

CS 386/586 ASSIGNMENT 7
FALL 2011

Question 1: Relational Algebra Equivalences

(10 points each) For each of the proposed equivalences below on relations r and s , give example instances where the equivalence does not hold. (Assume the expressions are syntactically valid.) Also, for each proposed equivalence, give an additional condition that will guarantee that the equivalence holds. The condition should be at the schema level, and the more general the condition, the better. X is a set of attributes and C is a single attribute. All joins are natural joins. $COUNT()$ is a function that returns the number of tuples in a relation.

a. $\pi_X(r \bowtie s) \equiv \pi_X(r) \bowtie s$

Suppose that relation r contains three attributes $\{A, B, C\}$ as set R , relation s contains three attributes $\{B, C, D\}$ as set S , and $X = \{A, B\}$. Then the attributes projected from $\pi_X(r \bowtie s)$ would be $\{A, B\}$ while the attributes projected from $\pi_X(r) \bowtie s$ would be $\{A, B, C, D\}$. Thus this equivalence doesn't hold. The equivalence holds if we have an additional condition: $R \subseteq X \subseteq S$.

b. $\pi_X(r - (\sigma_{C=c}(r))) \equiv \pi_X(r) - \pi_X(\sigma_{C=c}(r))$

Suppose we have an example instance for relation r contains three attributes $\{A, B, C\}$, $X = \{B\}$ and $C = c$.

r	(A	B	C)
		a	e	b	
		b	e	c	
		a	f	d	

Then we can get: $\pi_X(r - (\sigma_{C=c}(r))) = \begin{matrix} e \\ f \end{matrix}$ $\pi_X(r) - \pi_X(\sigma_{C=c}(r)) = f$

Thus this equivalence doesn't hold. The equivalence holds if we add an additional condition that X contains a key for relation r .

c. $\text{COUNT}(r \bowtie s) \equiv \text{COUNT}(r) * \text{COUNT}(s)$

Suppose we have example instances for relation r contains three attributes $\{A, B, C\}$, and relation s contains three attributes $\{B, C, D\}$ as shown below.

r	(<u>A</u>	<u>B</u>	<u>C</u>)
		a	b	c	
		a	b	d	

s	(<u>B</u>	<u>C</u>	<u>D</u>)
		b	c	d	
		b	e	f	

Then we can get $\text{COUNT}(r \bowtie s) = 1$ and $\text{COUNT}(r) * \text{COUNT}(s) = 4$.

Thus this equivalence doesn't hold. The equivalence holds if relations r and s have no attributes in common so the join is a cross product in this case.

d. $\text{COUNT}(r \bowtie s) \equiv \text{COUNT}(r)$

Suppose we have example instances for relation r contains two attributes $\{A, B\}$, and relation s contains two attributes $\{B, C\}$ as shown below.

r	(<u>A</u>	<u>B</u>)
		a	b	

s	(<u>B</u>	<u>C</u>)
		b	c1	
		b	c2	

Then we can get $\text{COUNT}(r \bowtie s) = 2$ and $\text{COUNT}(r) = 1$.

Thus this equivalence doesn't hold. The equivalence holds if relations r contains a foreign key to relation s .

Question 2: Statistics

a. (5 points) Determine the min and max salary values in the agent table, and the number of rows in that table.

```
SELECT MIN (salary), MAX (salary), COUNT (salary)
FROM Agent
```

min	max	count
50008	366962	662

b. (5 points) Give an estimate for the number of rows in agent with salary < 75000, assuming a uniform distribution between min and max salary. Explain how you derived your estimate.

We can get the increment amount on salary of each row by $(max\ salary - min\ salary) / rows = (366962 - 50008) / 662 \approx 479$.

Since we assume a uniform distribution between min and max salary, for agents with salary less than 75000 we can get the estimate number of agents by $(75000 - min\ salary) / increment\ amount = (75000 - 50008) / 479 \approx 52$

c. (5 points) Find the 25th, 50th and 75th percentile values for salaries in the agent table. (The 50th percentile value, for instance, is the smallest number s such that 50% of the rows have salary value less than or equal to s.)

The number of rows for the three given percentile values are:

$$622 * 0.25 = 166$$

$$622 * 0.5 = 331$$

$$622 * 0.75 = 497$$

Then we can get the value of salary with the following query by change the value in the LIMIT clause. Take row number = 166 for example.

```
SELECT MAX (Temp.salary)
FROM (SELECT salary
      FROM Agent
      ORDER BY salary
      LIMIT 166) AS Temp
```

max

54802

The 25th percentile value for salaries is 54802.

