CS 486/586 Final

Fall 2021

Instructions.

This file contains the Final Exam for CS 486-586 - Fall 2021.

To complete this exam, please make a copy of the google doc and edit it. When you are ready to turn in your exam, please save the doc as a pdf and turn it in on Canvas. (If you wish to use a different editing software, that is fine, what is required is that you turn in a nice, legible pdf.)

The midterm is **due Thursday 12/9 12:05 pm.**

This exam is **open book, open notes, open Internet, open everything**. However, if you use material from the Internet or sources other than the textbook, course slides or your notes, **you must cite the web site or material that you used.** For example, you could include a link to the web site you used or the name of a book you used in your answer.

200 points total.

Please do be sure to read questions completely.

Breakfast Schema with Catalog Tables:

This exam uses the Breakfast Schema, which was selected as you are somewhat familiar with it. I have added two catalog tables called relation_info and attribute_info, which provide information about relations and attributes. These tables are loosely based on the pg_class table and pg_stats table.

The underlined attributes are primary keys. There are indices on primary keys and on Companies.StockSymbol.

Breakfast Schema:

BreakfastFoods

Id	Name	Calories	Fat	Sodium	Carbs	MadeBy
1	Bacon, Sausage & Egg Wrap	640	33	1090	58	5
2	Scrambled Eggs (2 eggs)	149	11	145	1.6	--
3	Peanut Butter Homestyle Granola	140	7	65	18	3
4	Nonfat Greek Yogurt	90	0	65	5	4
5	Coffee with half and half	42	3.5	19	1.5	5

Consumers

Id	Name	FavoriteBreakfast
1	Olivia	3
2	Roman	--
3	Mikhail	4
4	Daniela	5
5	Miranda	5
6	Kiran	4

Companies

Id	Name	StockSymbol
1	Hostess Brands, Inc	TWNK
2	J.M. Smucker Company	SJM
3	Bob's Red Mill	--
4	Fage	--
5	Starbucks	SBUX

Notes:

- Consumers are people who eat breakfast.
- Calories, Fat, Sodium and Carbs are per serving.
- Bob's Red Mill and Fage are privately held and so do not have a stock symbol.
- Scrambled eggs are homemade, so 'Made by' is null

Sources:

- https://www.starbucks.com
- https://www.bobsredmill.com/
- https://fdc.nal.usda.gov
- https://usa.fage

relation_info

relationname	numtuples	numpages	tuplesperpage
BreakfastFoods	6,000	600	10
Consumers	100,000	5,000	20
Companies	80	4	20

relationname -- the name of the relation
numtuples -- is the number of tuples in the relation
numpages -- is the number of pages in the relation
tuplesperpage -- the average number of tuples on each page

attribute_info

relationname	attrname	minvalue	maxvalue	distinctvalues
BreakfastFoods	Id	1	200	200
BreakfastFoods	Name	--	--	200
BreakfastFoods	Calories	5	500	180
BreakfastFoods	Fat	0	15	16
BreakfastFoods	Sodium	0	1500	1000
BreakfastFoods	Carbs	0	40	30
BreakfastFoods	MadeBy	1	50	45
Consumers	Id	1	100,000	100,000
Consumers	Name	--	--	98,000
Consumers	FavoriteBreakfast	1	200	175
Companies	Id	1	40	40
Companies	Name	--	--	40
Companies	StockSymbol	--	--	40

relationname -- the name of the relation
attrname -- the name of the attribute
minvalue -- the smallest value that occurs in the table relationname for attribute attrname. For example, the smallest number of Calories in any breakfast food in the BreakfastFood table is 5.
maxvalue -- the same as minvalue, but maximum value instead of minimum value
distinctvalues -- the number of distinct values in that attribute. For example, there are 200 different breakfast food Names in the BreakfastFood table.

Note: min and max values are not provided for string attributes

Name: Ajay Babu Gorantla

Q1 Index & Index Matching (15 points).

