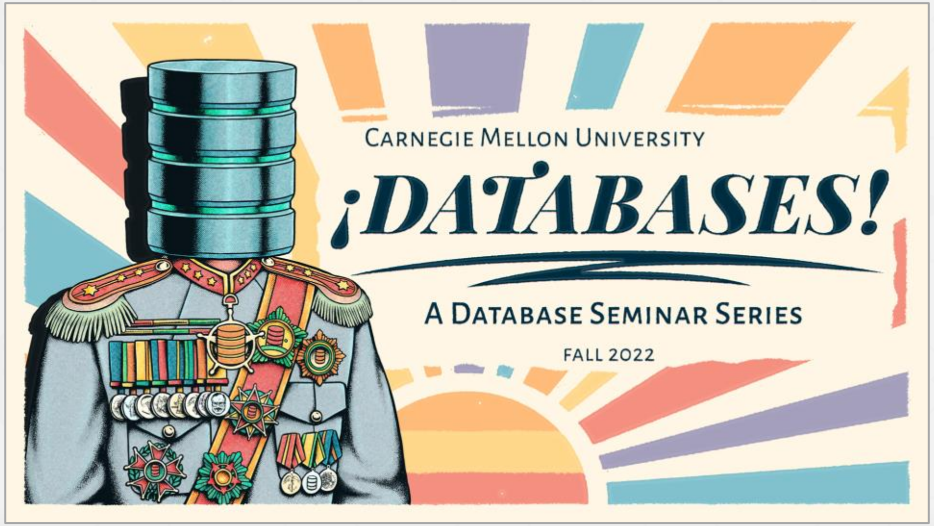
DATABASE TECH TALKS

Vaccination Database Tech Talks

→ Mondays @ 4:30pm (starting today)

→ https://db.cs.cmu.edu/seminar2022



DISK-ORIENTED ARCHITECTURE

The DBMS assumes that the primary storage location of the database is on non-volatile disk.

The DBMS's components manage the movement of data between non-volatile and volatile storage.

PAGE-ORIENTED ARCHITECTURE

Insert a new tuple:

→ Check page directory to find a page with a free slot.
→ Retrieve the page from disk (if not in memory).
→ Check slot array to find empty space in page that will fit.

Update an existing tuple using its record id:

→ Check page directory to find location of page.
→ Retrieve the page from disk (if not in memory).
→ Find offset in page using slot array.
→ Overwrite existing data (if new data fits).

DISCUSSION

What are some potential problems with the slotted page design?

→ Fragmentation
→ Useless Disk I/O
→ Random Disk I/O (e.g., update 20 tuples on 20 pages)

What if the DBMS could not overwrite data in pages and could only create new pages?

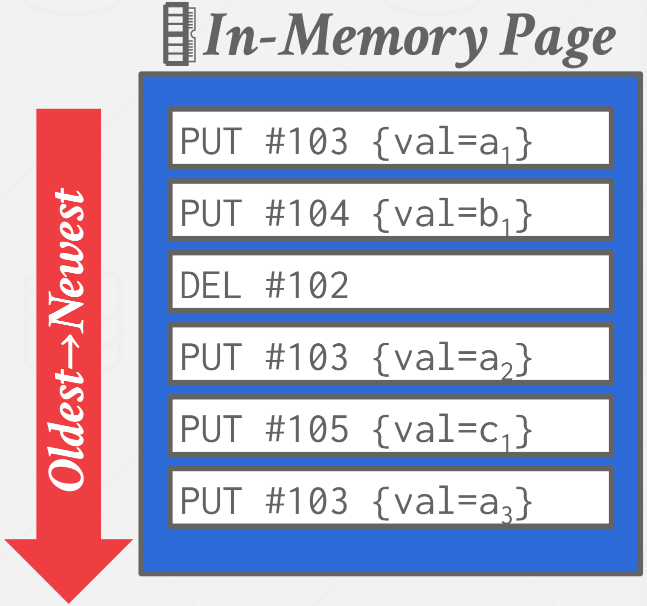
→ Examples: Cloud storage (S3), HDFS

LOG-STRUCTURED STORAGE

DBMS stores log records that contain changes to tuples (**PUT**, **DELETE**).

→ Each log record must contain the tuple's unique identifier.
→ Put records contain the tuple contents.
→ Deletes marks the tuple as deleted.

