CS 486/586 Final

Spring 2022

Any notes or updates would be added here:

In **Q6 Query Optimization -** instead of "For the query in **Q6.b**." it needs to be changed to **"**For the query in **Q5.b**."

Instructions.

This file contains the Final Exam for CS 486-586 - Spring 2022.

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 file 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 Final is **due Thursday 06/09 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 answer based on your understanding of the material and you must cite the website or material that you used.** For example, you could include a link to the website you used or the name of a book you used in your answer.

200 points total.

Please make sure to read questions completely.

Sports Schema with Catalog Tables:

This exam uses the Sport Schema, as you are somewhat familiar with. The name of the Sports table is changed to SportsInfo. 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.

Note:

Football refers to international football or soccer
METs: Metabolic Equivalents
Weight class: could be a number from 1-5
AverageCalories: Refers to the averaged amount of calories burnt by an individual in a specific weight class during one hour.

Sport Schema:

SportsInfo

Id	Name	Type	METs	DifficultyRank
1	Volleyball	Team	4.0	20
2	Football	Team	8.0	10
3	Basketball	Team	6.0	4
4	Weight lifting	individual	6.0	44
5	Boxing	individual	7.0	1

Players

Id	Name	Sport	Age	WeightClass
1	Arin	2	22	2
2	Inaya	1	34	2
3	Ashley	5	40	3
4	Charlie	4	35	4
5	Owen	3	52	1
6	Li	2	19	3

7	Zaina	3	20	1

Calories

Id	**Sport**	**WeightClass**	**AverageCalories**
1	1	2	285
2	2	2	498
3	2	3	623
4	3	1	365
5	3	2	715
6	4	4	526
7	5	3	434

attribute_info

relationname	**attrname**	**minvalue**	**maxvalue**	**Distinctvalues**
SportsInfo	Id	1	6000	6000
SportsInfo	Name	--	--	6000
SportsInfo	Type	--	--	180
SportsInfo	METs	1	20	40
SportsInfo	DifficultyRank	1	60	60
Players	Id	1	100,000	100,000
Players	Name	--	--	98,000
Players	Sport	1	6000	6000
Players	Age	1	100	100
Players	WeightClass	1	5	5
Calories	Id	1	30000	30000
Calories	Sport	1	6000	6000
Calories	WeightClass	1	5	5
Calories	AverageCalories	100	2000	25000

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 burnt in playing a sport for an hour is 100.
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 6,000 different sports on the SportsInfo table.
Note: min and max values are not provided for string attributes

relation_info

Relationname	**Numtuples**	**numpages**	**tuplesperpage**
SportsInfo	6,000	600	10
Players	100,000	5,000	20
Calories	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

Name: Parth Parashar

NOTE for the Instructor: -ALL THE COST ESTIMATION NUMERICALS WILL PRODUCE ANSWERS WITH UNIT I/O(s).

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 SportsInfo
   WHERE Id = 50;
   ```

2. ```
   SELECT *
   FROM SportsInfo
   WHERE METs <= 10 and METs > 12.5;
   ```

3. ```
   SELECT *
   FROM SportsInfo S, Players P
   WHERE S.Id = P.Sport
           AND S.METs = 7.0;
   ```

Indexes:

A. clustered index on SportsInfo.Id
B. unclustered index on SportsInfo.DifficultyRank
C. unclustered index on SportsInfo.METs
D. clustered index on Players.Sport

Query	Indexes	Brief Justification
1 (5 points)	A	Because the where clause is on only ID and it will directly cluster based on it.
2 (5 points)	C	Because the where clause uses single column for matching which will act as
   ```

		the indexing column.
3 (5 points)	C, D	Because the use of clustering on players.sport will have in the index value sorting using SportsInfo.MET

