09 – Database Compression

1. Compression Background
    A. Trade-off : speed vs. compression ratio
        i. In-memory DB usually choose speed
        (since they don't have disk I/O)
        ii. Disk-oriented DB choose compression radio
        (their performance are dominated by disk I/O)
    B. Real-world dataset tend to be highly skewed.
2. Naïve Compression
    A. DBMS should decompress data before read or update operation occurs to that data.
3. OLAP Columnar Compression
    A. Null Suppression
    Variant of RLE, just compress consecutive NULL to how many there were and where they existed
    B. Run-Length Encoding
    Compress consecutive same value to triplets (value, start position, length)
    not good for alternating pattern of different data.
    C. Bitmap Encoding
    maintain bitmap of unique values to indicate positions of values' position
    not good for dataset which has many unique values.
    D. Delta Encoding
    store the difference between values
    if it scheme combines with RLE, it can be better
    (if gap between value is consecutive)
    E. Incremental Encoding
    variant of Delta Encoding, maintain suffix and prefix length

F. Mostly Encoding
   if most values has less size than largest size, store them as smaller type

G. Dictionary Encoding
   map pattern and smaller code, replace pattern with corresponding code.

   i. Dictionary data structure
      array, hashtable, B+Tree….
      need to consider speed and memory consumption

4. OLTP Index Compression

A. Prefix compression

   i. Extract common prefix and just store suffix of data.
      Manage prefix separately.

B. Suffix truncation

   i. In inner node, do not store entire key,
      just store needed prefix to identify route to actual records

C. Hybrid Index

   i. Maintain two index,
      dynamic index to update records,
      static index to read records,
      merge dynamic index to static index to maintain updated indexes.
      use bloom filter to choose which index to use to read records.