Similarly, we can get the following results.

The 50th percentile value for salaries is 58430.

The 75th percentile value for salaries is 89643.

d. (5 points) Give an estimate of the number of rows in agent with salary < 75000 , assuming in a uniform distribution in each quartile determined in c. Explain how you derived your estimate.

The value of salary 75000 is between the salary range 58430 and 89643 as we got in part(c). We assume in a uniform distribution in each quartile determined in part(c), then we can get the estimate of the number of agents with salary less than 75000 by $(75000 - 58430) / (89643 - 58430) * (497 - 331) + 331 = 419$

e. (5 points) How many rows in agent have salary < 75000 ?

```
SELECT COUNT (*)
FROM Agent
WHERE salary < 75000
```

427 rows

Question 3: Selectivity Estimation

a. (5 points) How many distinct cities in the agent table?

```
SELECT COUNT (DISTINCT city)
FROM Agent
```

46 rows

b. (5 points) Give an estimate for the number of rows in agent that satisfy the predicate $city = C$ (assuming city C exists).

We assume the records are uniformly distributed, then there are $662 / 46 \approx 14$ agents satisfy the predicate $city = C$.

c. (5 points) How many distinct countries in the agent table?

```
SELECT COUNT (DISTINCT country)
FROM Agent
```

22 rows

d. (5 points) Give an estimate for the number of rows in agent that satisfy the predicate country = N (assuming country M exists).

We assume the records are uniformly distributed, then there are $662 / 22 \approx 30$ agents satisfy the predicate country = N.

e. (5 points) If city and country are uncorrelated, then one estimate for the number of rows that satisfy the predicate city = C AND country = N is #rows/((# distinct cities)*(# distinct countries)). Calculate this value.

The estimate value = $662 / (46 * 22) = 0.654$

f. (10 points) How many different city-country pairs actually appear in agent? What does that number suggest about the independence of city and country? Propose a better formula to estimate the number of rows that satisfy city = C AND country = N.

```
SELECT DISTINCT city, country
FROM Agent
```

There are 47 rows retrieved from above query which means there is a city appears twice for two different countries. It suggests that the city and country are not independent.

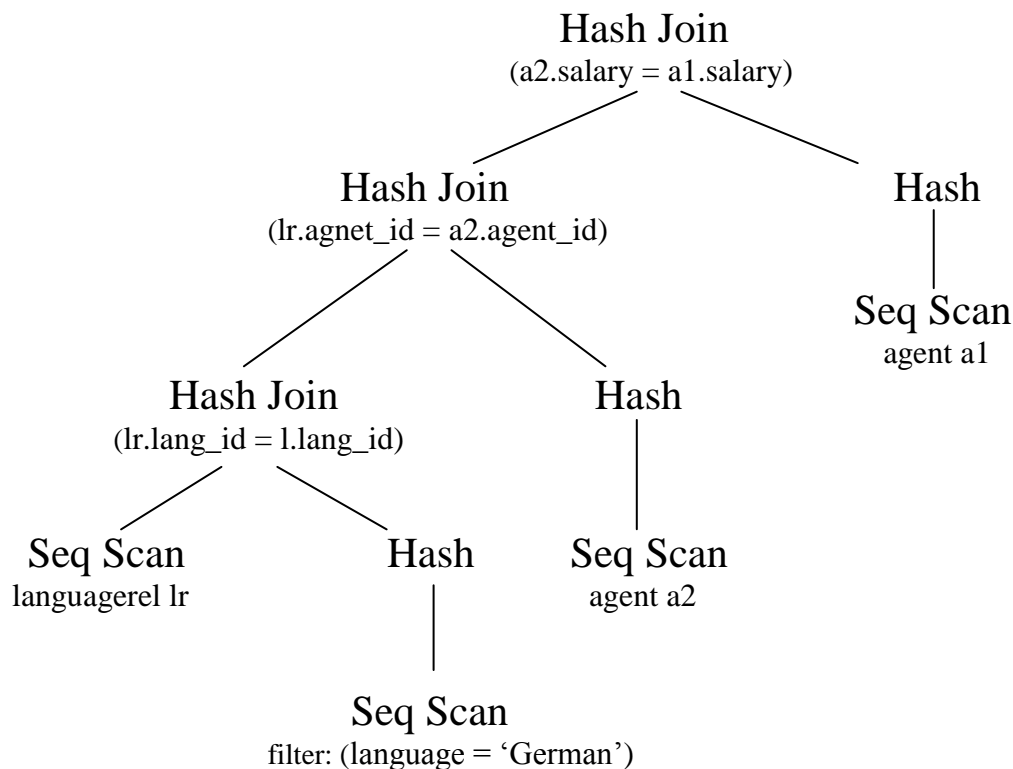
A better formula to estimate the number of rows which satisfy city = C and country = N could be based on the number of distinct city and country pair: # rows / # (DISTINCT city, country). And the estimated number of agents for a City-Country pair is $662 / 47 = 14$.

