Verteilte Informationssysteme WS 2019/20
Übungsblatt 4

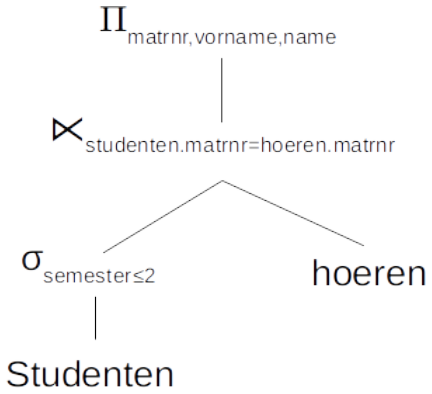
Aaron Winziers - 1176638

11. Dezember 2019

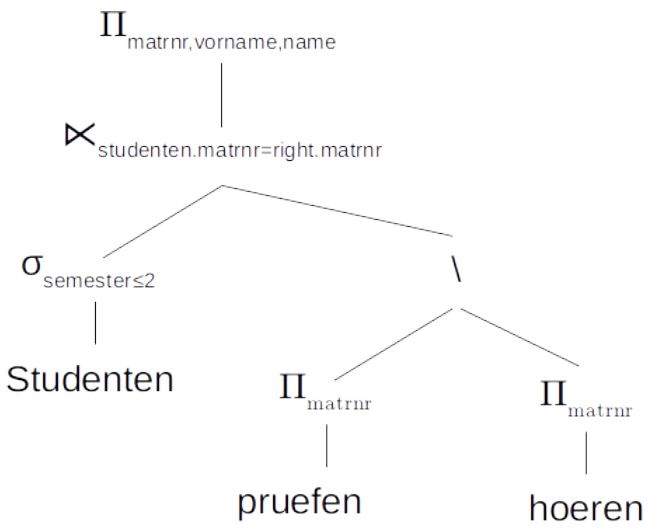
Aufgabe 1

a) Ein Semi-Join ist immer dann sinnvoll, wenn eine Beziehung zwischen den Reihen der beiden Tabellen vorhanden sein muss, aber die Informationen der einen TAbelle nicht benötigt werden. Wenn man eine Liste der Professoren die Vorlesungen halten haben will ohne wissen zu wollen was es für Veranstaltungen sind, könnte ein Semi-Join zwischen $professoren$ und $vorlesungen$ durchgeführt werden.

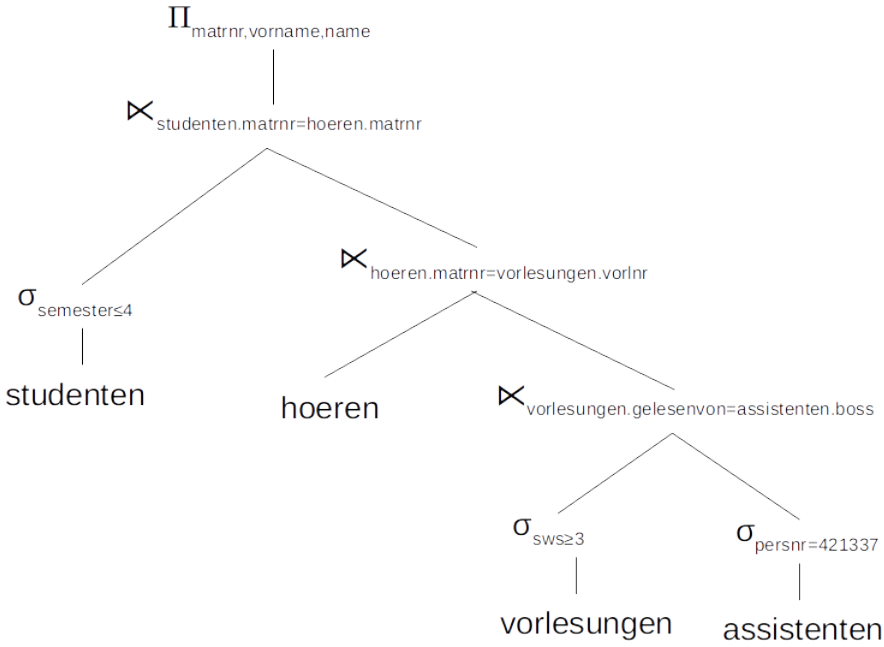
b) Die Matrikelnummer, Vornamen und Namen aller Studenten die im zweiten oder früheren Semester sind die eine oder mehrere Vorlesungen hören



c) Die Matrikelnummer, Vornamen und Namen aller Studenten die im zweiten oder früheren Semester sind und die an einer Prüfung teilnehmen für die sie die Vorlesung nicht gehört haben.



d) Die Matrikelnummer, Vornamen und Namen aller Studenten die im vierten oder früheren Semester sind, die eine Veranstaltung hören die 3 SWS oder mehr Aufwand haben und von dem Boss von dem Mitarbeiter mit Personalnummer 421337 gehalten werden.

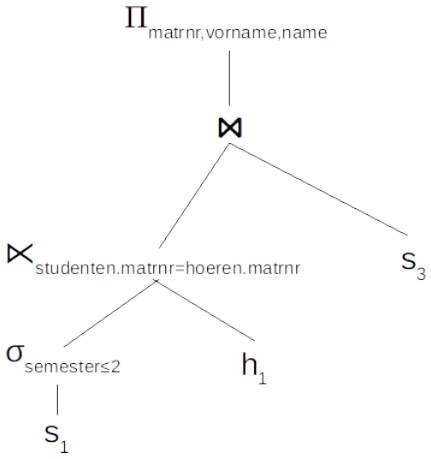


In allen Fällen wurden die Selektionen und Projektionen so weit runter wie möglich in den Abfragebäumen gezogen um die Anzahl der Tupel die von den Joins verarbeitet werden müssen zu minimieren. Kartesische Produkte wurden durch Semi Joins so oft wie möglich ersetzt, erstens damit die Anzahl der resultierende Tupel stark reduziert wurde, und um die Größe der Tupel zu minimieren(nicht benötigte Daten zu sparen).

Die unteren zwei Joins in Teilaufgabe d wurden mit Attribute ausgeführt die mit Hash Tabellen indexiert wurden was zu einer schnelleren Durchführung der Joins führen sollte. Da die Anzahl der Tupel in *hoeren* sehr hoch ist sollte dies zu besserer Performance führen. Der letzte Join in der Aufgabe erfolg nicht mehr mittels Hash Tabellen, aber da die Anzahl der Tupel die in die rechte Seite des Joins kommen schon stark reduziert wurde sollte dies kein Problem darstellen.

Aufgabe 2

a) Hier wurden Selektionen wieder so weit runter in den Baum gezogen wie möglich. Hier wurde der Semi Join ausgeführt vor dem Natural Join, da die Partitionen s_1 und h_1 auf dem gleichen Knoten liegen, und somit weniger Daten zwischen den Knoten versendet werden mussten.



b) Da die Menge von Tupel die in *hoeren* enthalten sind viel größer ist als die von *vorlesungen*, wurden die Partitionen von *vorlesungen* redundant gejoined auf beiden Knoten die h_1 und h_2 enthalten. Dies sollte zu geringeren Kommunikationskosten führen, da nur eine reduzierte Menge an von *hoeren* kommuniziert werden müsen.