Lec 09 Data Compression

对于Disk-Oriented DB，IO操作成为瓶颈。

对于In-Memory DB，关键trade-off在于 **speed** 与 **compression ratio**

所以关键在于减少DRAM的处理开销，In-Memory通常倾向于追求速度，一些性能比较好的压缩算法可能非常 **computationally expensive**

Goal

- Goal1

Must produce **fixed-length** values.（Exception is var-length data）

- Goal2

late materialization 在执行query时候尽可能推迟decompression

- Goal3

Lossless 不能损失数据.

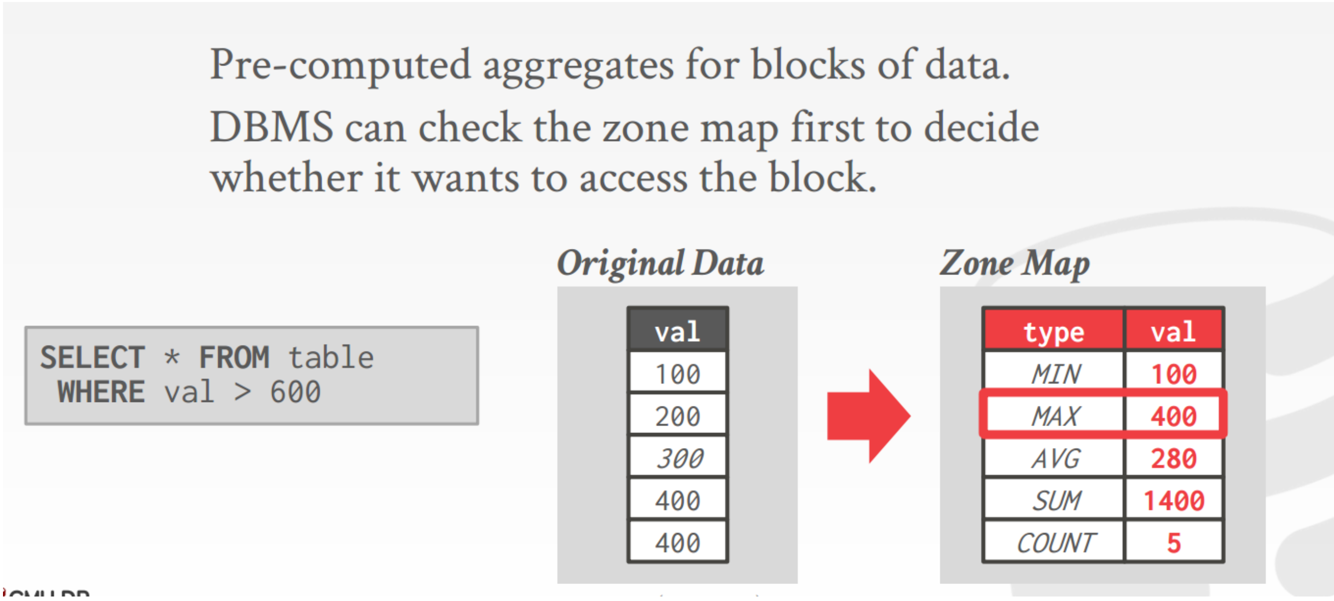
DATA SKIPPING

- Approach1: **Approximate Queries**

Execute queries on sampled subset of the entire table to produce **approximate results.**

- Approach2: **Zone Maps**

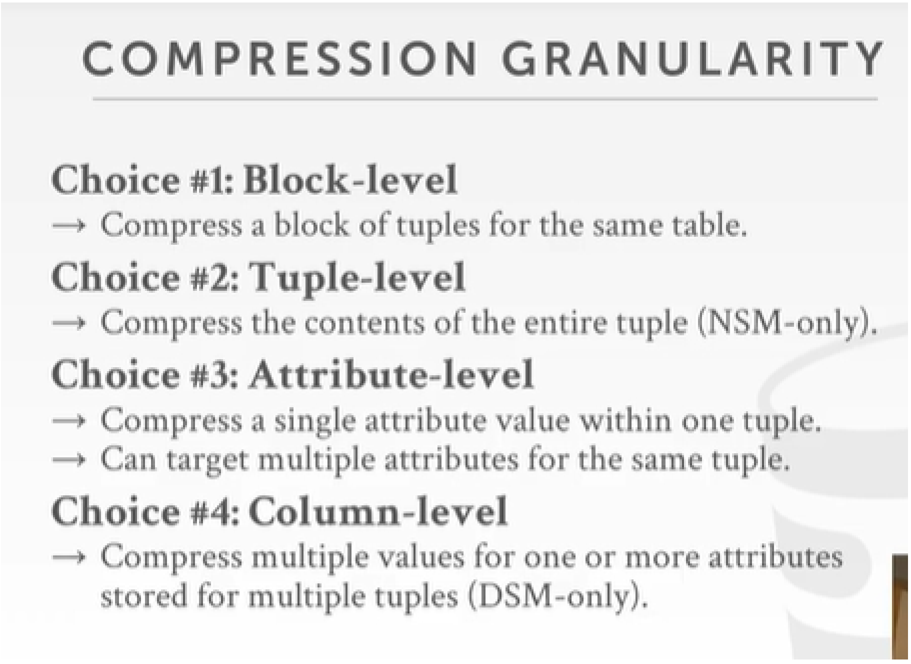
Pre-compute columnar aggregations per block that allow DBMS to check whether queries need to access it.



大意是对某个block的data计算其aggregation function,如MIN,MAX,AVG,SUM,COUNT,记录在Zone Map上。

然后query到达的时候，先检查zone map看这个block是否有符合其要求的数据。如果没有的话直接skip掉这个block。例如上面的语句val > 600， 然而MAX = 400,说明这个block不可能包含query所需要的数据。

