OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG INF FAKULTÄT FÜR INFORMATIK

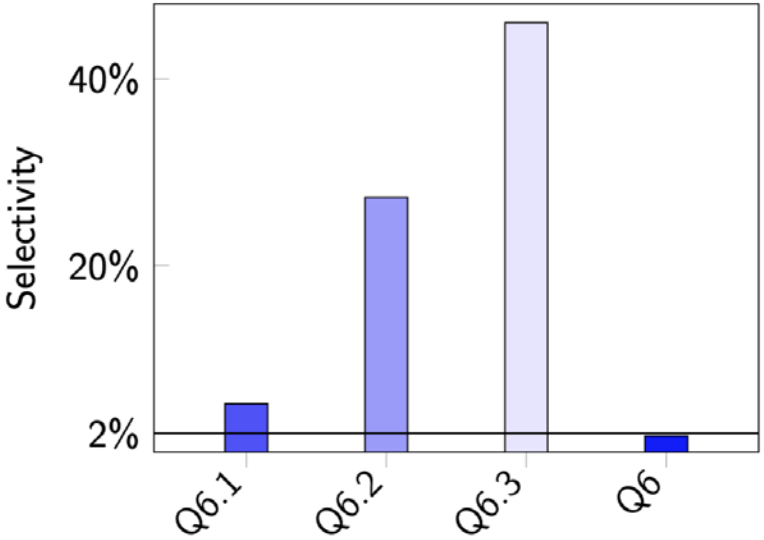
Elf

— Multi-Dimensional Tree Structure —

David Broneske, Martin Schäler, Veit Köppen

Multi-Column Selection Predicates

- Single scans for each predicate reasonable
  - No B-Tree would beat it!
- Combined selectivity pretty high
  - Case for a multi-dimensional index structure?



Q6
- Q6.1 l_shipdate >= '[DATE]' and l_shipdate < '[DATE]' + '1' year
- Q6.2 and l_discount between [DISCOUNT] − 0.01 and [DISCOUNT] + 0.01
- Q6.3 and l_quantity < [QUANTITY]

TPC-H Query Q6

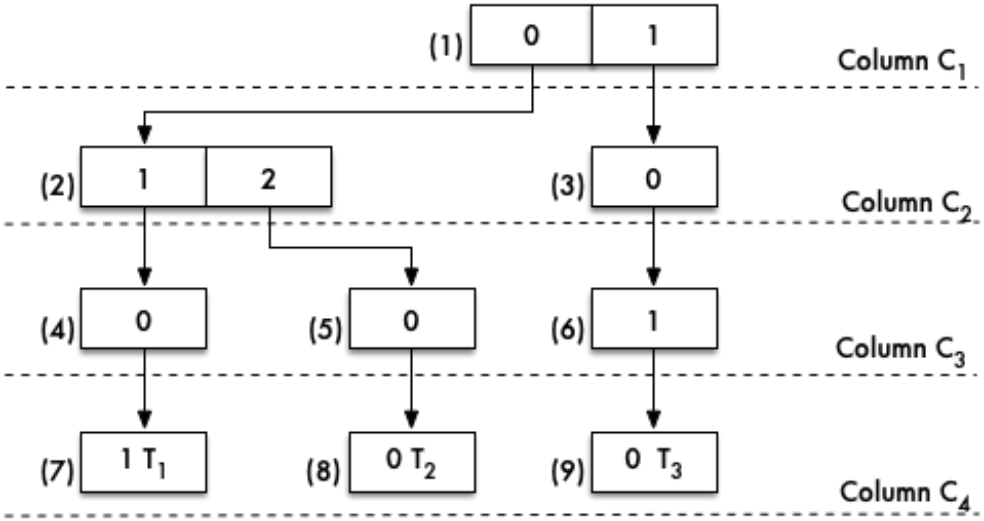
Conceptual Design of Elf [BKSS17]