Below is a list of queries and a list of indexes. For each query, complete the table below to list the index or indexes that could be used to improve each query's performance and a brief justification — a few words or a short phrase — for choosing or not choosing an index. Consider both if the index matches the query and if the index will improve the query's performance.

Queries:

1.
```
SELECT *
FROM BreakfastFoods
WHERE Id = 5;
```

2.
```
SELECT *
FROM BreakfastFoods
WHERE Sodium <= 5 and Calories > 30;
```

3.
```
SELECT *
FROM BreakfastFoods B, Consumers C
WHERE B.Id = C.FavoriteBreakfast
      AND B.Calories = 130;
```

Indexes:

A. clustered index on BreakfastFoods.Id
B. unclustered index on BreakfastFoods.Calories
C. unclustered index on BreakfastFoods.Sodium
D. clustered index on Consumers.FavoriteBreakfast

Query	Indexes	Brief Justification
1 (5 points)	A	WHERE clause uses only 'Id' attribute
2 (5 points)	B, C	WHERE clause uses more than 1 attribute ('Sodium' and 'Calories' attributes)
3 (5 points)	A,B,D	First condition in WHERE clause uses 'Id' and 'FavoriteBreakfast' attributes and clustered indexes are available on those two attributes

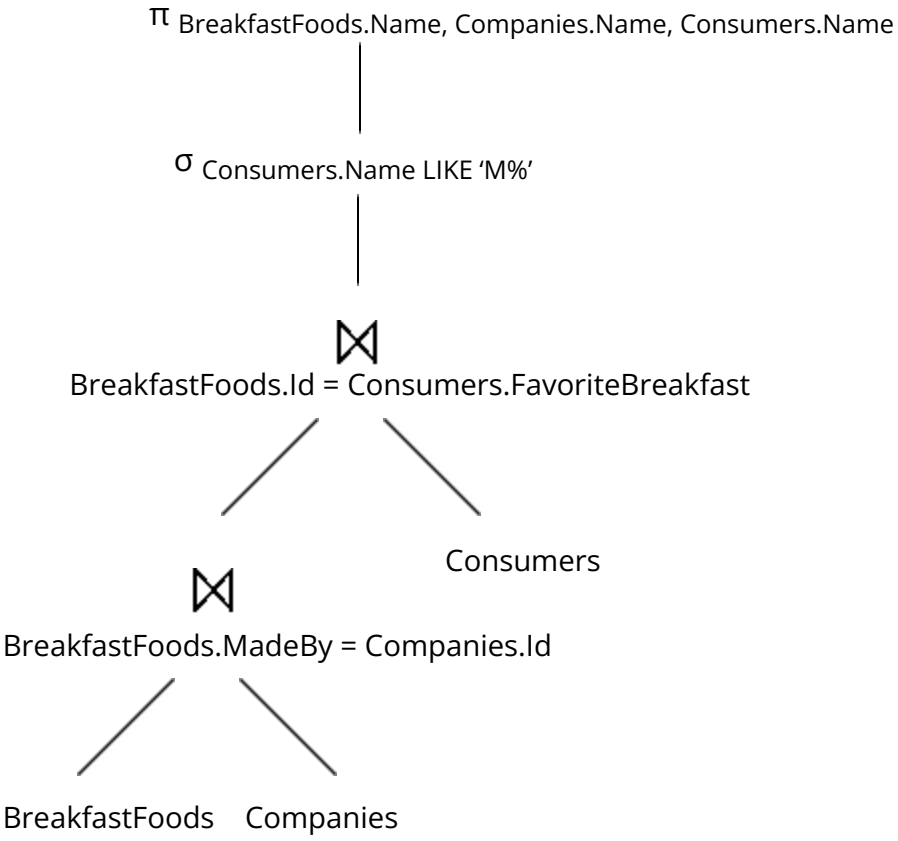
Q2 RA Equivalences (20 points).

