die Gruppe **aller** Tupel, die in obiger Relation nach dem WHERE noch enthalten sind.

Aufgabe 1.3

Frage

Enthält unten stehende SQL-Anfrage einen Fehler? Wenn ja, geben Sie diesen an und korrigieren Sie ihn. Wenn nein, erklären Sie umgangssprachlich, was die Anfrage ausgibt.

```
SELECT    Geburtsjahr
FROM      (SELECT *
           FROM   Personen
                  JOIN Saenger*innen
                       ON SID = PID
           WHERE  Genre = 'Rock'
          ) AS Rocksingers
GROUP BY PID
```

Aufgabe 1.3

Lösung

Diese SQL-Anfrage ist syntaktisch nicht korrekt, da im SELECT auf "Geburtsjahr" und im GROUP BY auf "PID" ohne den Tabellenprefix "Rocksingers" zugegriffen wird, was insofern problematisch ist, als dass es sich hierbei um eine Unteranfrage handelt und Tabellenprefixe hier zwangsweise benötigt werden. Um dieses Problem zu beheben, muss man daher vor beide Attribute das Tabellenprefix "Rocksingers" hinzufügen.

Aufgabe 2

Frage

Übersetzten Sie folgende Ausdrücke der relationalen Algebra in SQL-Anfragen ohne Unteranfragen.

1. $R_1 := (\sigma_{\text{Wohnort} = \text{'Berlin'}} \text{ Personen}) \bowtie_{\text{PID} = \text{Sänger*in}} \text{singen\_live}$
   $\gamma_{\text{Song,Datum,count(*)}} R_1$
2. $R_1 := (\sigma_{\text{Titel} \neq \text{'Schools Out'}} \text{ Songs}) \bowtie_{\text{Song} = \text{SongID}} \text{singen\_live}$
   $R_2 := \gamma_{\text{Datum, Sänger*in,max(Veröffentlichungsdatum)}}(\sigma_{\text{Arena} = \text{'Barclays'}} R_1)$
   $\pi_{\text{Sänger*in,Datum}}(\sigma_{\text{max(Veröffentlichungsdatum)} \geq 10.03.2004} R_2)$
3. $R_1 := \sigma_{\text{Kapital} > 3500 \wedge \text{Gründungsjahr} < 2000} \text{ Musiklabels}$
   $R_2 := \gamma_{\text{Label,count(*),min(Veröffentlichungsdatum)}}(\text{Songs} \bowtie_{\text{Label=MID}} R_1)$
   $\pi_{\text{Label, count(*)}}(\sigma_{\text{min(Veröffentlichungsdatum)} > 11.05.2000} R_2)$

Aufgabe 2.1

Lösung

```
SELECT    Song, Datum, COUNT(*)
FROM      Personen
          JOIN singen_live
               ON PID = Saenger*in
WHERE     Wohnort = 'Berlin'
GROUP BY  Song, Datum;
```

Aufgabe 2.2

Lösung

```
SELECT    Saenger*in, Datum
FROM      Songs
          JOIN singen_live
               ON Song = SongID
WHERE     Arena = 'Barclays'
          AND Titel <> 'Schools Out'
GROUP BY  Saenger*in, Datum
HAVING    MAX(Veroeffentlichungsdatum)
          >= '2004-03-10';
```

Aufgabe 2.3

Lösung

```
SELECT    Label, COUNT(*)
FROM      Musiklabels
          JOIN Songs
               ON Label = MID
WHERE     Kapital > 3500
          AND Gruendungsjahr < 2000
GROUP BY  Label
HAVING    MIN(Veroeffentlichungsdatum)
          > '2000-05-11';
```

Aufgabe 3

Frage (1/2)

Eine IT-Firma speichert intern Daten darüber, zu welchen Zeitpunkten ihre Mitarbeiter*innen arbeiten. Diese Zeiten werden von den Arbeitenden (die auch im Home-Office arbeiten können) selbst verwaltet:

[Mitarbeiter*innen] : {[MID:int, Name:string]}

[arbeiten] : {[Mitarbeiter*in:(Mitarbeiter*innen→ MID), Tag: date, Stunde: int]}

Durch einen kürzlich erschienenen Zeitungsartikel über ein gelöstes Verbrechen ist die Firma nun allerdings auf die Fähigkeiten von Timmy (10) aufmerksam geworden, mit dessen Hilfe sie Zugriff auf das Browserverhalten ihrer Mitarbeitenden bekommen haben:

[Websiten] : {[WID:int, Typ: string, Name: string]}

[besuchen] : {[Mitarbeiter*in:(Mitarbeiter*innen→ MID), Website:(Websites→WID), Zeitpunkt: timestamp]}

Aufgabe 3

Frage (2/2)

Wie kann die IT-Firma diese illegalerweise erworbenen Daten nutzen, um problematische Mitarbeitende zu identifizieren und entsprechend abzustrafen? Erläutern Sie.
Geben Sie zusätzlich eine (oder mehrere) umgangssprachliche Anfragen an, die diese entsprechenden Mitarbeiter*innen identifizieren.