压缩的粒度



NAIVE COMPRESSION

考虑常规的数据压缩算法snappy,gzip等

考虑因素

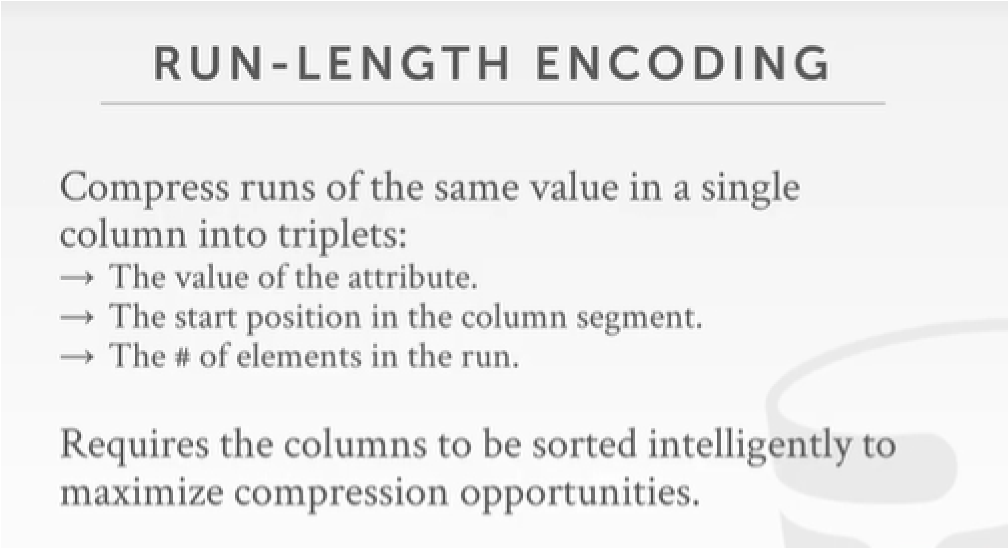
- 计算开销
- 压缩/解压速率

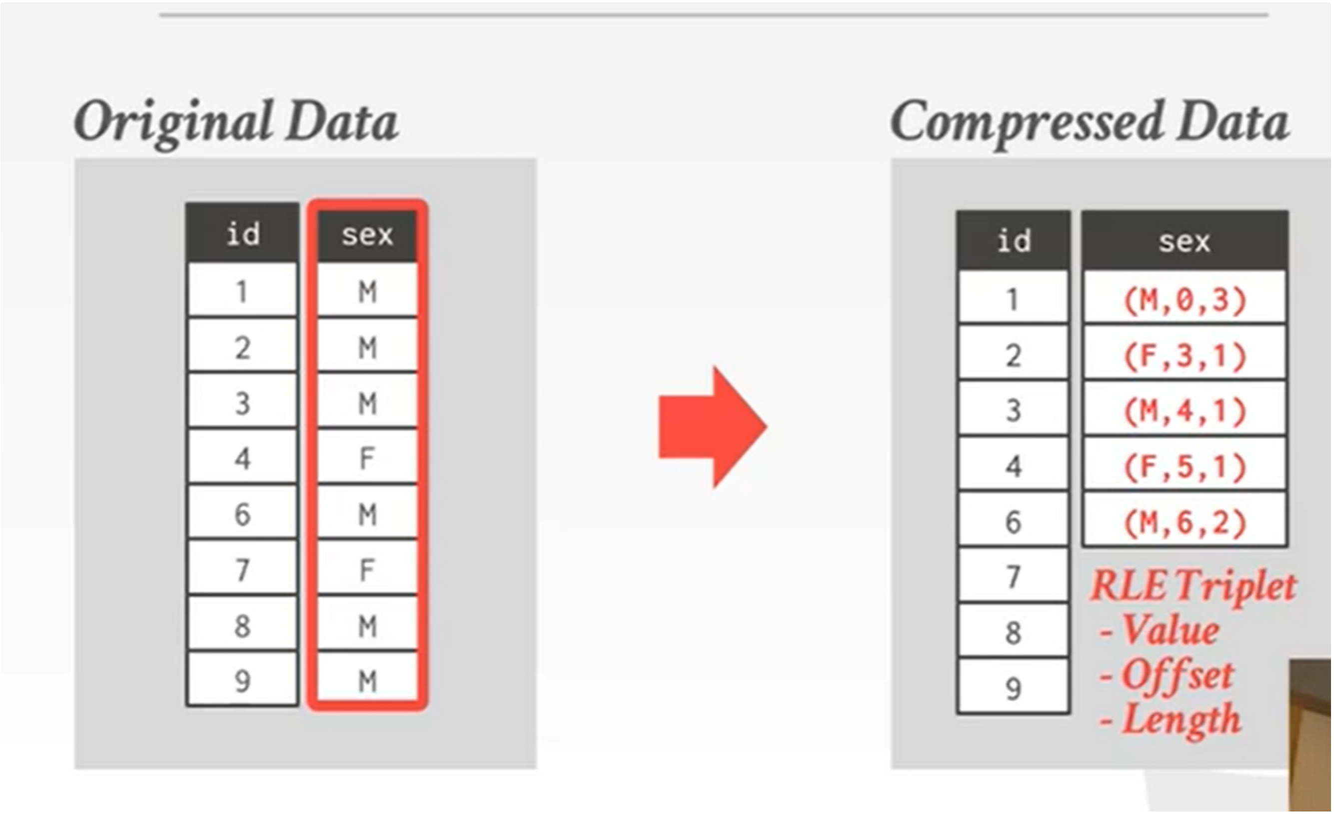
COLUMNAR COMPRESSION

Null Suppression

连续的0或者空白用**这个位置有多少个0或者空白来代替**

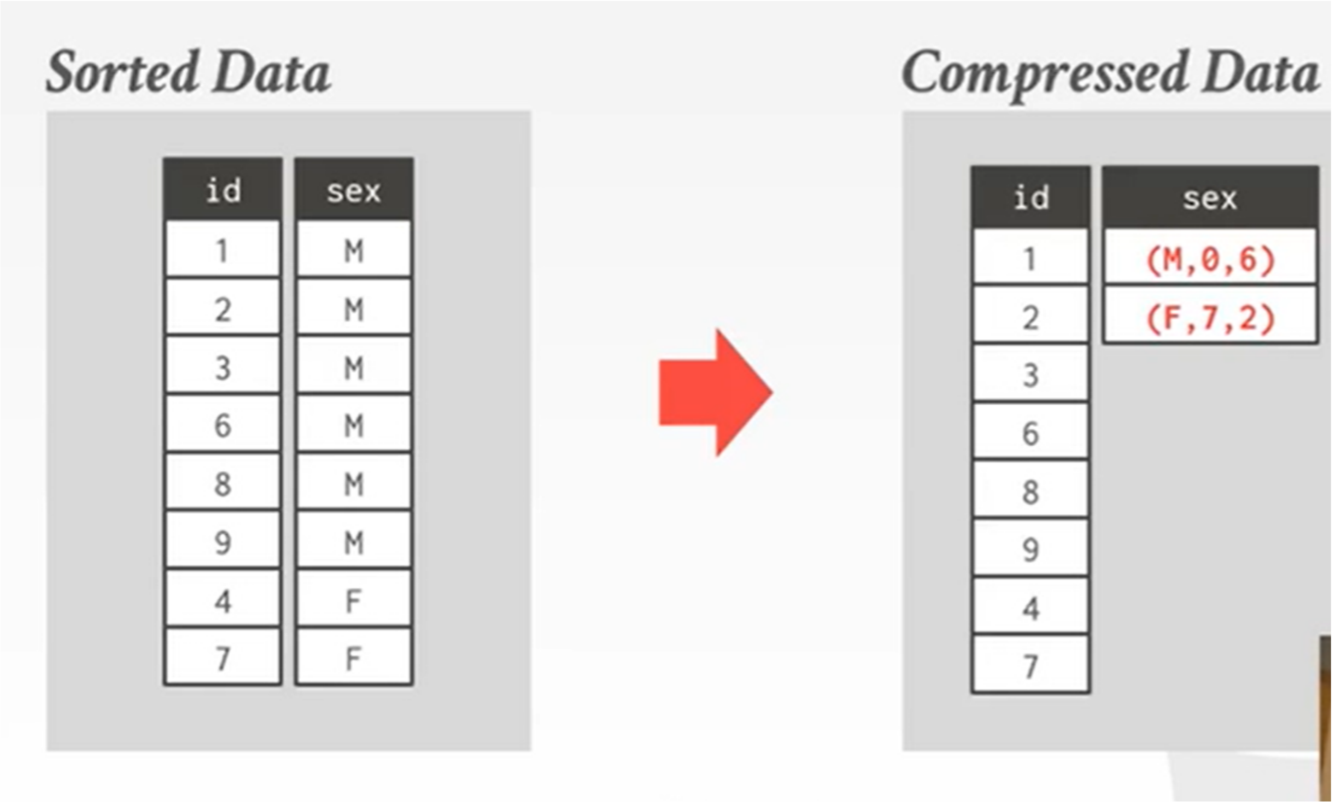