Consider the Relational Algebra tree (a.k.a. logical query tree) below. Transform this tree twice - that is: transform the RA tree below into a new RA tree and then transform the new RA tree into a third RA tree.

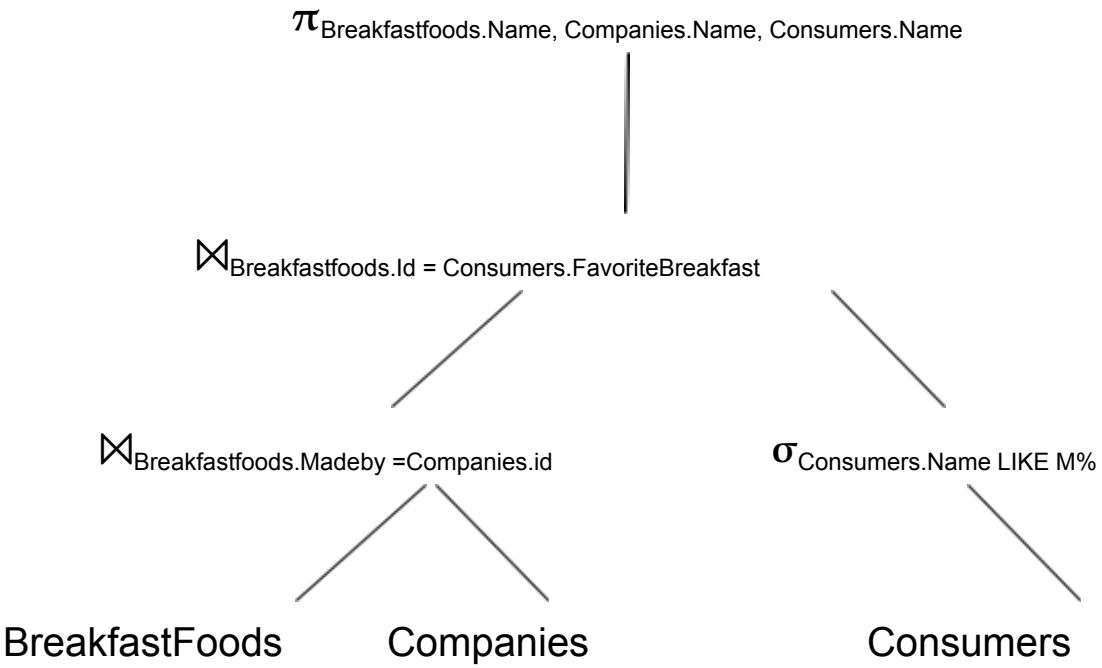
- Use the two RA Equivalences below in your transformations.
- Use the RA equivalences in the order given, that is the first transformation should be done with RA 1 and the second transformation done with RA 2.