- Tree-based index structure
- Level = data of one column
- Nodes (`DimensionLists`) consist of
  - Value
  - Pointer
- Partitioning creates an N-1 dimensional subspace

Table

	C_1	C_2	C_3	C_4
T_1	0	1	0	1
T_2	0	2	0	0
T_3	1	0	1	0

Elf



Elf — Construction

Procedure:

1. Sort data on current column C_i

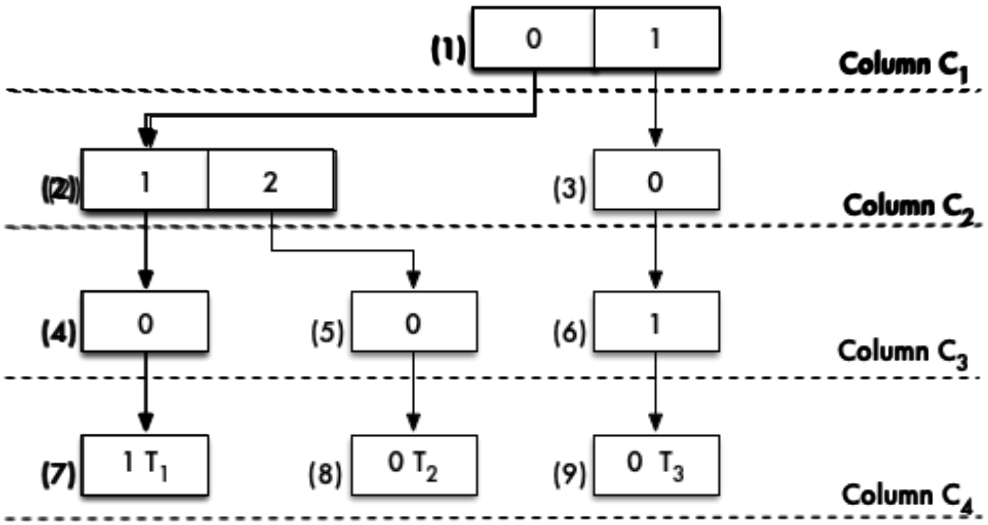
2. Create node entry in `DimensionList` for each distinct value.

