<File Organization>
- Managing Disk Space: uses row-store to organize data // disk space is organized into files, files are made of pages, pages contain records
- Unordered Heap Files: contains records in no order / as file grows & shrinks, disk pages are allocated & deallocated, each record must have identifiers to track storage(pid(pageID), rid(recordID), free space)
- Heap File as a Linked List: Header page contains pointers to data pages. Pages link to each other, and pages with free space are tracked separately
- Heap File as a Page Directory: Instead of a linked list, a directory structure is used. Each entry keeps track of: Whether a page is free or full. Number of free bytes for efficient allocation. This approach speeds up finding pages for new records
- Page Organization: Since higher-level database operations work on records, not just pages, a file system must support: Insert/Delete/Modify operations. Record lookup using Record ID (rid). Scanning records based on conditions (e.g., filtering rows).
- Page Formats: Slotted Page Format: Stores records in slots with metadata. Each slot contains a record. The rid = (page id, slot number). Allows easy search, insert, and delete operations.
- Fixed-Length Records: Packed organization: Stores records in consecutive slots. Efficient but problematic for record references. Unpacked organization: Uses a bitmap to locate records instead.
- Variable-Length Records: Uses offsets and lengths to store variable-length fields. Insertion/Deletion Considerations: Deleted records set offset = -1. If no space is available, page reorganization occurs. rid remains unchanged, as it references a slot number instead of an absolute position.
- Record Format: Fixed Length: The same length and number of fields across all records. Field positions can be calculated from the system catalog.
- Record Format: Variable Length: Two storage options: Using delimiters between fields(e.g., $ markers). Using an offset array at beginning of record
- Column Stores: Traditional row-stores read entire rows, even when querying a single column.
- Column-Store Implementation: Data is stored vertically: Each column is placed in a separate file. Advantages: Faster queries for specific columns. Better compression (similar data types in one column). Suitable for OLAP (analytical queries). Disadvantages: Insertions and updates require modifying multiple files.
- Compression in Column Stores: Columnar storage enables better compression as same-column values