RA 1: $\sigma_c(R \bowtie S) \equiv R \bowtie_c (S)$

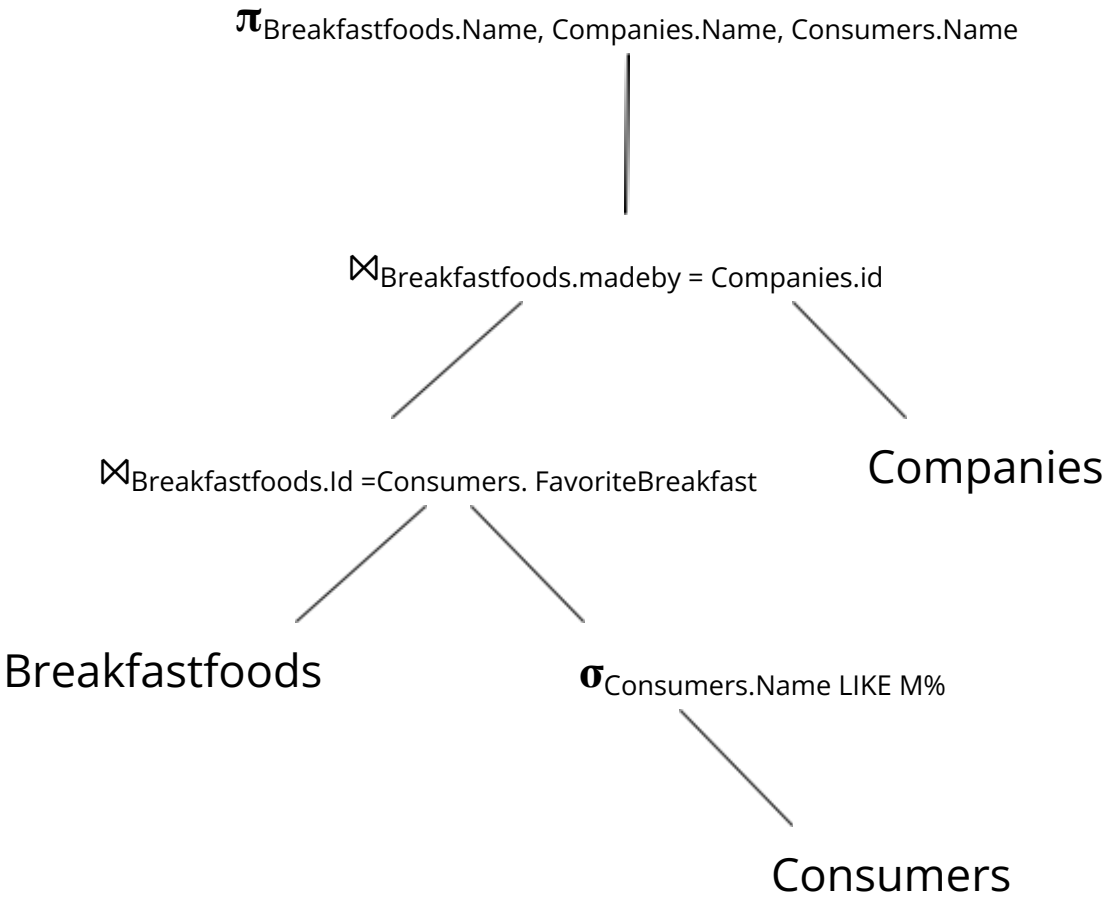
RA 2: $R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$.



RA1


πBreakfastfoods.Name, Companies.Name, Consumers.Name
⋈Breakfastfoods.Id = Consumers.FavoriteBreakfast
⋈Breakfastfoods.Madeby =Companies.id
σConsumers.Name LIKE M%
BreakfastFoods
Companies
Consumers

RA2


πBreakfastfoods.Name, Companies.Name, Consumers.Name
⋈Breakfastfoods.madeby = Companies.id
⋈Breakfastfoods.Id =Consumers. FavoriteBreakfast
Companies
Breakfastfoods
σConsumers.Name LIKE M%
Consumers

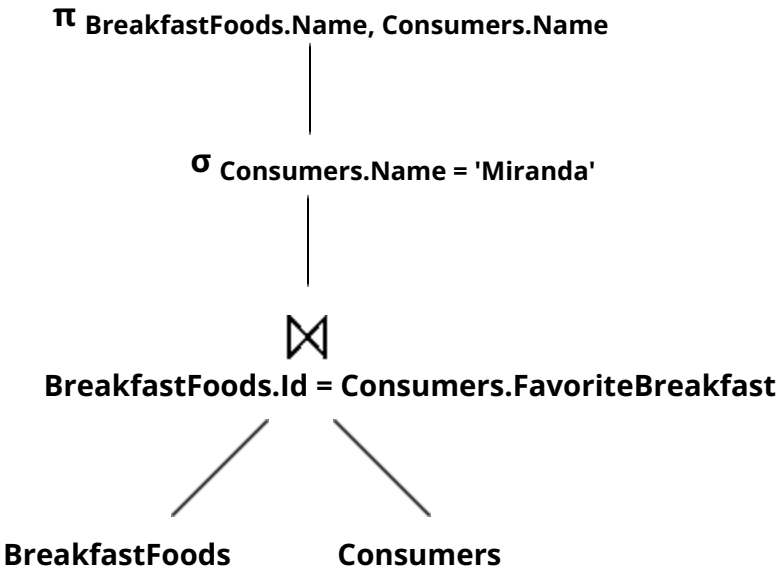
Q3 Cost Estimation (25 points).
For the two RA trees below, estimate the join cost assuming the query optimizer chooses a hash join.