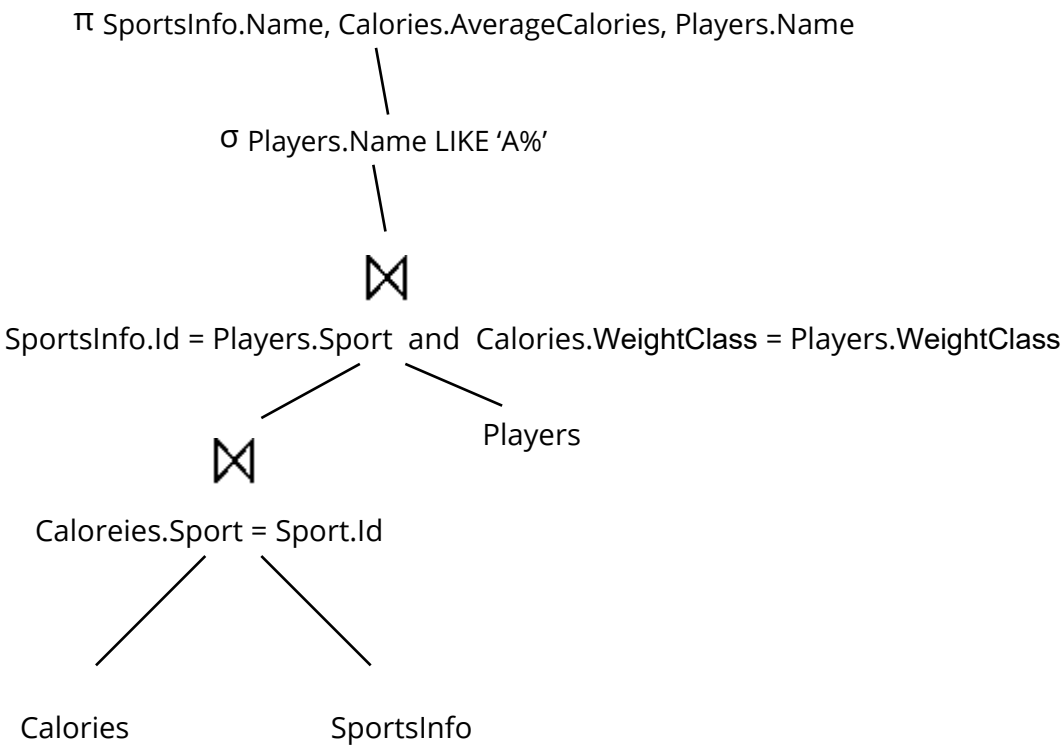
Q2 RA Equivalences (20 points).

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.

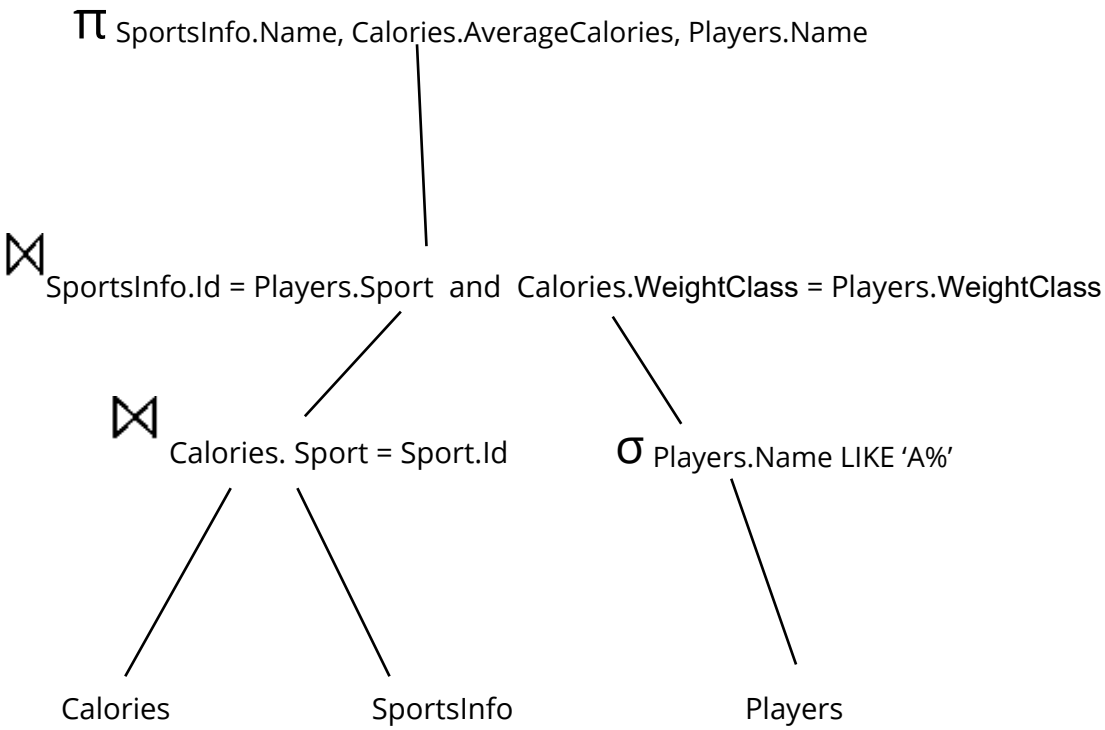
- Use the two RA Equivalences below in your transformations.
- Use the RA Equivalences in the order given, that is the first transformation should be done with RA 1 and the second transformation done with RA 2.