Question 4: Query Plans

For each SQL statement below, draw the query plan the Postgres produces (which you can obtain with the EXPLAIN command). For each plan, suggest a reason that the particular join algorithm(s) is being chosen.

a. (10 points) `SELECT A1.last, A2.last FROM agent A1, agent A2, languagerel LR, Language L WHERE A1.salary = A2.salary AND A2.agent_ID = LR.agent_ID AND LR.lang_ID = L.lang_ID AND L.language = 'German'`

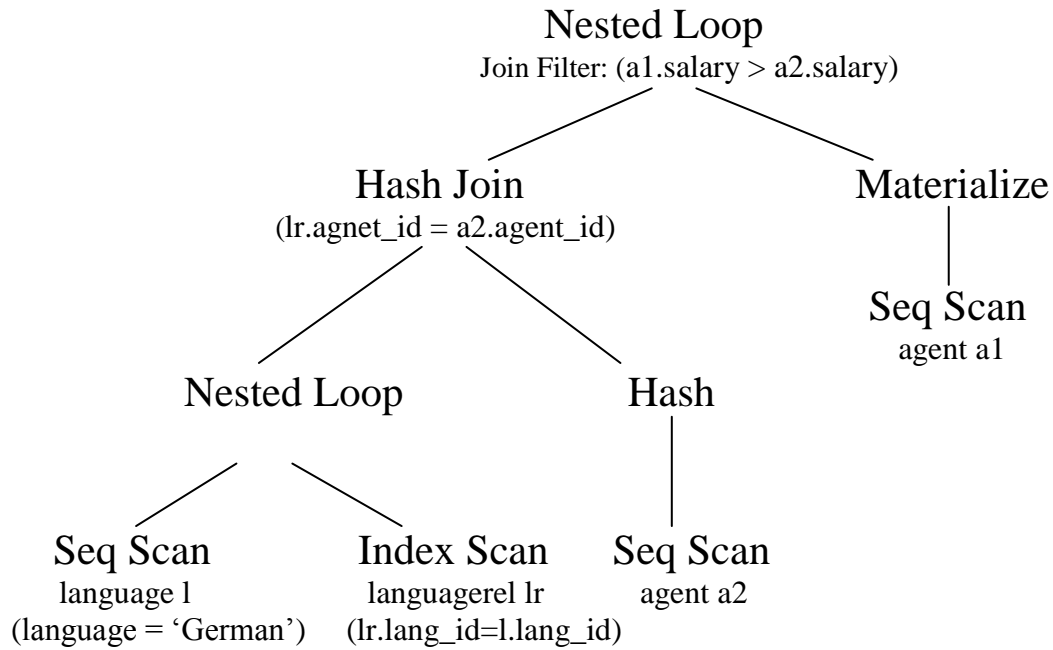
The query plan can be represented as following tree structure.



Hash joins are used by the optimizer in this query plan since all the joins are equi-joins. We can observe that the optimizer is putting the smallest inputs as the hash inputs to hash join. So the first hash join is done with LanguageRel and Language on lang_id. And it also does the first hash on the estimated smallest input on Language where language = 'German' returns the smallest rows (7 rows) of sequential scan which would reduce the number of intermediate rows flowing up the plan. Then the second hash join and the third hash join are done with the second estimated smallest input (662 rows) on Agent. So the optimizer can avoid hashing join with large tables which would produce much more cost.