For each RA tree, please:

- **give the estimated cost of the join**
- **state which relation on which you would build the hash table**
- **for b., describe how you estimated the size of the right input to the join**

Assume that the number of buffer pages available to the join is 75; that is BP = 75 (BP is the same as B). Please use the numbers in the relation_info table for your calculations.

a.



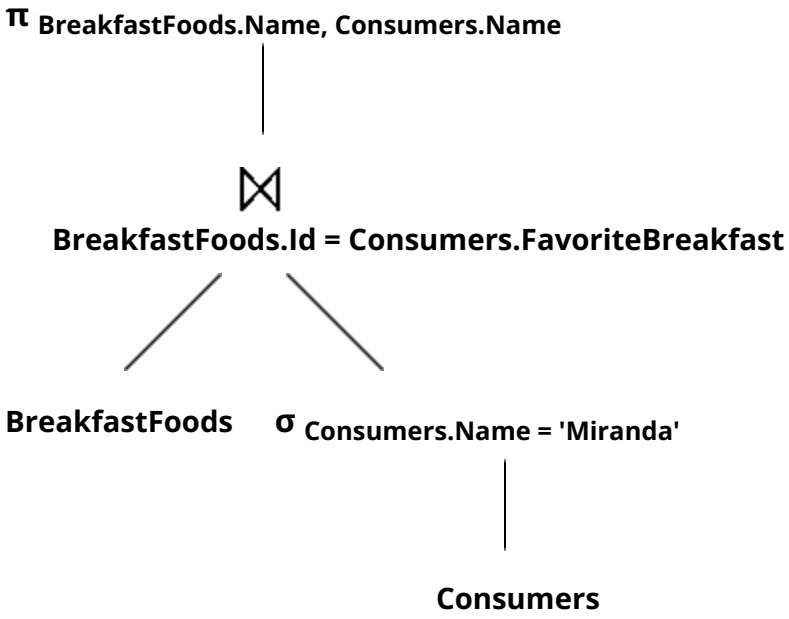
Let **M** be the number of pages in 'BreakfastFoods' Relation
Let **N** be the number of pages in 'Consumers' Relation

As the Query optimizer chooses a Hash Join, the Estimated cost of the join would be:

3(M+N) = 3(600+5000) =16800

So, the estimated cost of the Hash Join is **16800**.
I would build a hash table on 'BreakfastFoods'

b.



Let **M** be the number of pages in 'BreakfastFoods' Relation
Let **N** be the number of pages in 'Consumers' Relation
Let $\mathbf{N_f}$ be the number of pages after SELECT query

The estimated cost of the join would be $\mathbf{M+N_f}$

$\mathbf{N_f}$ would be equal to '1' because the total number of consumers and distinct consumer.id values are equal for consumer.name = 'Miranda' it would take 1 page.