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

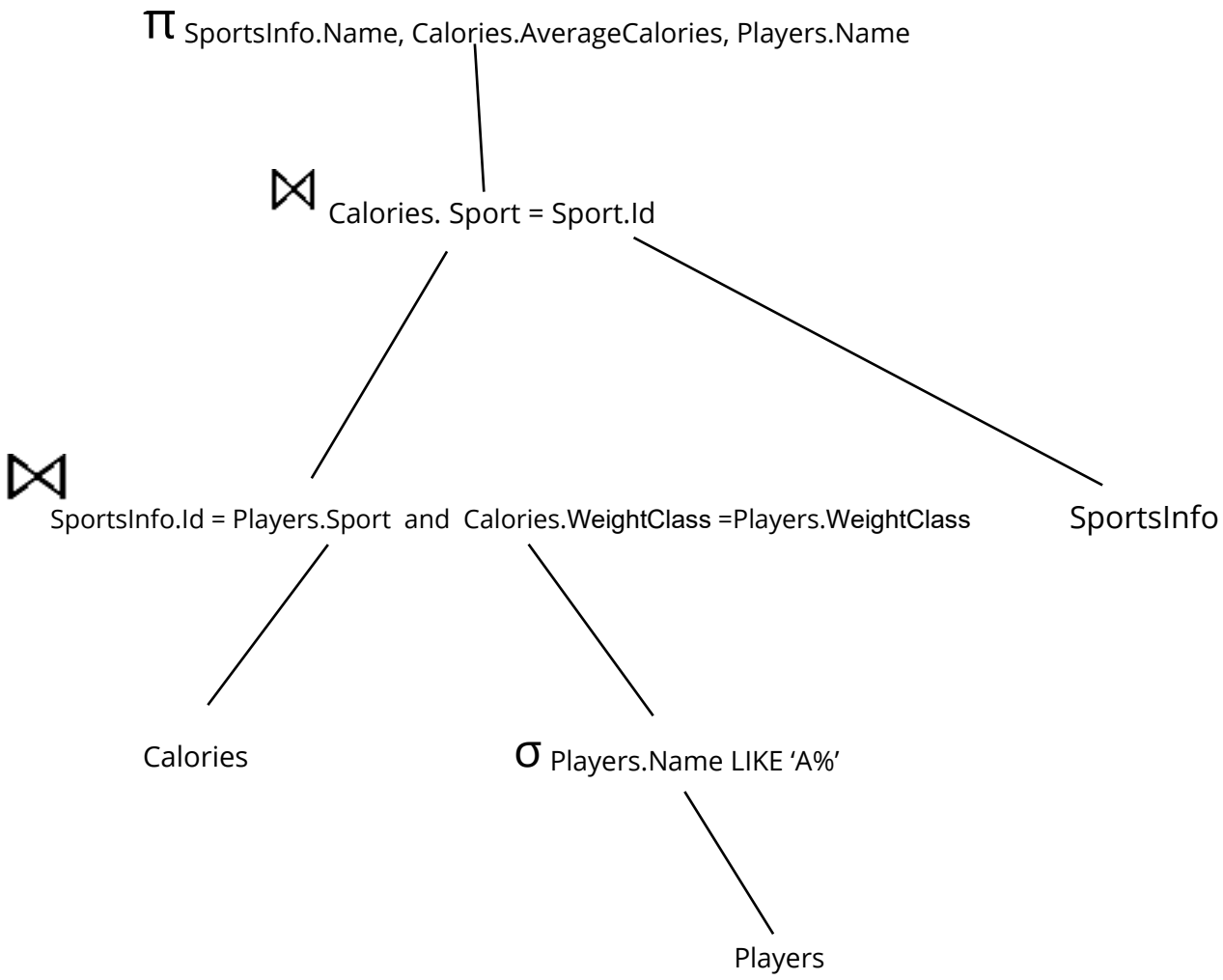
  RA 2: $R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$.



