Exercise 2
Released: April 08, 2019 · Discussion: April 15, 2019

1 Buffer Manager — Replacement Strategies

The lecture discussed the functioning of the buffer manager.

• What are the responsibilities of the buffer manager?
• Describe the strategies used to fulfill these responsibilities.
• What are their advantages and disadvantages?

2 Replacement Strategies — Implementation

In the lecture different replacement strategies for buffer managers were discussed. In this exercise, you have to implement at least the two following replacement strategies for a simple buffer manager:

• Least Recently Used
• Clock

Download the file 01_buffer_manager.zip from the course website¹ and extract it. The archive contains a rudimentary buffer manager with a random replacement strategy as a cmake project.

• Complete the files src/storage/lru_policy.cpp and src/storage/clock_policy.cpp respectively.
• If you implement more than the given strategies you may have to modify the file 01_buffer_manager.cpp as well.
• In order to test your replacement strategies we provide the file workload/postgres_buffer_manager_trace.txt which contains a set of buffer manager calls of a transactional workload². The attributes pin/unpin and reln describe the respective action and page number.

¹http://dbis.cs.tu-dortmund.de/cms/en/teaching/ss19/arch-dbms/exercises/01_buffer_manager.zip
²The trace was created by the TPC-C benchmark on a PostgreSQL instance.
• Which replacement strategy performs best, which performs worst? How many pages are evicted by the different strategies?

**Build instructions:**

1. Extract the archive and navigate into the extracted folder.
2. Run `cmake` to create a makefile for your system: `cmake .`
3. Run `make` to create an executable binary file: `make`
4. Execute the created binary file (e.g. `./01_buffer_manager` on linux)