So, the estimated cost would be:

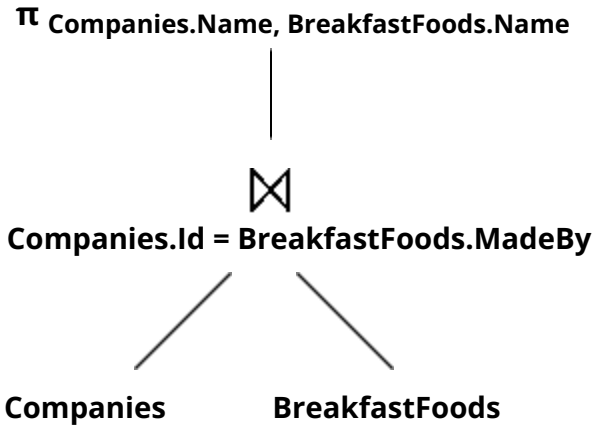
$\mathbf{M+N_f}$ = 600 +1 = 601

So, the estimated cost of the Hash Join is **601**.

I would build a hash table on 'BreakfastFoods'

Q4 Join Implementation (30 points).

Given the RA tree below and the hash function h(x) = x mod 4.



a. **Please draw the hash table that would be created for this join using the tuples given in Breakfast Schema.**

Hash Table built on Companies	
Hash Val	Tuple
0	(4,Fage,--)
1	(1,Hostess Brands Inc,TWNK), (5,Starbucks,SBUX)
2	(2,J.M. Smucker Company,SJM)
3	(3,Bob's Red Mill,--)

hash function h(x) = x mod 4.
Companies.Id 1 => 1 mod 4 = 1
Companies.Id 2 => 2 mod 4 = 2
Companies.Id 3 => 3 mod 4 = 3
Companies.Id 4 => 4 mod 4 = 0
Companies.Id 5 => 5 mod 4 = 1

b. Using the first tuple from the relation on which you did not build the hash table, please give a 3-step example of probing the hash table using that tuple.

I haven't built a hash table on Breakfastfoods

So, Using the first tuple from Breakfastoods,

Tuple:

(1,Bacon, Sausage & Egg Wrap,640,33,1090,58,5)

Step 1: First, fetch join attribute from Breakfastfoods i.e Breakfastfoods.MadeBy which is '5' in this case.

Step 2: Now, calculate the hash value using the given hash function

h(x)= x mod 4

= 5 mod 4

= 1

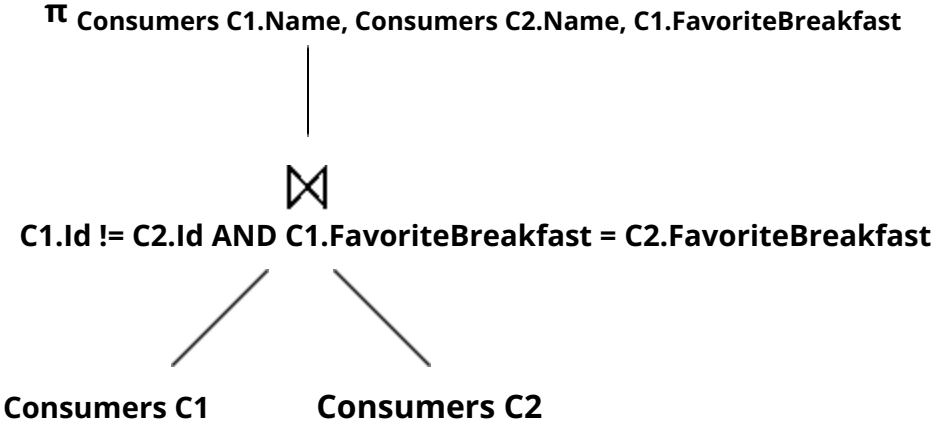
Step 3: Then, probe the Hash table using the hash value

Hash val	tuple
1	(1,Hostess Brands Inc,TWNK), (5,Starbucks,SBUX)

Q5 Query optimization - join selection (30 points)

For the queries below, estimate the join costs. Assume BP = 75 (BP is same as B). Please use the numbers in the relation_info table for your calculations.

a. **Estimate the cost of this query using:**
   i. **HashJoin**
   ii. **Sort-Merge Join**
   iii. **BlockNestedLoop Join**