Answer: - When the given question is converted to **RA1** according to the conditions given, it becomes as: -



Now for **RA2**, we will have the following result: -



Q3 Cost Estimation (20 points).

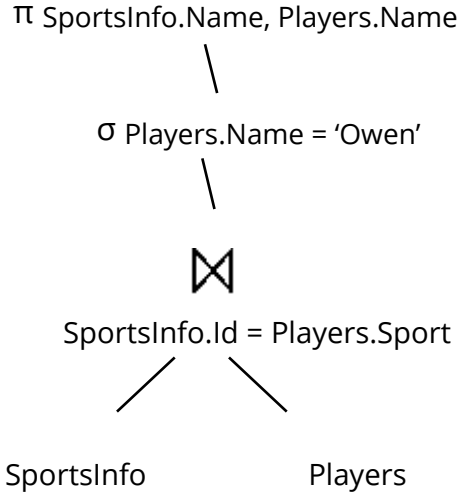
For the two RA trees below, estimate the join cost assuming the query optimizer chooses a hash join.

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.



Answer: - Let M be the number of pages in 'Sports Info' Relation

Let N be the number of pages in 'Players' Relation

Since we are using a Query Optimizer which chooses Hash Join, the estimated cost of the join becomes as: -

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

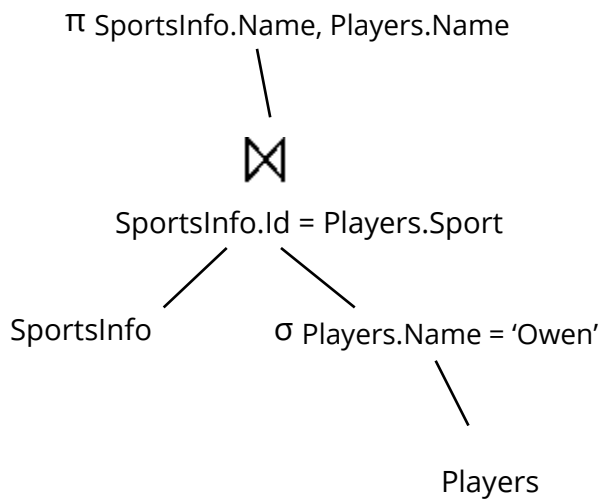
=3(5600)

= 16800

So, the estimated cost of the hash join is 16800

I would build a hash table on 'Sports Info.'

b.



Answer: - Let M be the number of pages in 'Sports Info' Relation
Let N be the number of pages in 'Players' Relation
Let X be the number of pages after the SELECT query. (For N)

The estimated cost of the join would be M+X.
In this case, the value of X would be equal to '1' because the value of select statement of players Name='Owen' is 1. This is because it is a distinct value present in the table. So the value of X becomes as 1.

This means that the estimated cost becomes as: -
M+X = 600+1 = 601
I would build a hash table around 'Sports Info'

Q4: Query optimization - join selection (20 points)

Mark true or false for the following statement and a brief (one sentence) reasoning for your choice.

- Block nested loop is suitable when we have equi-joins
- Hash-join requires indexing
- Block-nested loop is efficient
- Sort-merge join is suitable when both relations are large (cannot fit in memory)

Answer: -

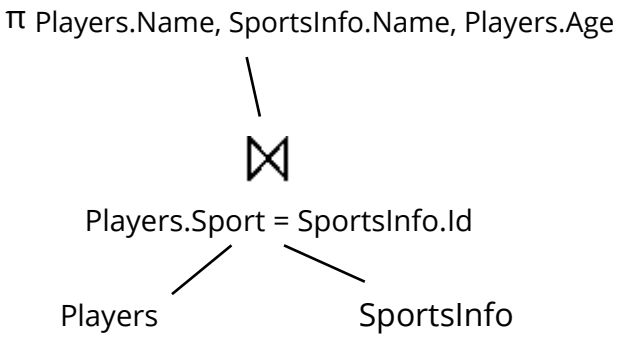
1) Block Nested loop is suitable when we have equi-joins. This statement is **FALSE.** This is because while Block-nested loop works both for equi-joins and non-equi joins relations, but at the same time, it is not very efficient at that. Other algorithms such as Hash Joins are much more efficient than block nested loop
2) Hash Joins requires indexing. This statement is **FALSE.** This is because Hash joins use Hash Tables for joining the predicates.
3) Block Nested Loop is efficient. This statement is **FALSE.** This is because block nested loop works with both equi-joins and non-equi joins, it still requires scanning a large portion of the table to form blocks to be used as predicates. This means that the calculation (I/O access) is still significantly big. There are other algorithms such as Hash-join and merge-sort join which are super-efficient than the Block Nested Loop.
4) Sort merge is suitable when both the relations are large. This statement is **TRUE.** Sort-merge is generally used when both the relations don't fit in the memory at the same time. The cost also indicates this as it is 3(M+N). Also, Sort-merge is based on the sorting and merging operations. What makes these operations super-efficient is the fact that if the merge has already resulted in sorted array in the previous steps, then sorting will be inexpensive.