3. Repeat from step 1 for each logical partition (subset) with C_{i+1}

Table

	C_1	C_2	C_3	C_4
T_1	0	1	0	1
T_2	0	2	0	0
T_3	1	0	1	0

Elf



Elf — Properties

Prefix-Redundancy Elimination

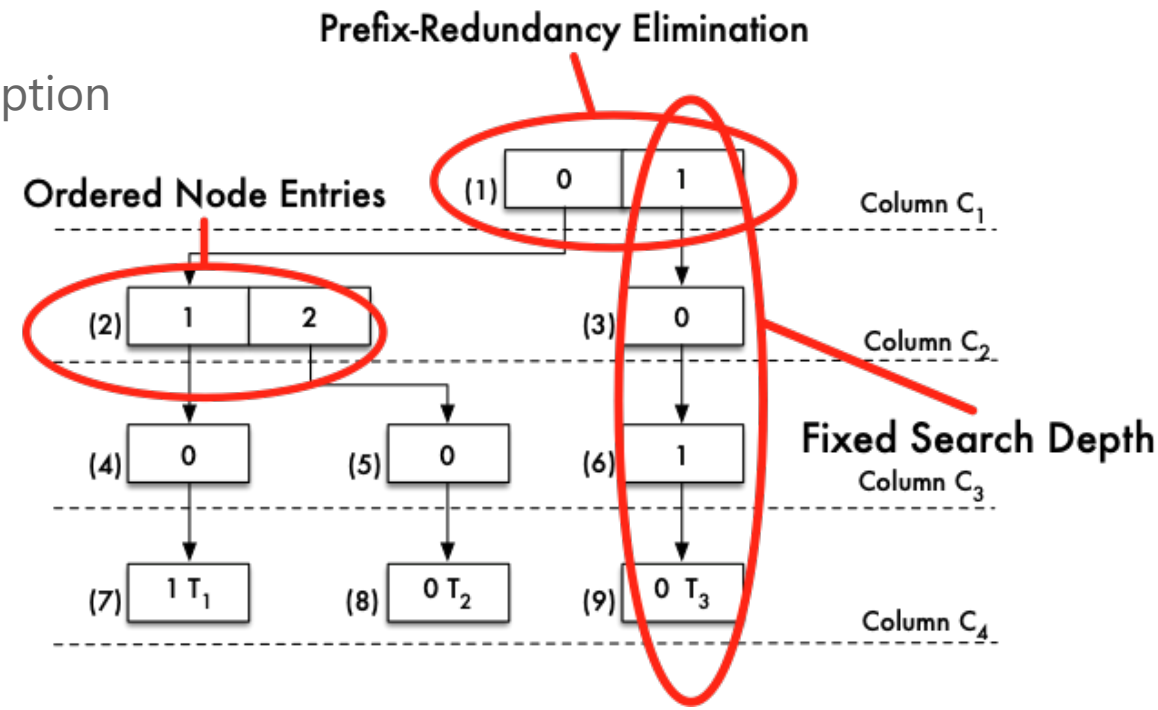
→ Reduced storage consumption

Ordered Node Entries

→ Early pruning possible (e.g. C2 = 0)

Fixed Search Depth

→ Limits hops, no recursion



Optimizations: MonoLists, Linearization, HashMap

Elf Optimizations: `MonoList`

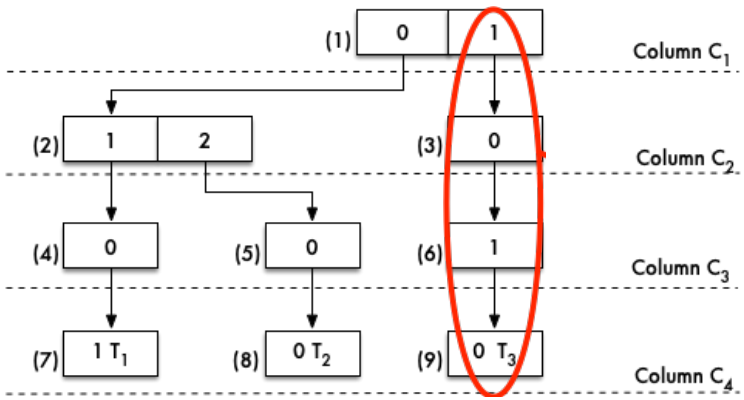
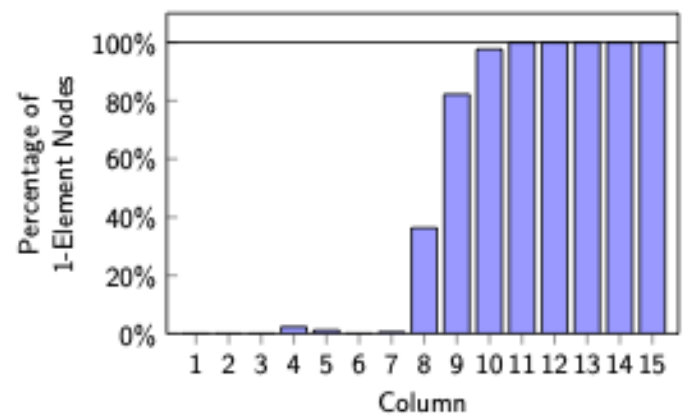