Run-length Encoding





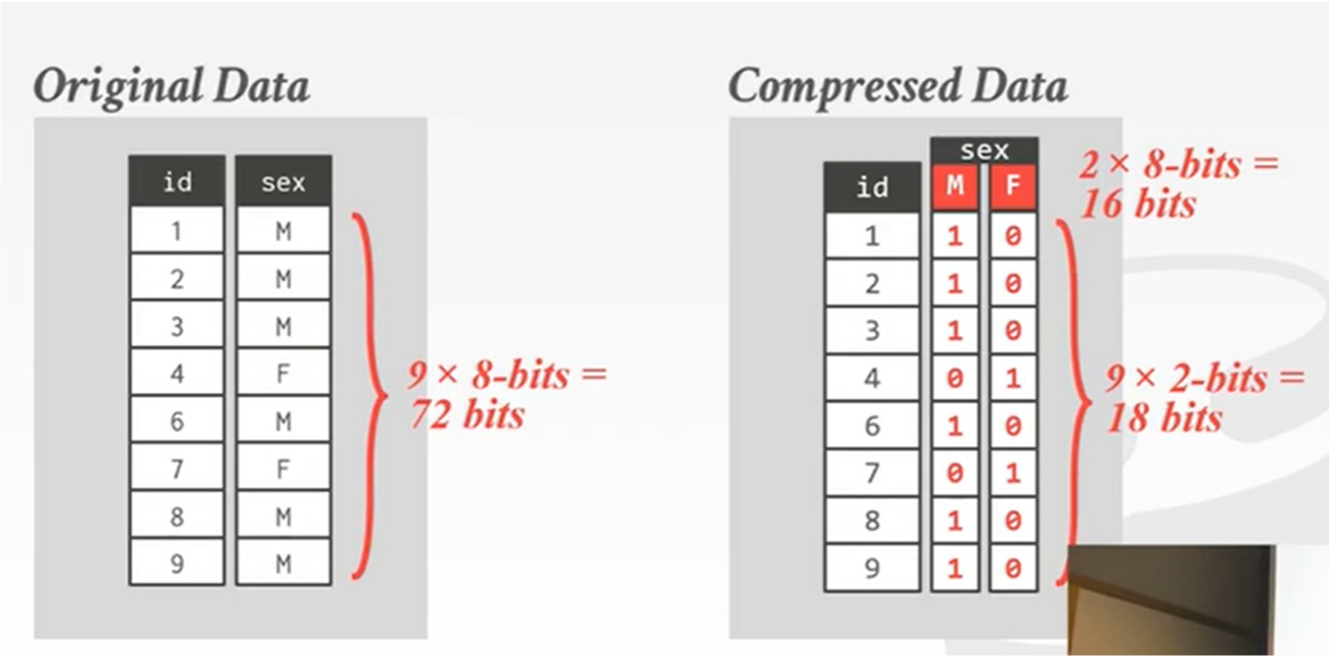
由上图所示，可以压缩成（value,offset,length）

After sorted: 压缩效果更好



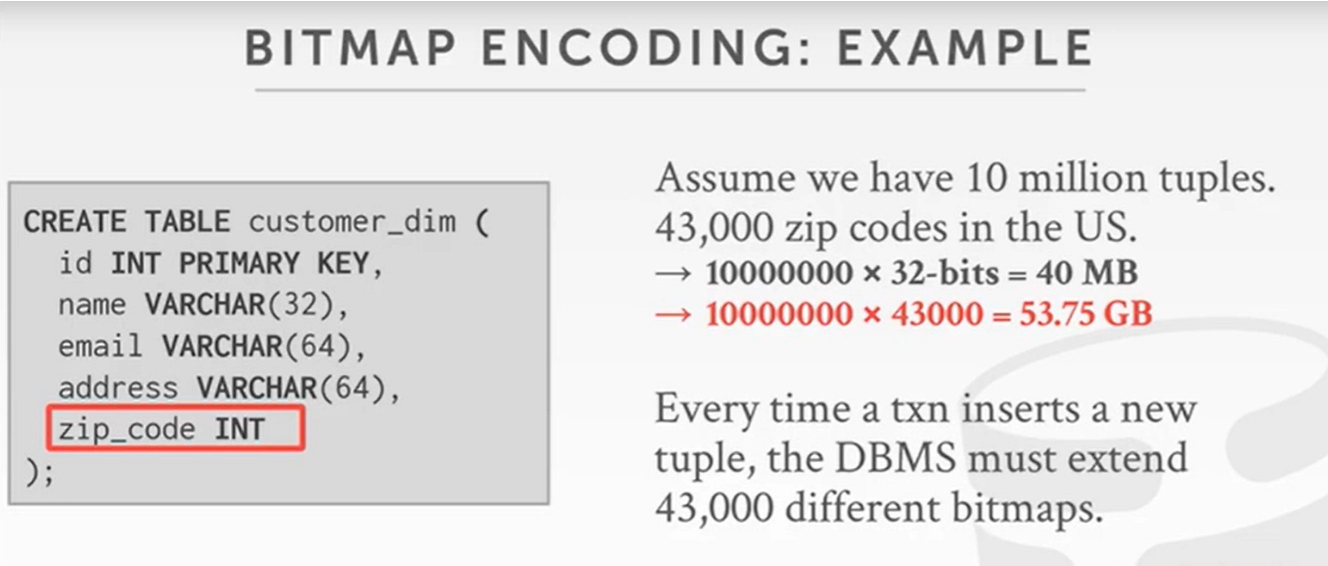
Bitmap Encoding

Only a good idea when the cardinality is small.



