# Project Proposal:
## Simulating Buffer Manager Strategies for Join / Selection Queries

**Group: disks_overloaded**
**Members:**

- **Utsav Basu (20CS30057)**
- **Anamitra Mukhopadhyay (20CS30064)**
- **Rounak Saha (20CS30043)**
- **Swarup Padhi (20CS30062)**
- **Spandan Halder (20CS10063)**

### Introduction:

This project aims to simulate a small buffer pool for simple join/selection queries on a few small tables. Popular buffer manager strategies such as LRU/MRU/CLOCK/Pinned blocks will be simulated, and these strategies will be compared in terms of the number of disk I/O required. For a more realistic simulation, we will use the SQLite C Library.

### Objective:

The main objective of this project is to compare and analyse the performance of different buffer manager strategies for Join/Selection queries in terms of the number of disk I/O required.

### Methodology:

The simulation will be implemented in the C/C++ language, and the SQLite C library will be used for more realistic simulation. We will create a small database with a few small tables that will be used for join/selection queries. We will implement the following buffer manager strategies:

**Least Recently Used (LRU):** In this strategy, the block that has not been accessed for the longest time will be replaced.

**Most Recently Used (MRU):** In this strategy, the block that has been accessed most recently will be replaced.

**CLOCK:** In this strategy, the buffer manager maintains a circular list of blocks. The blocks are marked with a bit indicating whether they have been accessed since the last time they were considered. When a block is to be replaced, the buffer manager scans the list of blocks in a circular fashion, and the first block with the access bit set to 0 is replaced. If all the blocks have their access bit set to 1, the buffer manager starts again from the beginning of the list.

**Pinned Blocks:** In this strategy, some blocks are marked as pinned, which means they cannot be replaced.

We will simulate join/selection queries using each of these strategies and record the number of disk I/O operations required for each strategy.

**Flowchart:**
Create a small database with a few small tables
Initialize buffer pool

**Implement LRU strategy**
a. Access the buffer pool
b. Replace the least recently used block
c. Record the number of disk I/O operations required

**Implement MRU strategy**

a. Access the buffer pool

b. Replace the most recently used block

c. Record the number of disk I/O operations required

**Implement CLOCK strategy**

a. Access the buffer pool

b. Scan the list of blocks in a circular fashion

c. Replace the first block with the access bit set to 0

d. Record the number of disk I/O operations required

**Implement Pinned Blocks strategy**

a. Pin some blocks in the buffer pool

b. Access the buffer pool

c. Replace unpinned blocks

d. Record the number of disk I/O operations required

Analyse and compare the performance of each strategy in terms of the number of disk I/O operations required.

Conclude and suggest the best strategy for join/selection queries.

**Conclusion:**

In conclusion, this project aims to simulate a small buffer pool for simple join/selection queries on a few small tables. The popular buffer manager strategies such as LRU/MRU/CLOCK/Pinned blocks will be simulated, and their performance will be compared in terms of the number of disk I/O required. This project will provide insights into the best buffer manager strategy for join/selection queries.

```
+-------------------------------------------------------+
|                     Create database                   |
+-------------------------------------------------------+
                            |
                            |
                            |
                            V
+-------------------------------------------------------+
|                Initialize buffer pool                 |
+-------------------------------------------------------+
                            |
                            |
                            |
                            V
+-------------------------------------------------------+
|                        LRU Strategy                   |
+-------------------------------------------------------+
|  Access buffer pool   |  Replace least recently used  |
|  Record disk I/O ops  |  block                        |
+-----------------------+-------------------------------+
                            |
                            |
                            |
                            V
+-------------------------------------------------------+
|                        MRU Strategy                   |
+-------------------------------------------------------+
|  Access buffer pool   |  Replace most recently used   |
|  Record disk I/O ops  |  block                        |
+-----------------------+-------------------------------+
                            |
                            |
                            |
                            V
+-------------------------------------------------------+
|                     CLOCK Strategy                    |
+-------------------------------------------------------+
|  Access buffer pool   |  Scan blocks in circular list |
|  Record disk I/O ops  |  Replace first block with     |
|                       |  access bit set to 0          |
|                       |                               |
+-----------------------+-------------------------------+
                            |
                            |
                            |
                            V
+-------------------------------------------------------+
|                 Pinned Blocks Strategy                |
+-------------------------------------------------------+
|  Pin blocks           |  Access buffer pool           |
|  Access buffer pool   |  Replace unpinned blocks      |
|  Record disk I/O ops  |                               |
+-----------------------+-------------------------------+
                            |
                            |
                            |
                            V
+-------------------------------------------------------+
|           Analyze and compare performance             |
+-------------------------------------------------------+
                            |
                            |
                            |
                            V
+-------------------------------------------------------+
|              Draw conclusions and suggest             |
|                 best strategy for queries             |
+-------------------------------------------------------+
```