Figure: Standard Elf

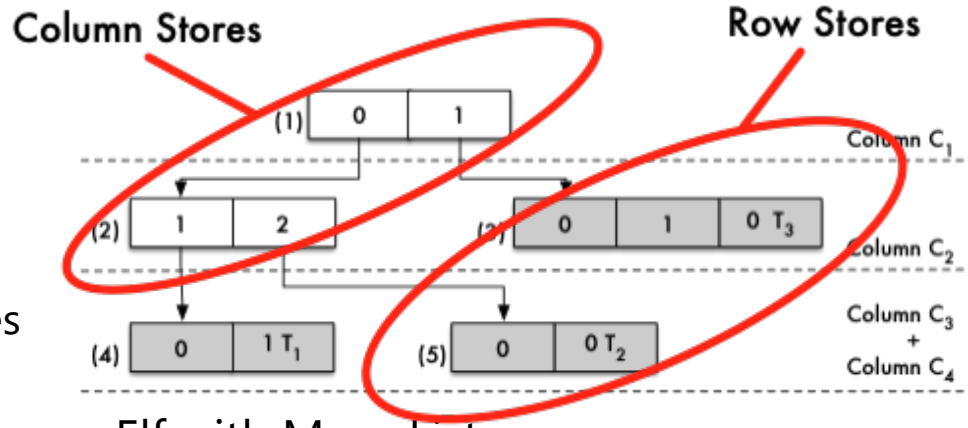


Solution:

- 1-Element nodes share common suffix
- Create MonoList for all paths w/o branching in underlying dimension

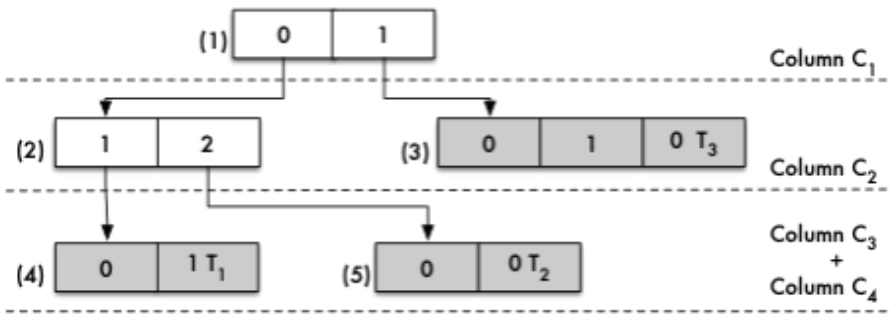
Column-wise Nodes
Represent data of one column for several tuples

Row-wise Nodes
Represent data of a single tuple across several columns

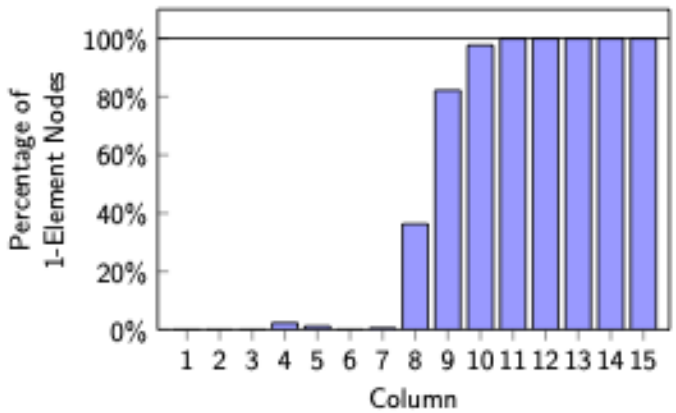


Elf with MonoLists

Elf Optimizations: `MonoList` cont.