Let **M** be the number of pages in 'Consumers' C1 Relation
Let **N** be the number of pages in 'Consumers' C2 Relation

(i) Cost = **3*(M+N)**

Because, neither of them would fit in the memory(BP),
however Square root of M and Square root of N are both less than BP

So, cost = 3*(5000+5000)
= 3*(10000)
= 30000

So, the cost for Hash Join is **30000**

(ii) Cost = **3*(M+N)**

Because, neither of the relations would be fit in memory(BP)

So, cost = 3*(5000+5000)
= 3*(10000)
= 30000

So, the cost for Sort-Merge Join would be **30000**

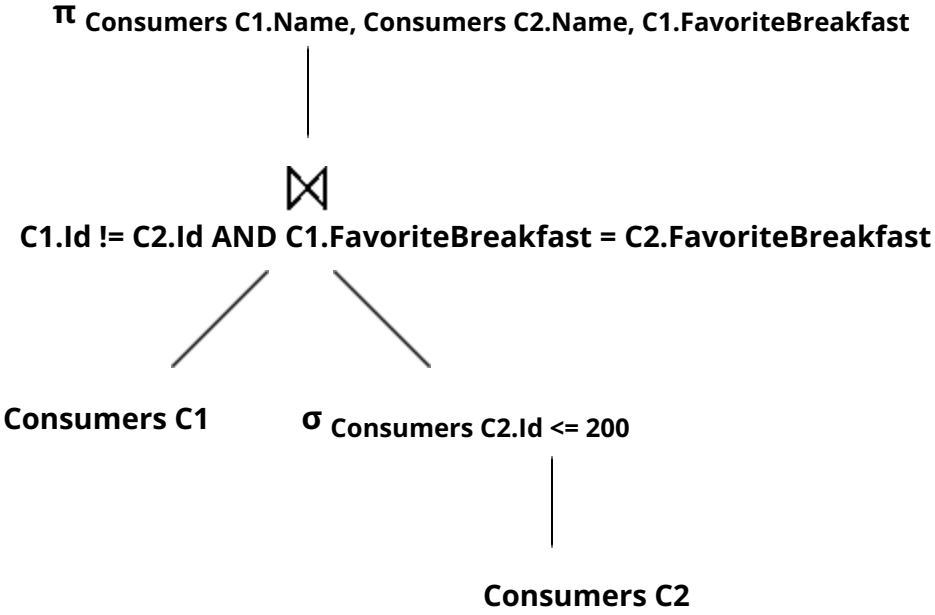
(iii) Cost = **M+(M/(BP-2))*N**

= 5000 + (5000/(75-2))*5000
= 5000 + (5000/73)*5000
= 5000 + 342465.75
= 347465.75

So, the cost for Block Nested Loop Join would be **347466 (approximately)**

b. Estimate the cost of this query using:

i. **HashJoin**
ii. **Sort-Merge Join**
iii. **BlockNestedLoop Join**



Let **M** be the number of pages in 'Consumers' C1 Relation
Let **N** be the number of pages in 'Consumers' C2 Relation
Let $\mathbf{N_f}$ be the number of pages in 'Consumers' C2 Relation

$\mathbf{N_f = 10}$

(i) Cost = $\mathbf{M+N_f}$
= 5000 + 10
= 5010

So, the cost for Hash Join is **5010.**

(ii) Cost = $\mathbf{3*(M+N_f)}$

Because $M+N_f > BP$

Cost = 3*(5000+10)
= 3*(5010)
= 15030

So, the cost for Sort-Merge Join would be **15030.**

(iii) Cost = $\mathbf{M+(M/(BP-2))*N_f}$

= 5000+(5000/(75-2))*10
= 5000+684.93
= 5684.93

So, the cost for Block Nested Loop Join would be **5685 (approximately)**

c. For the hash joins, what attribute would you build the hash join on?