As the application makes changes to the database, the DBMS appends log records to the end of the file without checking previous log records.

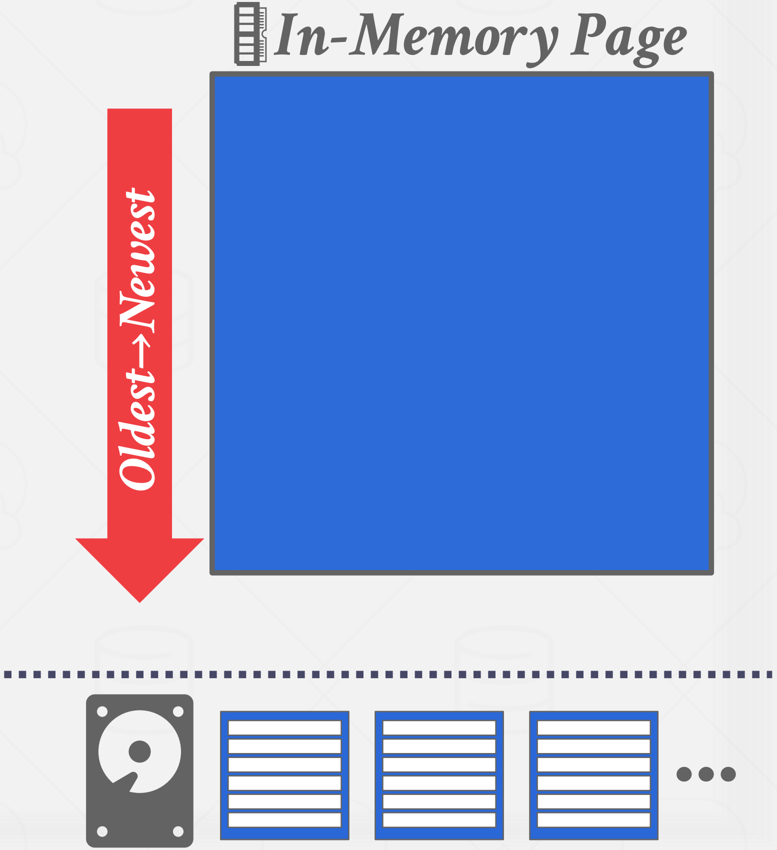


LOG-STRUCTURED STORAGE

When the page gets full, the DBMS writes it out disk and starts filling up the next page with records.

→ All disk writes are sequential.

→ On-disk pages are immutable.



LOG-STRUCTURED STORAGE

To read a tuple with a given id, the DBMS finds the newest log record corresponding to that id.

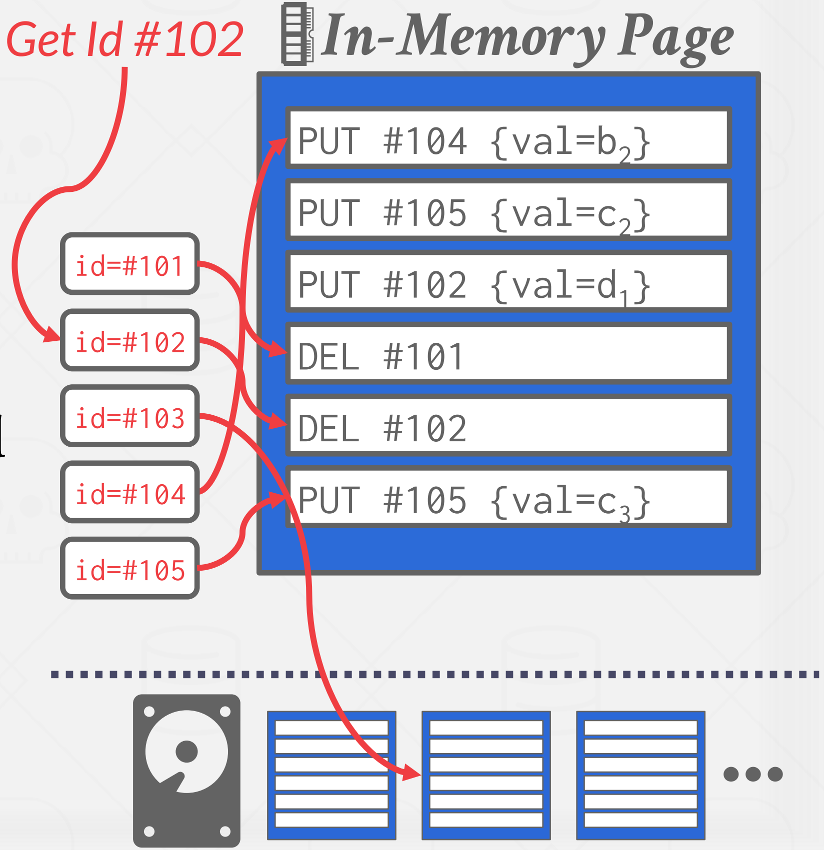
→ Scan log from newest to oldest.