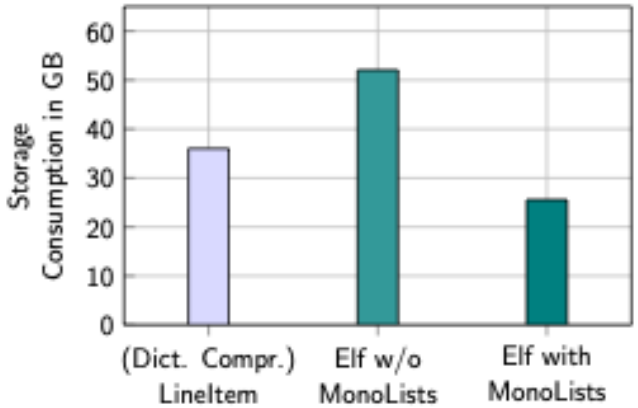


Elf with MonoLists

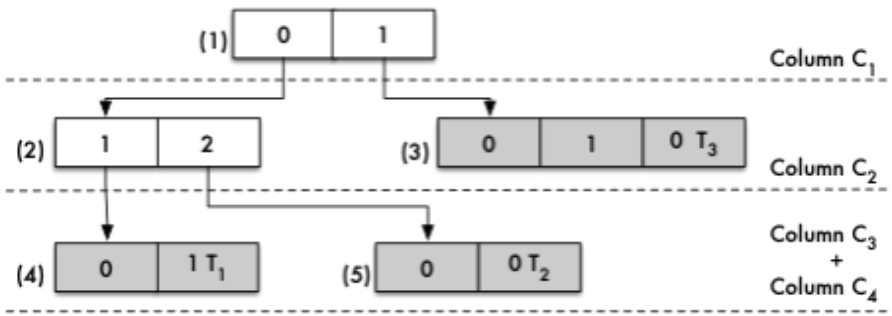


Result:

- → MonoLists half the storage comparison of Elf
- → 70% storage space of original data



Elf Optimizations: Linearization



Elf with MonoLists

Idea:

- → Transform pointer addresses to array offsets
- → Reduces cache misses and storage consumption

Linearized Elf

- Pairs of value, pointer
- MSB set at value → List ends
- MSB set at pointer → Child is MonoList

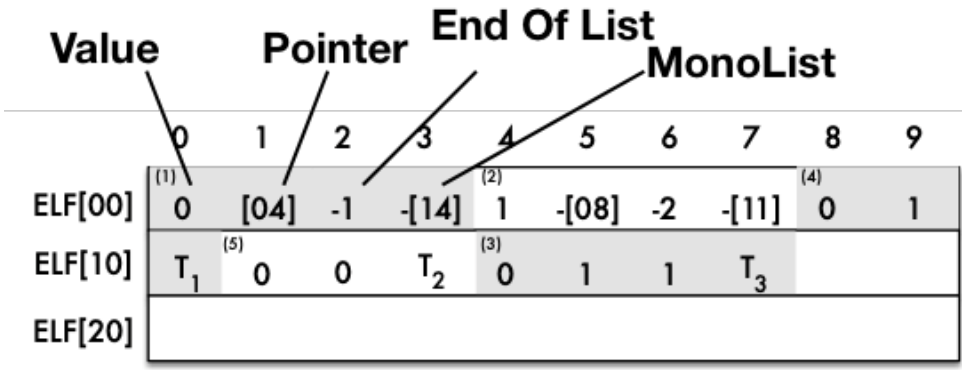
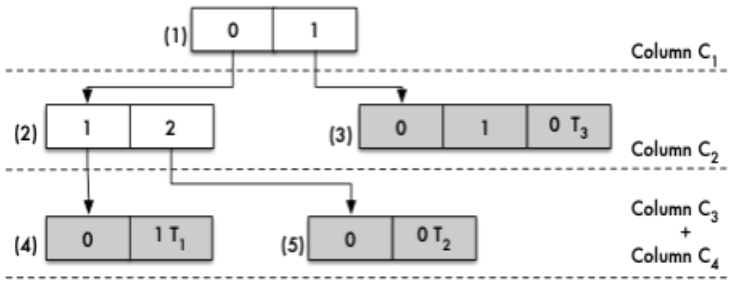


Figure: Linearized Elf

Elf Optimizations: Linearization cont.