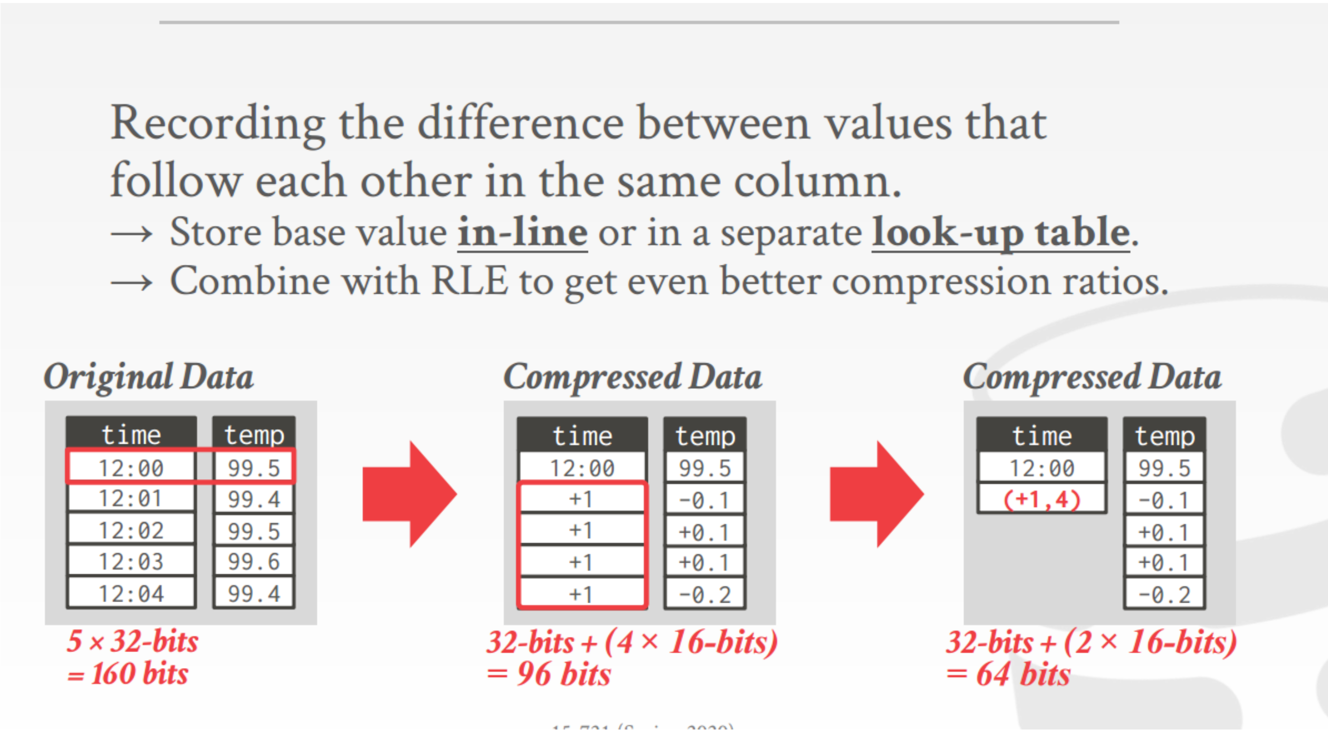
With the following circumstances:

THIS IS TERRIBLE:

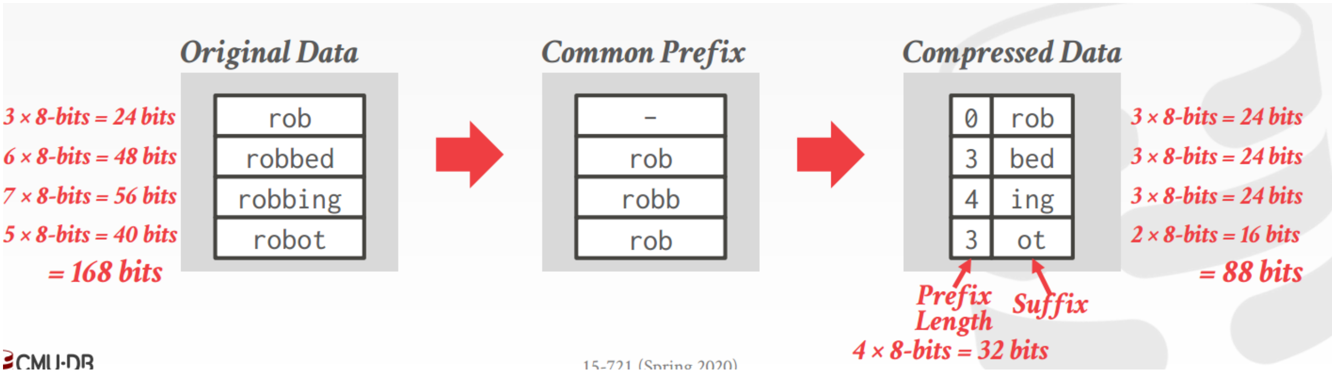


Delta Encoding

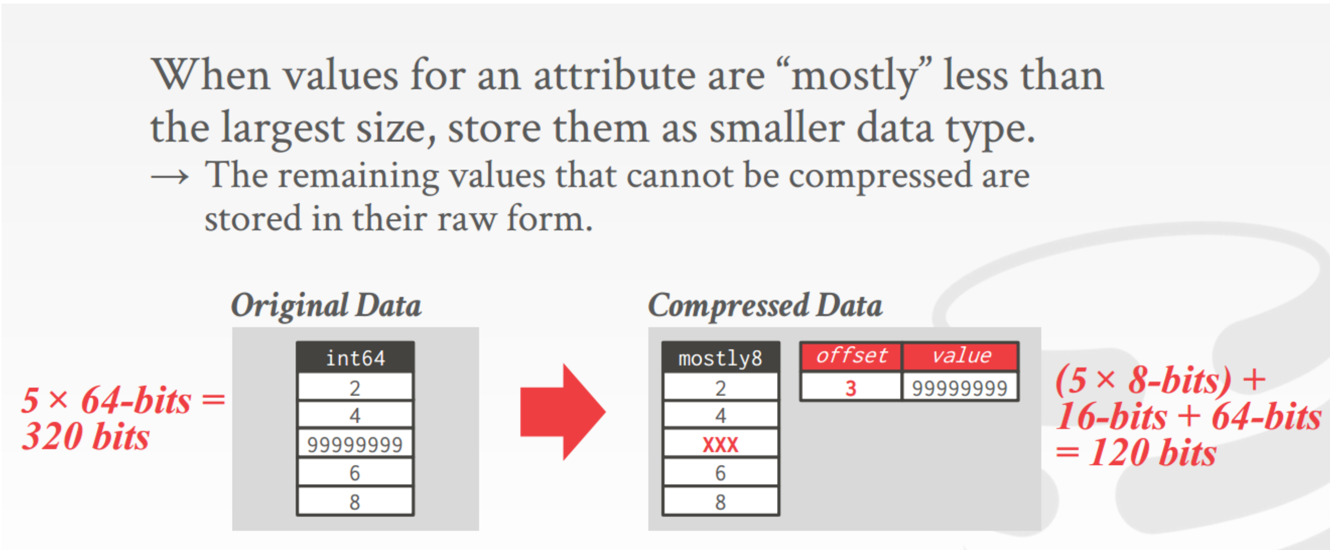
适用于变化比较小的value的压缩，可以结合run-time encoding 达到更好效果