Maintain an index that maps a tuple id to the newest log record.

→ If log record is in-memory, just read it.

→ If log record is on a disk page, retrieve it.

→ We will discuss indexes in two weeks.



LOG-STRUCTURED COMPACTION

The log will grow forever. The DBMS needs to periodically compact pages to reduce wasted space.

Page 1		*Page 2*		
PUT #103 {val=a_1}	+	PUT #104 {val=b_2}	→	PUT #103 {val=a_3}
PUT #104 {val=b_1}		PUT #105 {val=c_2}		PUT #104 {val=b_2}
DEL #102		PUT #102 {val=d_1}		DEL #101
PUT #103 {val=a_2}		DEL #101		DEL #102
PUT #105 {val=c_1}		DEL #102		PUT #105 {val=c_3}
PUT #103 {val=a_3}		PUT #105 {val=c_3}		

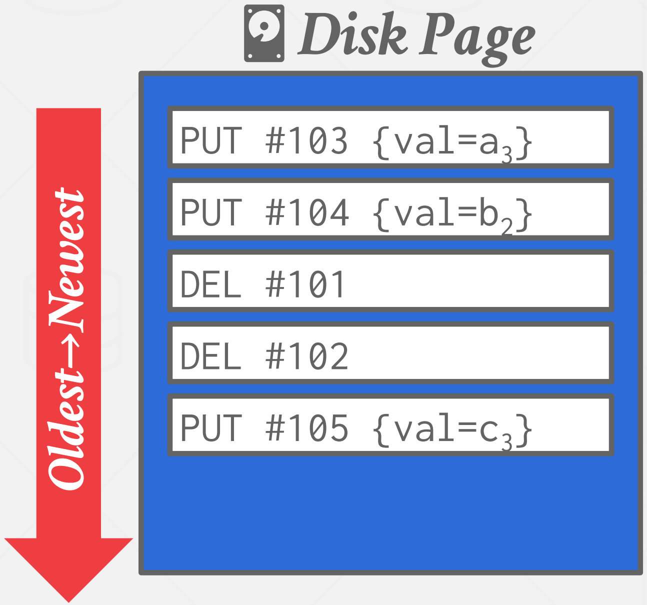
LOG-STRUCTURED COMPACTION

After a page is compacted, the DBMS does need to maintain temporal ordering of records within the page.

→ Each tuple id is guaranteed to appear at most once in the page.

The DBMS can instead sort the page based on id order to improve efficiency of future look-ups.

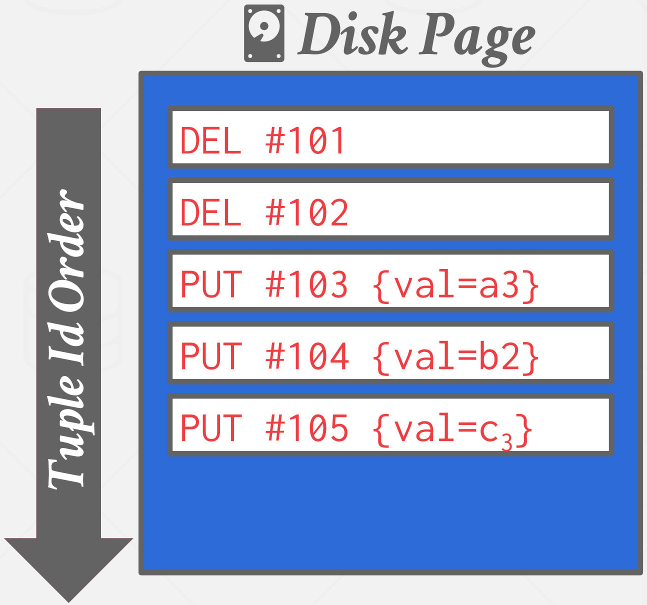
→ Called Sorted String Tables (SSTables)



LOG-STRUCTURED COMPACTION

After a page is compacted, the DBMS does need to maintain temporal ordering of records within the page.
→ Each tuple id is guaranteed to appear at most once in the page.

