CS 564 Problem Set #3

A: THE YELP DATABASE **[70pts]**

Suppose we are given a database with the following schema.

Users (UserID INTEGER, Name CHAR(30), Age INTEGER, ReviewCount INTEGER)

Businesses (BusinessID INTEGER, BName CHAR(30), City CHAR(20), State CHAR(2))

Checkins (BusinessID INTEGER, Weekdays INTEGER, Weekends INTEGER)

Reviews (ReviewID INTEGER, UserID INTEGER, BusinessID INTEGER, Stars REAL)

Reviews (UserID) is a foreign key referring to **Users** (UserID).
Reviews (BusinessID) is a foreign key referring to **Businesses** (BusinessID).
Checkins (BusinessID) is a foreign key referring to **Businesses** (BusinessID).

A page is 8 kB in size (1kB = 1024B). The RDBMS buffer pool has **10,000** pages, all of which are available. Initially, the buffer pool is empty.

The relation instances have the following statistics. Assume there are no `NULL` values. Each integer or real is 8B, and each character is 1B (so as an example CHAR(20) is 20B).

Relation	Number of Pages
Users	75,000
Businesses	42,000
Checkins	20,000
Reviews	500,000

Answer the following questions. *Clearly explain how you obtained your answer for each.*

1. **[15pts]** Name 5 different indexes (hash, clustered B+ tree) on the table Users that **match the predicate** in the following SQL query.

```
SELECT  *
FROM    Users
WHERE   NOT ((Name <> "John" AND NOT (Name = "Mary"))
        OR (Age <> 20 AND Age <= 50));
```

2. **[15pts]** Suppose we are given a clustered B+ tree index each on Businesses (BusinessID) and Checkins (BusinessID). Also, suppose that both indexes follow the alternative of storing the data records directly in the leaf pages of the index. Which join algorithm among the following has the **lowest I/O cost** for a **natural join** of Businesses and Checkins: (a) Block Nested Loop Join, (b) Sort-Merge Join, or (c) Hash Join? Your answer must calculate the I/O cost of all three algorithms.

3. **[15pts]** Suppose that there is no index on the Businesses relation. Consider the following SQL query.

```
SELECT    City, COUNT (BusinessID)
FROM      Businesses
GROUP BY  City;
```

   What is the **maximum number of cities** for which it is possible to implement hash-based aggregation using a **one pass** algorithm? The fudge factor of creating an in-memory hash table is $f = 1.4$. Show all of your calculations clearly.

4. **[15pts]** Suppose that there are no indexes on any relation and no relation is sorted on any attribute. Propose a physical plan for the following SQL query that achieves the **smallest possible I/O cost**. Assume that the values of `Stars` are real numbers uniformly distributed between 1 and 5 (inclusive), and the values of `Age` are integers uniformly distributed between 10 and 99 (inclusive). Show all of your calculations clearly.

```
SELECT  COUNT (UserID)
FROM    Users U, Reviews R
WHERE   U.UserID = R.UserID AND R.Stars < 1 AND U.Age = 18;
```

5. **[10pts]** Consider the following query expressed in Relational Algebra:

$$\pi_{BName}(\sigma_{Stars>4 \wedge ReviewCount \geq 100}((Users \bowtie Reviews) \bowtie Businesses))$$

   Write an equivalent Relational Algebra expression where the selections and projections are pushed down the plan as far as possible.

B: ADDITIONAL QUESTIONS **[30pts]**

1. **[15pts]** Suppose we are joining two tables S and R with respective number of pages $4BN_S$ and $8BN_R$, wherein $4BN_S \gg 8BN_R$. The number of buffer pages is $4B + 1$ and the buffer pool is initially empty. We are also given that $2fN_R = 4B - 1$, where f is the hash table fudge factor.

   The distribution of the join attribute values in S and R are such that after the first hash partitioning phase, we get exactly $4B$ partitions of S, each of length N_S pages, but not all partitions of R are of the same length. Suppose R gets partitioned as follows: $2B$ partitions of length N_R pages, B partitions of length $2N_R$ pages, and B partitions of length $4N_R$ pages.

   What is the I/O cost of the regular hash join algorithm discussed in class? Exclude the cost of writing the output of the join. Assume perfect uniform splitting occurs during the recursive repartitioning. Show all of your calculations clearly.

   (Hint: The answer is of the following form: $xBN_S + yBN_R$, where $x \in \{12, 14, 16, 18\}$ and $y \in \{24, 28, 32, 36\}$.)

2. **[15pts]** The *mode* of a list of values is the most frequent value. There can be more than one mode for a list. For example, the list $\{5, 2, 2, 3, 6, 6, 2, 5, 5, 10\}$ has two modes, 5 and 2. Assume that no index is available. Suppose we want to compute the mode of attribute A of a relation R with $N = 1,000,000$ pages. The size of the buffer pool is $B = 1,100$ frames.

   Describe a 2-pass algorithm that computes the mode, and compute its I/O cost.

   (Hint: Your algorithm should work in 2 passes for every possible input.)

DELIVERABLES

Submit your answers using a **single pdf** file. Upload the file at Canvas (under PS3).