Elf with MonoLists

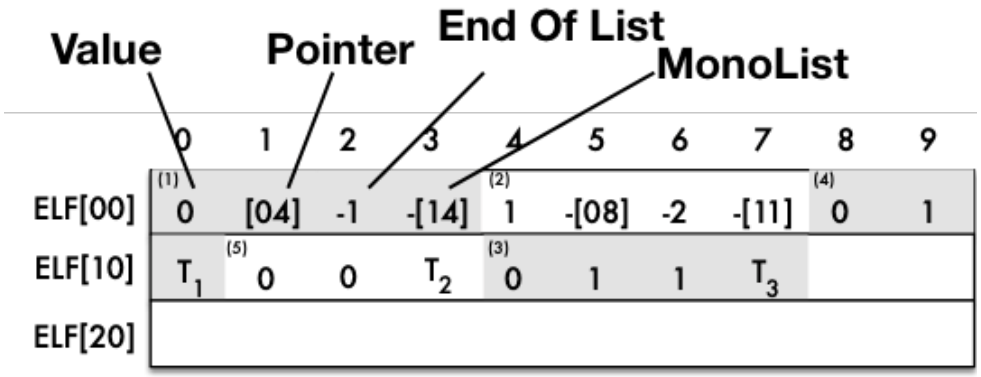


Figure: Linearized Elf

Solution - Pre-Order Traversal

Recursively:

1. Create current `DimensionList`, remember pointer address
2. For each child
    1. If child is `DimensionList`: Go to Step 1
    2. If child is `MonoList`: Write `MonoList`, propagate pointer, visit next child
3. Propagate pointer of `DimensionList` to parent

Elf Optimizations: HashMap

Observation — Values in first dimension are

1. **Unique:** Due to prefix-redundancy elimination
2. **Ordered:** Due to ordered DimensionList entries
3. **(Maybe) Dense:** E.g., by dictionary compression

→ Store pointers only!
(Position of pointer encodes value belonging to it)

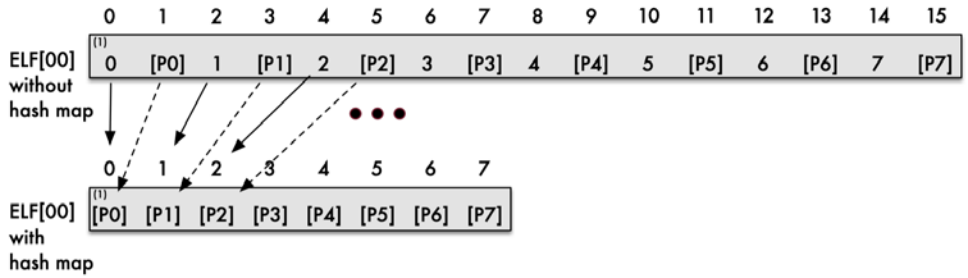
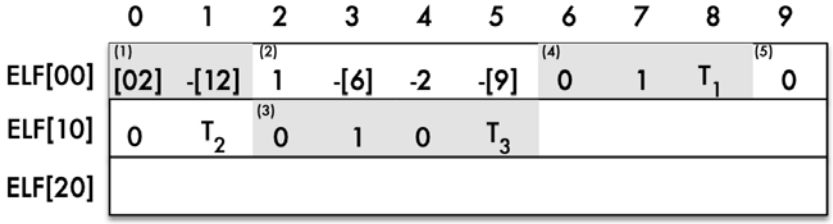


