ECS165A
Milestone 2 Overview
Haley Raizes, Brittany Bates,
Jim McHerney, Nicholas Chen, Chris Bried
Objectives

- Disk Structure
- Bufferpool Design
- Eviction Policy
- Indexing
- Merge
- Performance
- Questions
- Demo
The Path to Durability

Milestone 1

Volatile Memory

Milestone 2

Non-Volatile Memory
Bufferpool Design

Frame directory = {
(table_name, page_range, base/tail page index, is_base_record) : frame_index
}

Main Methods:
evict_page()
load_page()
commit_page()
at_capacity()
is_record_in_pool()
add_frame_to_directory()
commit_all_frames()

Frame Array

Frame 1 Frame 2 Frame 3 Frame 4 ...

*Bufferpool size is set in the configuration file

Frame:
- all_columns
- pin
- dirty_bit
- time_in_bufferpool
- access_count
- path_to_page_on_disk
- frame_key

* A frame loads 1 base or tail page
Eviction Policy

- Synthesis of LRU and LFU policies
  - Among the least frequently used records, evict the least recently used
- Chose this method primarily for speed & simplicity
- Does not distinguish between privileged and unprivileged data
Indexing

- Hash index
- Maps column values to list of RIDs
- Used for select and update
- Index created automatically for primary key
- Bufferpool frames are committed before index creation
- Index persisted as .pkl file
1. Call to update instantiates merge after X number of updates, where X is defined in config.py.

2. Merge gets its own thread.

3. Load in base and tail pages per page range for merge.

4. Bufferpool instance.

5. After Merge:
   - TPS = 30
   - 30 * 60
   - Write updated records back to disk.

* Merge uses a separate bufferpool instance that does not interfere with query.
Optimizations & Performance

*These times are based on 10 run averages using the provided __main__.py

Specs: 6 core Intel Core i7, 2.6 GHz, 256KB l2 cache per core, 12 MB L3 Cache, 16GB Memory
Questions
Demo