Incremental Encoding



Mostly Encoding



Dictionary Encoding

Choice #1: Block-level

→ Only include a subset of tuples within a single table.

→ Potentially lower compression ratio, but can add new tuples more easily.

Choice #2: Table-level

→ Construct a dictionary for the entire table.

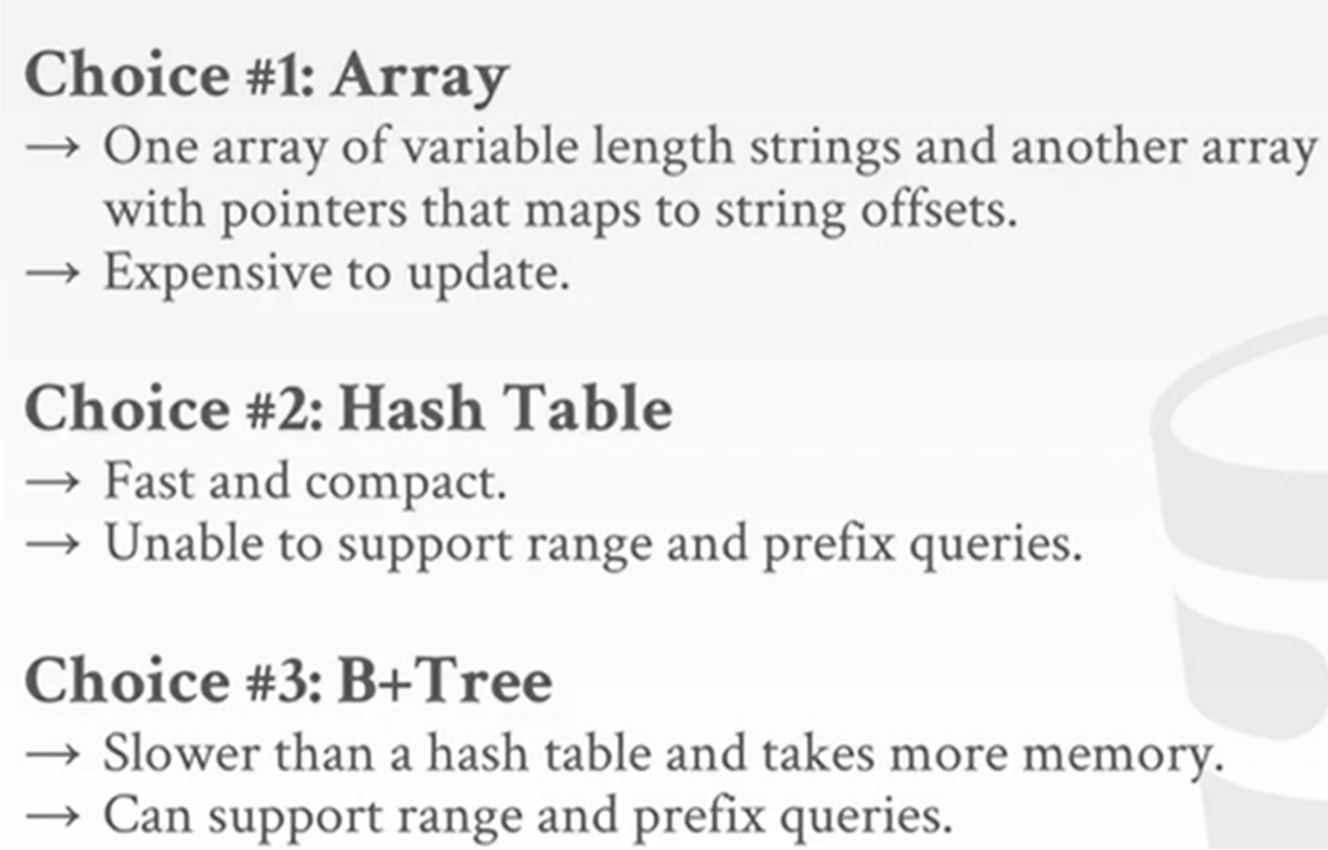
→ Better compression ratio, but expensive to update.

Choice #3: Multi-Table

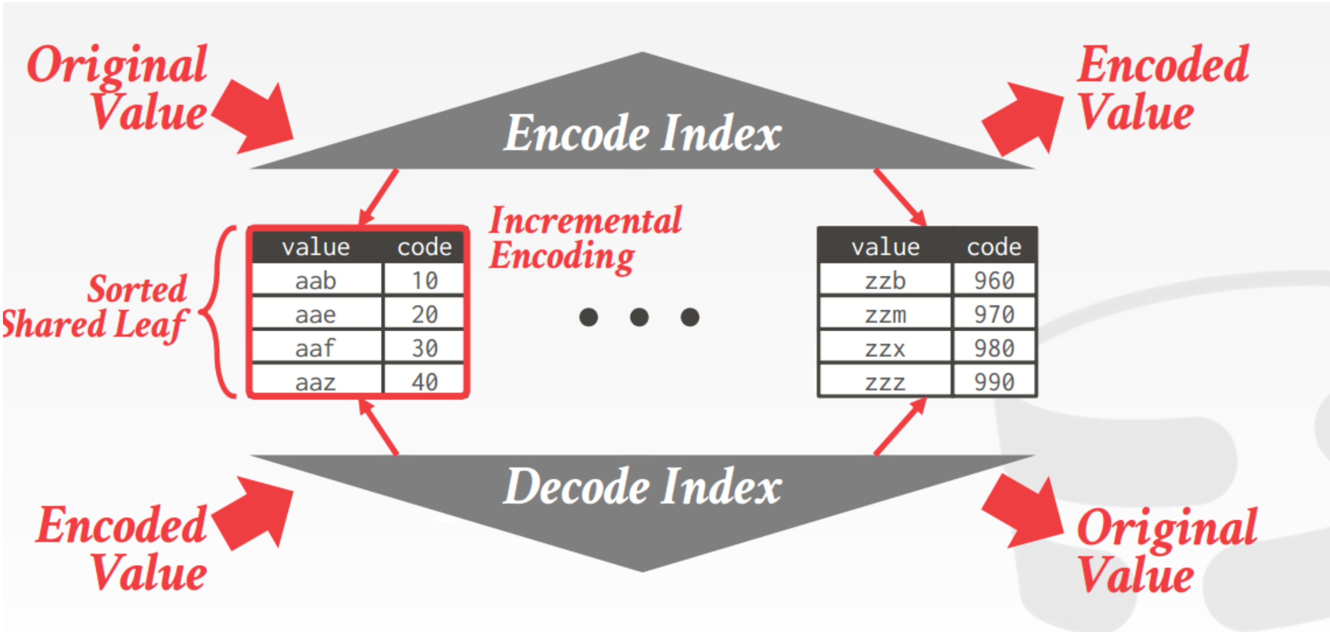
→ Can be either subset or entire tables.