Fig. 4. Hash-map property of the first `DimensionList`

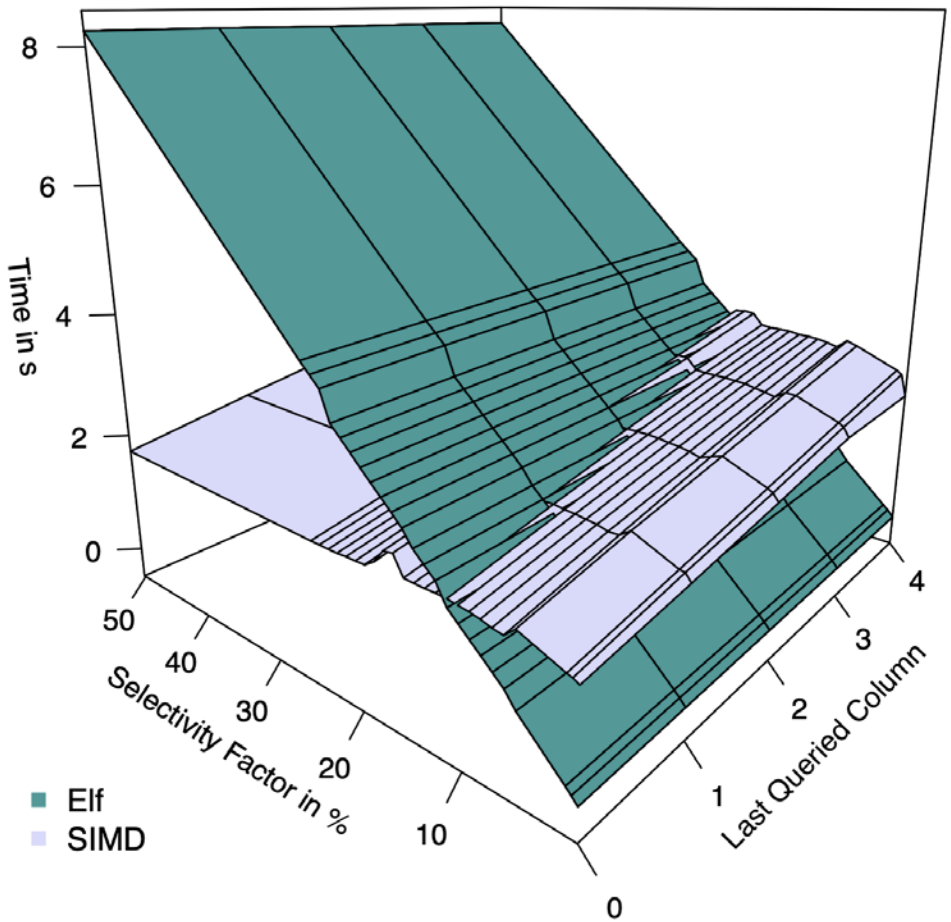


. 7. Optimized memory layout

Elf — Micro Benchmark

Number of Columns vs. Selectivity:

- Dominant impact factor: selectivity
- Threshold SIMDScan vs Elf:11-18%



Elf - Summary

- Efficient for highly selective workloads
- The more columns are queried, the better the selectivity threshold
- Elf scales linear with a factor smaller than 1

Challenges:

- Complex Predicates
- Insertions
- Column order

References

D. Broneske, V. Köppen, G. Saake, and M. Schäler, “Accelerating multi-column selection predicates in main-memory - the Elf approach,” in ICDE. IEEE, 2017, pp. 647–658.

Y. Sismanis, A. Deligiannakis, N. Roussopoulos, and Y. Kotidis, “Dwarf: Shrinking the PetaCube,” in SIGMOD. ACM, 2002, pp. 464–475.

H. Zhang, D. G. Andersen, A. Pavlo, M. Kaminsky, L. Ma, and R. Shen, “Reducing the storage overhead of main-memory OLTP databases with hybrid indexes,” in SIGMOD. ACM, 2016, pp. 1567–1581.

D. Das, J. Yan, M. Zait, S. R. Valluri, N. Vyas, R. Krishnamachari, P. Gaharwar, J. Kamp, and N. Mukherjee, “Query optimization in Oracle 12c database in-memory,” VLDB, vol. 8, no. 12, pp. 1770 – 1781, 2015.

Sikka, V., Färber, F., Lehner, W., Cha, S. K., Peh, T., & Bornhövd, C. (2012, May). Efficient transaction processing in SAP HANA database: The end of a column store myth. In SIGMOD, pp. 731-742, ACM.