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



Answer: -

Let M be the number of pages in the “Players” Relation

Let N be the number of pages in the “SportsInfo” Relation

1) Hash Join: -

M = 5000

N = 600

The first step that we need to do is calculate the square root of M and N

Therefore, the square root of M is: - 70.71

Therefore, the square root of N is: - 24.49

We can see that neither of the relations will fit in the memory.

Since sqrt(M) < BP or sqrt(N)<BP, therefore we will follow the Harder case and Cost becomes as 3(M+N)

Cost = 3*(M+N)

Cost = 3* (5000+600)

Cost = 3* (5600)

Cost = 16800

The cost for Hash join would be 16800.

2) Sort-Merge Join: -

M = 5000

N = 600

Sqrt(M) = 24.49

Sqrt (N) = 70.71

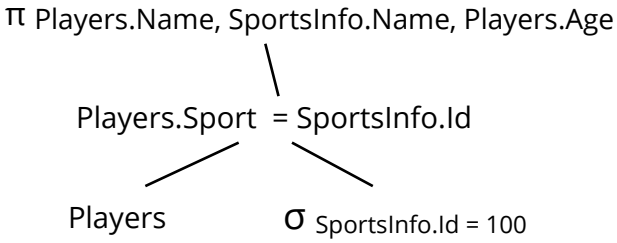
We can see that neither of the relations will fit in the memory.
(M+N) > BP and sqrt(M) < BP and sqrt(N) < BP
Cost = 3*(M+N)
Cost = 3*(5000+600)
Cost = 3*(5600)
Cost = 16800

The cost for sort-merge would be 16800.

3) Block Nested Loop Join
M = 5000
N = 600
Cost = M + (M/(BP-2))*N
Cost = 5000 + (5000/73)*600
Cost = 5000 + 41095.89
Cost = 46,095.89
The cost for block nested loop join would be 46096 approximately.

b. Estimate the cost of this query using:

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



Answer: -

Let M be the number of pages in the 'Players' relation
Let N be the number of pages in the 'SportsInfo' table
Let X be the number of pages in the 'SportsInfo' table after the value for select sports info has been factored in.

1) Hash Join: -

M = 5000

N = 600

Now we know that the number of distinct values for players' id is 1 for id=1.

Therefore, X = 1.

Since X fits in the memory and X<BP

Cost = M+N.

Cost = M + X (Replace the value of N by X)

Cost = 5000 + 1

Cost = 5001

So, the cost for hash join becomes as 5001

2) Sort-Merge Join

M = 5000

N = 600

X = 1

Now since M+X > BP

Therefore, Cost = 3*(M+X) (Replace the value of N by X)

Cost = 3*(5000 + 1)

Cost = 3*(5001)

Cost = 15003

So, the cost for Sort-merge becomes as 15003

3) Block-Nested Loop Join

M = 5000

N = 600

X = 1

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

Cost = 5000 + (5000/75-2)*1 (Replace the value of N by X)

Cost = 5000 + (5000/73)

Cost = 5068.49315

So, the cost for block-nested loop join becomes as: - 5069 approximately.

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

Answer: - In general, I would build a hash join on Players.Age attribute. This is because it has 100 distinct values which will require 5 pages which is less than BP(75). This would fit in the memory and will therefore be suitable to build a in-

memory hash table which would be fast to access. It would speed up the entire process as well.

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.)

Answer: - 1) when the data in the attribute_info table has not been updated for a long time and it is out-of-date whereas the data in the other tables (relation_info) have been updated with new data.

This would mean that the calculations would be faulty and there could be more than 1 person for the select statement.

For example, there are new sports (like 10 new sports type with the sports id as the same). We will be calculating with the old value which would lead to wrong cost estimations

2) Cost estimation errors compounded over time could lead to wrong cost estimations.