→ Sometimes helps with joins and set operations.

- Data Structures



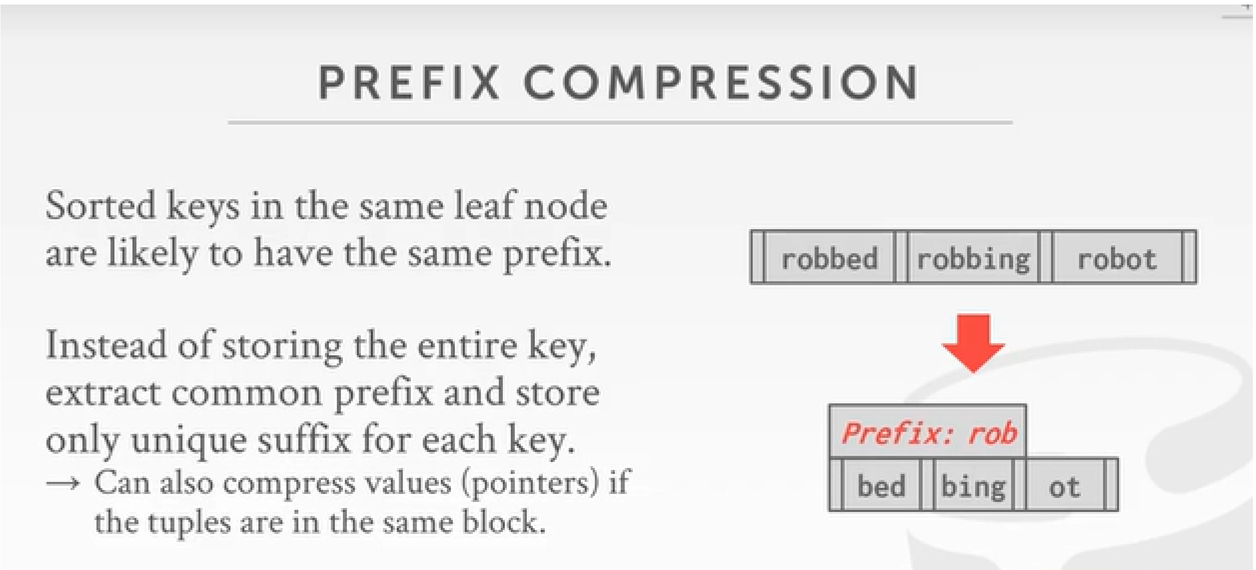
- B+Tree Shard Index



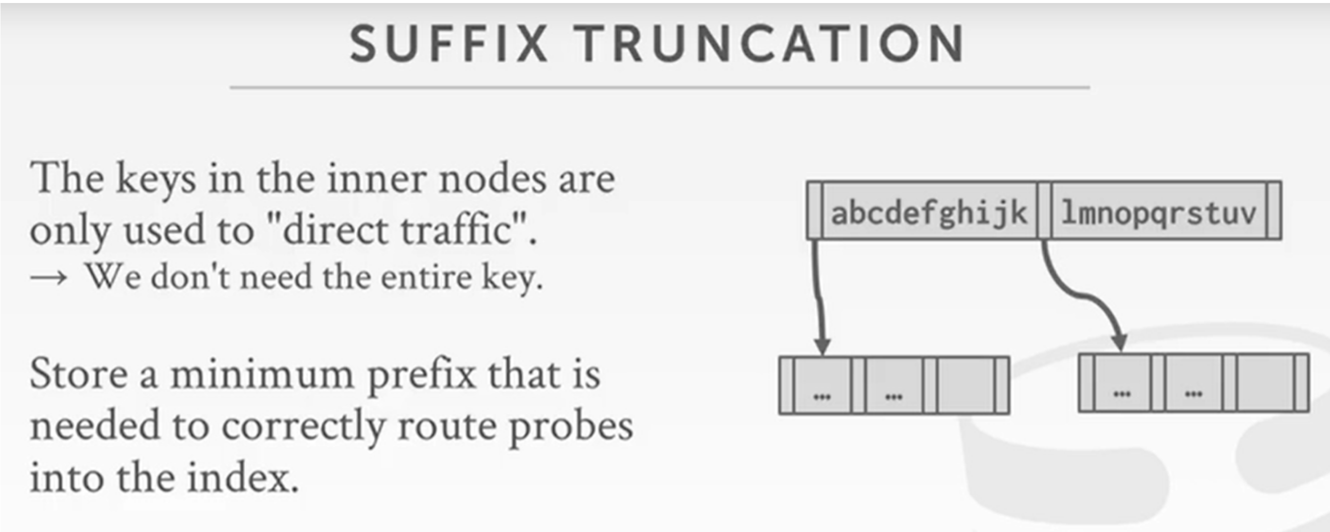
到此为止。我们讨论的所有压缩trick都只是适用于OLAP，OLTP是不能够使用这些trick的，因为OLTP需要支持快速的随机访问。而在OLTP中索引占用了很大一部分的内存开销。

