**Week 2: Storage Formats**

**Overview**

Databases are often designed to store data persistently (meaning, on-disk).  Databases take two general approaches for storing table data: by row or by column. This week, we are going to learn about the file formats that databases use to store data on disk. As part of that, we'll discuss the movement of data between disk and RAM and approaches for optimizing this movement.

A database could naively search its index (e.g., a B+-tree) starting from the root for every query but that would be slow and require many disk accesses.  Instead, databases often manage an in-memory cache of frequently-used data.

To see why this is beneficial, think of two subsequent queries.  In the first query, the database needs to read every node from the root of the tree down to the leaf.  In the second query, however, any nodes that in common along the paths from the root to the two key-value pairs are already in memory and do not be read again.  This reduces the number of nodes that need to be read from disk in order to execute a query.

The cache is often called a page cache because it caches entire pages from disk, rather than individual elements.  (Again, this is because disks read chunks of data with each read operation, even if you are only accessing a single byte).  The techniques for implementing a page cache are very similar to the caches used by the operating system to manage programs with data larger than memory and caches for each disk.  As a result, we're going to review concepts from memory management you would have seen in an undergraduate operating systems class and discuss their applications to databases.

(For your assignment, you will rely on the operating system's file caching functionality rather than implementing your own page cache.  Nonetheless, it's good for you to understand the concepts and challenges since it is important to database performance and consistency.)

**Reflection Questions**

* What is a heap file? How are data organized in heap files?
* What is a page cache (or database buffer)?
* What happens when a process requests a page that isn't in the cache?
* Generally, what problem are page replacement algorithms trying to solve?
* What are the differences between the LRU, toss-immediate, and MRU page replacement algorithms?
* Why can it be more efficient for database to manage the page cache instead of the OS?
* If a database serves multiple users, what are the advantages and disadvantages of global and per-user page caches?
* How could a page cache be problematic if multiple users are accessing the same data?
* How does column-oriented storage differ from row-oriented storage? What are its advantages and weaknesses?