This means that when the cost estimation errors are not rectified and the same mistake is made over a longer period of time, then the cost estimates are wrong.

3) Wrong value updation in the attribute_info table as compared to the relation_info table.

4)Translated values from attribute_info to the page calculation is wrong. This means that the specific column value from the rows are not considered due to null values. This means that the data in the attribute_info table would mean that several null values are considered as one and hence leads to wrong calculations.

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 SportsInfo SET DifficultyRank = 22 WHERE Id = 1;
Q2: SELECT * FROM SportsInfo WHERE Id = 1;
Q3: UPDATE Calories SET AverageCalories = 500 where Id = 2;

Q1 & Q3:
Q1 & Q2:
Q2 & Q3:

Answer: -

1) Q1 and Q3: - No there are no potential conflicts because the updates are on different tables.
2) Q1 and Q2: - Yes, there could be potential conflicts and there are definite potential issues with this pair. This is because Q-2 might select and read the tuples in the table SportsInfo before the execution of Q1 (update) has been completed leading to false data retrieval.
3) Q2 and Q3: - No, there are no potential conflicts with this combination because both the queries are run on different tables

Q8 Concurrency (20 points).

List two reasons supporting concurrency is a good idea.

Answer: -

1) Concurrency is a good idea because when different queries are run together, then the overall throughput is increased. This also enables faster query results and the overall user experience in the context of a bigger project increases where data is being loaded from the database to the supposedly front-end of the application which the user is using.
2) Concurrency enables better utilization of resources i.e., while one process uses the CPU, the other process can perform some other work. In the context of the database, while one process can handle one query on a specified location, the other process can handle other query on the other

specified location. All this leads to better utilization of resources. To enable this, generally databases have thread pools which enables concurrency.

Q9 Recovery (20 points).

Football has become a very popular Sport. Consider the series of SQL commands below which modify the database to show that Ashley, Charlie and Owen all now list Football as their main sport.

```
BEGIN;
UPDATE Players SET sport = 3 WHERE Id = 3;
UPDATE Players SET sport = 3 WHERE Id = 4;
UPDATE Players SET sport = 3 WHERE Id = 5;
COMMIT;
```

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

- The page that contains the tuple with Id = 3 (Ashley) is in memory.
- The page that contains the tuple with Id = 4 (Charlie) has been written to disk.
- The page that contains the tuple with Id = 5 (Owen) 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?

Answer: - In this situation, Ashley with ID 3 would be problematic because it is still in memory and the system has crashed. The other two have already been written in the disk and hence they will not be problematic to commit or rollback but since Ashley is still in memory, it could lead to potential problems and might never be recovered.

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

Answer: - Redo is the correct way forward to fix this situation.

Q10 Transactions & Recovery (20 points).

You receive a tip that Arin's (Id 1) main sport has changed to Volleyball (Id 1), you begin to write a query to update the database... but then you are told that actually, Arin's main sport has not changed and remains Football (Id 2), 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 sport = 1 WHERE Id = 1;
```

At this point the user types ROLLBACK.

Please answer:
What does the system need to do to make sure that Arin's main sport has not changed and remains Football?
Answer: - The sequence of queries that have been types are: -
BEGIN
UPDATE Consumers SET sport=1 where Id=1;
ROLLBACK

To ensure that Arin's main sport has not been changed and remains Football, the rollback command needs to be executed. This means that the system must not crash between the execution of update and rollback command.
We can also make use of the READ UNCOMMITTED, in case of system failure but that means that we will have to use different processes to do that.
Ensuring that the system does not crash before ROLLBACK is the best way to achieve the desired solution.

There can also be the use of UNDO command which can help us retrieve the lost original value in case rollback does not happen after the update.

Extra Credit: Hash Join (15 points)

List two cases that a database system will prefer to use a hash join when running a query.

Answer: - The two cases when the system will prefer to use a hash join when running a query is: -

1) If one-relation fits in memory and the other relation is too large to fit in the memory, in that case, we will make use of the hash-join. In this case, the cost estimation will be: -
   Cost = M+N where M and N are the number of tuples in the two tables respectively. The pre-requisite for this is M<BP and N<BP
   This scenario is also known as simple case

2) If neither of the two relations fits in the memory, then, we will make use of the hash-join as well.
   In this case, the cost estimation would be 3*(M+N) where M and N are the number of tuples in each relations respectively. The pre-requisite for this is sqrt(M)<= BP or sqrt(N)<=BP.
   This case is known as the Harder case.