接下来讨论OLTP的压缩trick

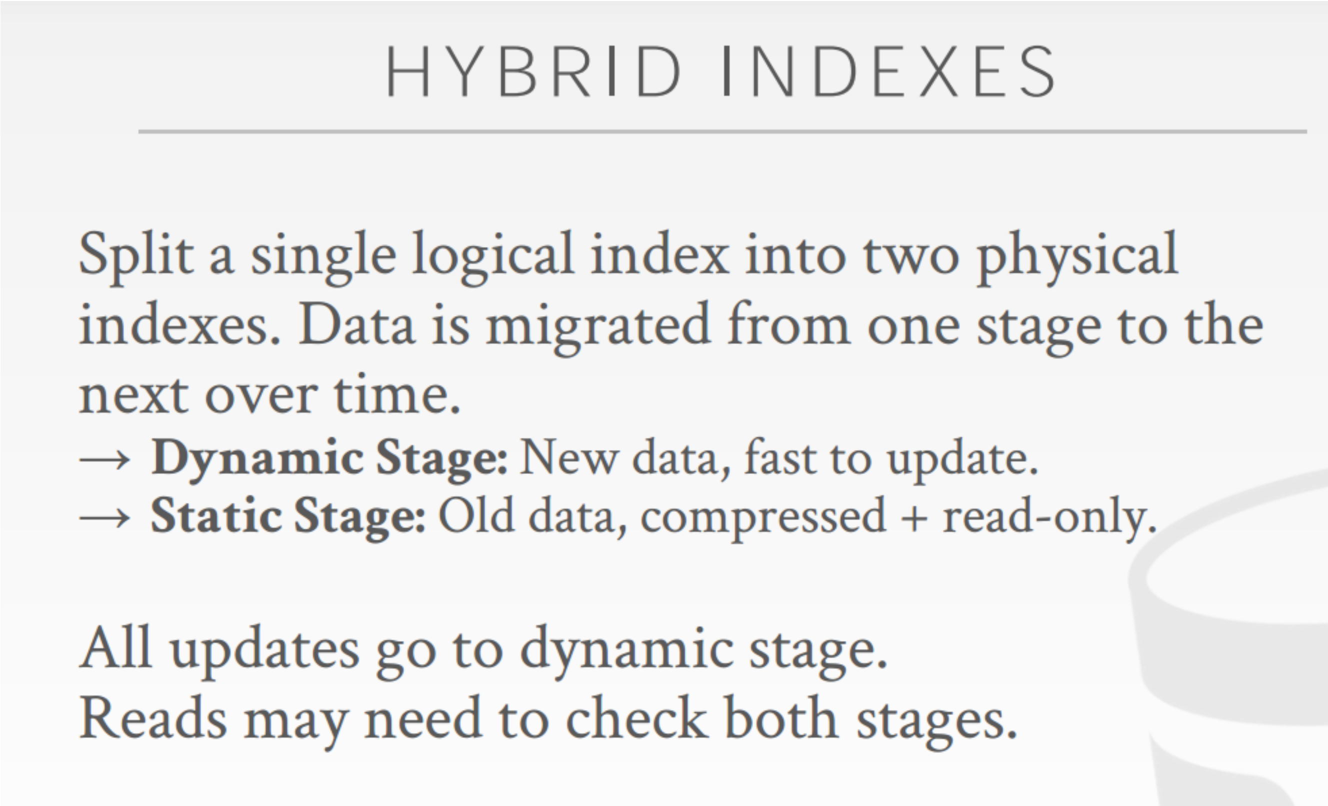
- Prefix Compression 压缩前缀

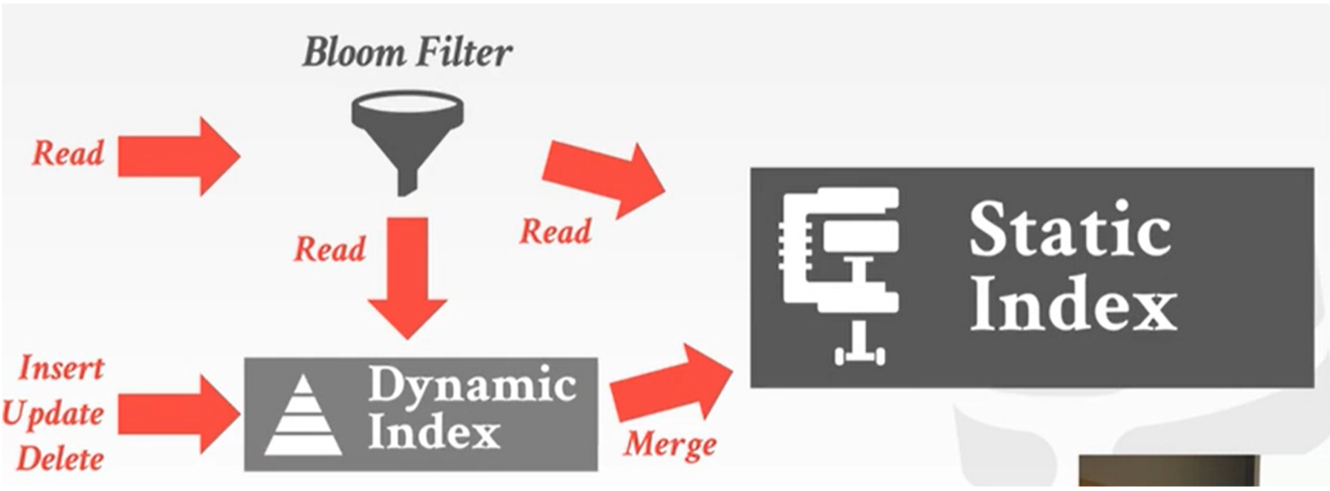


- Suffix Truncation 后缀截断

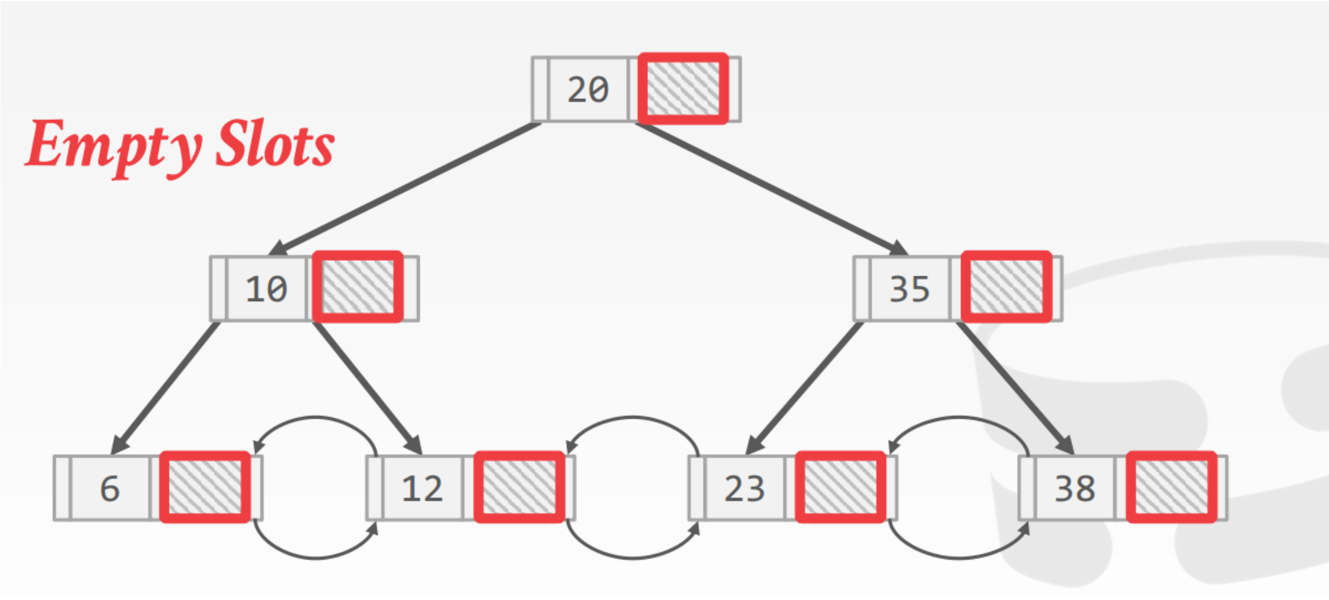


- Hybrid Index：