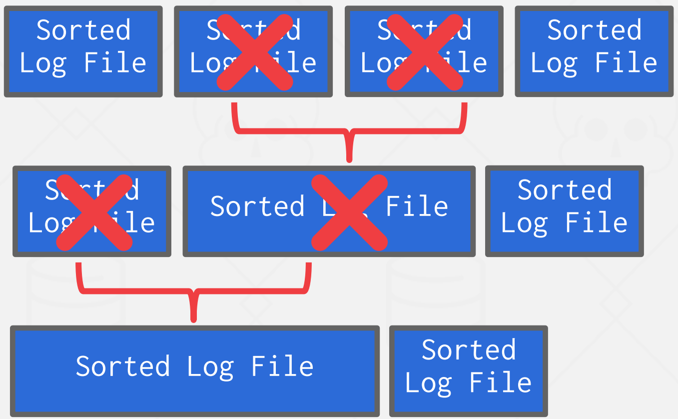
The DBMS can instead sort the page based on id order to improve efficiency of future look-ups.
→ Called Sorted String Tables (SSTables)



LOG-STRUCTURED COMPACTION

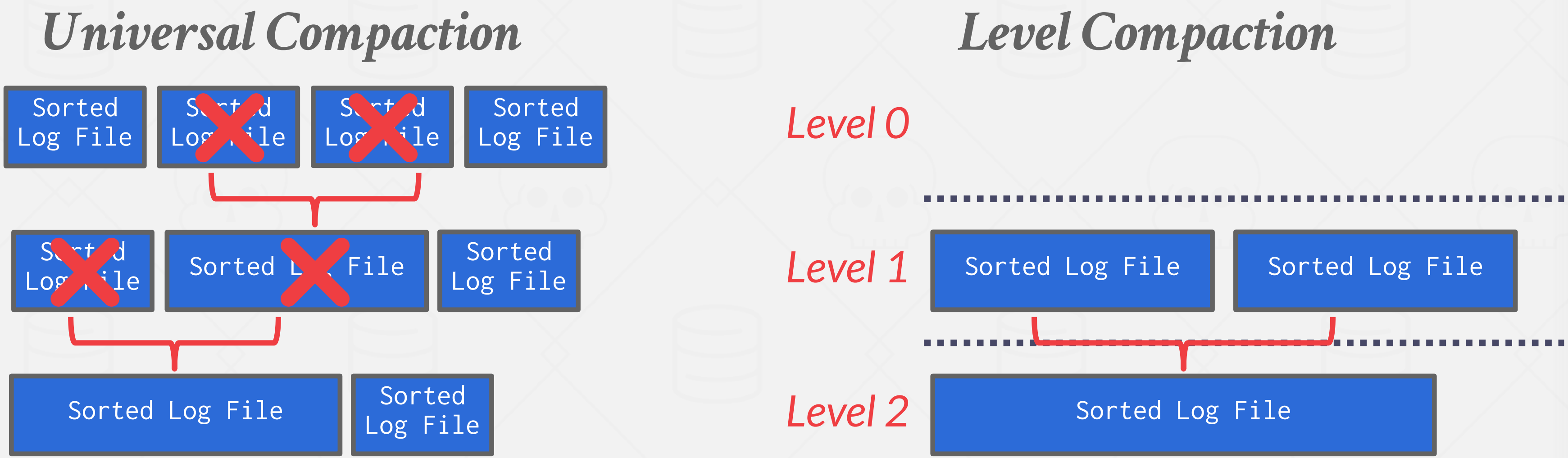
Compaction coalesces larger log files into smaller files by removing unnecessary records.

Universal Compaction



LOG-STRUCTURED COMPACTION

Compaction coalesces larger log files into smaller files by removing unnecessary records.



DISCUSSION

Log-structured storage managers are more common today. This is partly due to the proliferation of RocksDB.

What are some downsides of this approach?

→ Write-Amplification
→ Compaction is Expensive

RocksDB
levelDB
APACHE HBASE
yugabyteDB
fauna
Dgraph
TiDB
CockroachDB
cassandra