Aufgabe 3

Lösung

Mit Hilfe der Browserdaten kann die IT-Firma überprüfen, ob ihre Mitarbeiter*innen während den angegebenen Zeiten tatsächlich gearbeitet haben oder etwa Websites besucht haben, die nichts mit ihrer Arbeit zu tun haben (z.B. Gamingwebsiten, Reisewebsites, etc.). Um nicht die illegalen Tätigkeiten zu offenbaren, könnte die IT-Firma anschließend die getätigte Arbeit als unzureichend betiteln und androhen, die Gehälter zu kürzen. Eine entsprechende Anfrage könnte folgendermaßen lauten:
Die MID der Mitarbeiter*innen, die schon einmal zu einer angegebenen Arbeitsstunde eine Website vom Typ Gaming oder Reisen besucht haben.
Alternativ könnte man auch zählen, wie oft dies schon geschehen ist und bei einmaligen Verstoßen nachsichtig sein:
Die MID der Mitarbeiter*innen und die Anzahl, wie oft diese bereits zu einer angegebenen Arbeitsstunde eine Website vom Typ Gaming oder Reisen besucht haben.

Aufgabe 4

Frage

Betrachten Sie folgende SQL Anfrage.

```
SELECT *
FROM A, B, C, D
WHERE A.b = B.a AND B.c = C.b AND C.d = D.c;
```

Gegeben seien die folgenden Tabellen- und Teilproblemgrößen:

$$|\{A\}| = 150 \quad |\{B\}| = 100 \quad |\{C\}| = 50 \quad |\{D\}| = 80$$

$$|\{A,B\}| = 300 \quad |\{B,C\}| = 150 \quad |\{C,D\}| = 40$$

$$|\{A,B,C\}| = 450 \quad |\{A,B,D\}| = 24.000 \quad |\{A,C,D\}| = 6.000 \quad |\{B,C,D\}| = 120$$

$$|\{A,B,C,D\}| = 360$$

Bestimmen sie die optimale Joinreihenfolge. Ihnen stehen folgende Kostenfunktionen zur Verfügung:

$$C_{\text{HashJoin}}(R \bowtie S) = |R| + |S|, \quad C(R \times S) = |R| \cdot |S|$$

Beachten Sie, dass sie einige Pläne von vornherein ausschließen können.

Aufgabe 4

Lösung

Da die Kostenfunktion symmetrisch ist, müssen wir für zwei Teilpläne A und B lediglich die von Kosten $A \bowtie B$ oder $B \bowtie A$ betrachten, aber nicht beide. Außerdem ignorieren wie Pläne, die Kreuzprodukte enthalten:

Teilplan	Kosten	Ergebnisgröße
A	0	150
B	0	100
C	0	50
D	0	80
$A \bowtie B$	$0 + 0 + 150 + 100 = 250$	300
$B \bowtie C$	$0 + 0 + 100 + 50 = 150$	150
$C \bowtie D$	$0 + 0 + 50 + 80 = 130$	40
$(A \bowtie B) \bowtie C$	$250 + 0 + 300 + 50 = 600$	450
$(B \bowtie C) \bowtie A$	$150 + 0 + 150 + 150 = 450$	450
$(B \bowtie C) \bowtie D$	$150 + 0 + 150 + 80 = 380$	120
$(C \bowtie D) \bowtie B$	$130 + 0 + 40 + 100 = 270$	120
$((B \bowtie C) \bowtie A) \bowtie D$	$450 + 0 + 450 + 80 = 980$	360
$((C \bowtie D) \bowtie B) \bowtie A$	$270 + 0 + 120 + 150 = 540$	360
$(A \bowtie B) \bowtie (C \bowtie D)$	$250 + 130 + 300 + 40 = 720$	360

Die optimale Joinreihenfolge ist $((C \bowtie D) \bowtie B) \bowtie A$.

Aufgabe 5

Frage

Geben Sie eine SQL-Anfrage an, die die Namen der Sänger*innen, sowie die Anzahl an verschiedenen Songs, die ein “t” im Titel tragen, die diese gesungen haben, ausgibt. Hierbei sollen lediglich Songs von Musiklabels betrachtet werden, deren ältester Song vor 2000 veröffentlicht wurde.
Sie dürfen sowohl Views als auch Unteranfragen benutzten.

Aufgabe 5

Lösung

```
CREATE VIEW labels AS
SELECT    Label
FROM      Songs
GROUP BY Label
HAVING    MIN(Veroffentlichungsdatum) < '2000-01-01'

SELECT    Name, COUNT(*)
FROM      (SELECT DISTINCT Saenger*in, Song
           FROM    labels JOIN Songs
                          ON Songs.Label = labels.Label
                          JOIN singen_live
                          ON Song = SongID
           WHERE   Titel LIKE '%t%'
          ) AS singers
          JOIN Personen ON PID = singers.Seanger*in
GROUP BY singers.Saenger*in, Name
```