b. (10 points) `SELECT A1.last, A2.last FROM agent A1, agent A2, languagerel LR, Language L WHERE A1.salary > A2.salary AND A2.agent_ID = LR.agent_ID AND LR.lang_ID = L.lang_ID AND L.language = 'German'`

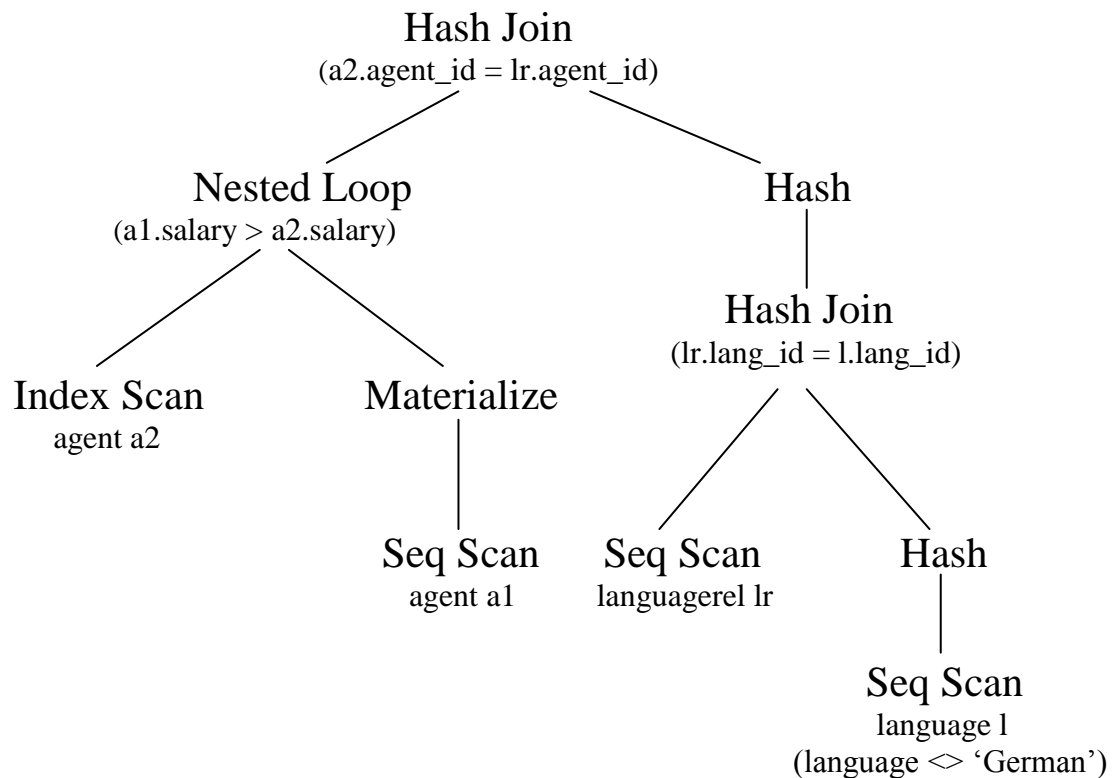
The query plan can be represented as following tree structure.



Nested loop join is used in this query at the highest level since it is the general join works with inequality conditions. The first nested loop join can be done quickly by using the index on LanguageRel (100 rows). By using index scan on languagerel, all needed records can be selected very fast. And it would be efficient to do the nested loop join with the 7 selected rows from the sequential scan on language = 'German'. Then a hash join on Agent is done next since it is more efficient for an equi-join (lr.agnet_id = a2.agnet_id). And the nested loop join at highest level is done on a an equality condition (a1.salary > a2.salary).

c. (10 points) `SELECT A1.last, A2.last FROM agent A1, agent A2, languagerel LR, Language L WHERE A1.salary > A2.salary AND A2.agent_ID = LR.agent_ID AND LR.lang_ID = L.lang_ID AND L.language <> 'German'`

The query plan can be represented as following tree structure.



In this query plan, the optimizer does the nested loop join on an equality condition (`a1.salary > a2.salary`) which would be more efficient here. The optimizer picks the result from the hash join on `lang_id` (1991 rows) rather than the result from nested loop join (146081 rows) for hashing since the cost on join of estimated number of rows on `lang_id` is much cheaper. And for the hash join with `LanguageRel` and `Language`, `Language` is used to be hashed here since it is smaller (1393 rows) than the `LanguageRel` (1991 rows).