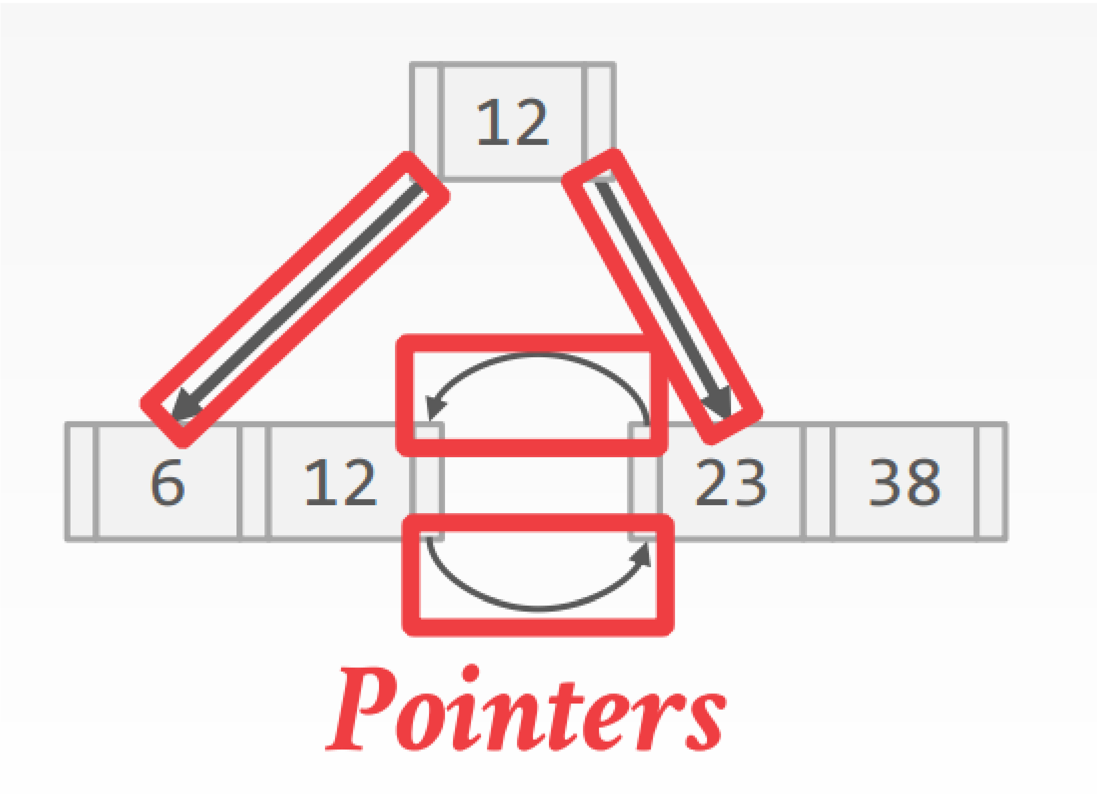


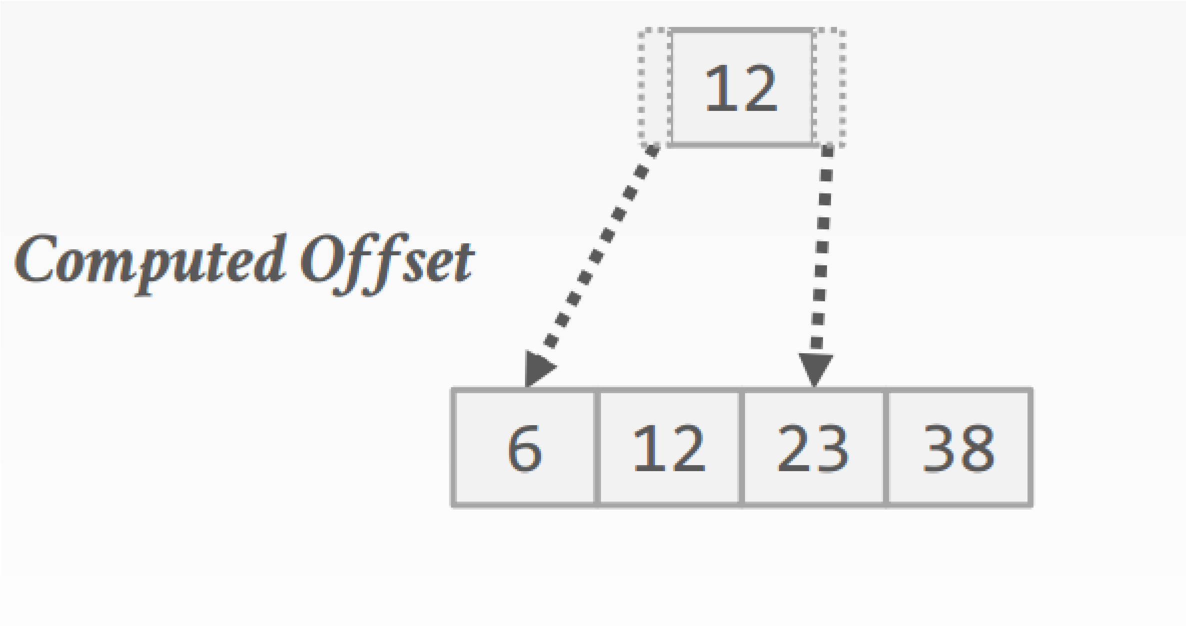


- Compaction of B+Tree



remove the empty slots, and instead of saving the pointers, you can just save the computed offset.




12
Computed Offset
6
12
23
38