I would build a hash join on **Consumers.FavoriteBreakfast**

Because, there are 175 unique values for that attribute and it would require approximately 8 pages which is less than BP(75). This would fit in the memory and hence I chose that attribute

Q6 Query Optimization (20 points).
For the query in Q5.b., give two specific examples of how your estimate of the cost of the join could be incorrect. (Please do not just repeat the information in the slides, instead, translate that information into specific examples using the query above.)

One instance where the cost estimate could be incorrect is when the catalog stats are out-of-date

For example, there are new consumers(like 1000 new consumers) added to the 'Consumers' table but the catalog stats are not updated, and when the cost estimation is done with the old catalog stats, the estimates would be incorrect.

Another instance, cost estimate could be incorrect is when the cost estimation errors are compounded over time.

For example, as mentioned in the above example if the mistake is made multiple times, then the overall cost estimates would be incorrect.

Q7 Concurrency (15 points).
Sometimes concurrency causes issues, sometimes it does not cause issues. For each pair of queries below tell whether there are any potential issues if that pair of queries ran at the same time or if there are no potential conflicts between the two queries.

Q1: UPDATE BreakfastFoods SET Calories = 210 WHERE Id = 1;
Q2: SELECT * FROM BreakfastFoods WHERE Id = 1;
Q3: UPDATE Companies SET Name = 'Peets' where Id = 5;

Q1 & Q2:
Yes, there are potential issues with this pair. Q2 might read the tuples in the table before the Q1 execution is completed.
Q1 & Q3:
No, there aren't any potential conflicts because both the UPDATE(s) are on different tables.
Q2 & Q3:
No, there aren't any potential conflicts because both the UPDATE(s) are on different tables.

Q8 Concurrency (10 points).
List two reasons supporting concurrency is a good idea.

- When different queries run concurrently, the overall throughput is increased.
- Concurrency enables better utilization of resources i.e. while one process uses the CPU, the other can write the disk and vice-versa.

Q9 Recovery (20 points).
Peanut Butter Granola has become a very popular breakfast food. Consider the series of SQL commands below which modify the database to show that Daniela, Miranda and Kiran all now list Apple Blueberry Granola as their favorite breakfast food.

```
BEGIN;
UPDATE Consumers SET FavoriteBreakfast = 3 WHERE Id = 4;
UPDATE Consumers SET FavoriteBreakfast = 3 WHERE Id = 5;
UPDATE Consumers SET FavoriteBreakfast = 3 WHERE Id = 6;
COMMIT;
```

At this point in time, after these commands have been executed:

- **The page that contains the tuple with Id = 4 (Jasmine) is in memory.**
- **The page that contains the tuple with Id = 5 (Miranda) has been written to disk.**
- **The page that contains the tuple with Id = 6 (Jorge) has been written to disk.**

And now, the system crashes and memory is lost.

Please answer:

a. **Which of the three tuples (give Id and Name) are problematic in this situation?**

   (4, Jasmine) would be problematic as it is still in memory, and the system crashed.

b. **What needs to be done to that tuple or tuple to fix the issue? Redo or Undo?**

   Redo should be done

Q10 Transactions & Recovery (15 points).
You receive a tip that Olivia's (Id 1) favorite breakfast is now Coffee with half & half, you begin to write a query to update the database… but then you are told that actually, Olivia's favorite breakfast has not changed and remains Peanut Butter Granola, thank goodness you got that information before you'd committed the transaction. Thus, you have typed the following series of SQL commands.

```
BEGIN;
UPDATE Consumers SET FavoriteBreakfast = 5 WHERE Id = 1;
ROLLBACK;
```

At this point, the page containing the tuple with Id=1 has been written to disk.

Please answer:

a. **Do you have a problem or not?**

   There is no issue, because the transaction has been rolledback

b. **If you do have an issue, what needs to be done to fix the issue? Redo or Undo?**

   The problem arises when the system crashes at the given point. If that's the case it should be 'UNDO'.