Columnar Storage and List-based Processing for Graph Database Management System

Pranjal Gupta, Amine Mhedhbi, Semih Salihoglu

UNIVERSITY OF WATERLOO · Data Systems Group · Graphflow

COLUMNAR RDBMSs

- Read-optimized systems, optimized for read-heavy workloads that involves reading and processing large chunks of data *sequentially*.

Storage

	PERSON. name	PERSON. age	PERSON. city
p4	`bob`	54	TO
p2	`jenny`	23	TO
p3	`alice`	45	VC
p1	`peter`	17	VC

Compression

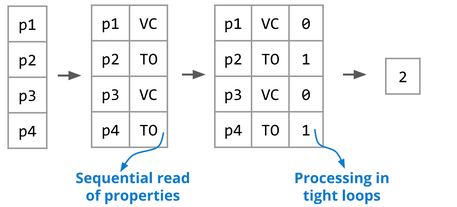
0011_b VC=0, TO=1

Dictionary-encoded PERSON.city

Block-based Processing

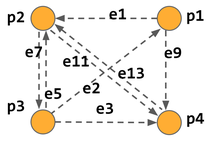
```
SELECT count(*)
FROM PERSON p
WHERE p.city = `TO`
```

SCAN p → SCAN p.city → FILTER p.city = `TO` → COUNT



GDBMSs and GRAPH DATA

- Read-optimized systems, performant on many-to-many joins.
- ***Adjacency Lists for Fast Joins***



p1	e1,p2	e9,p4
p2	e7,p3	e11,p4
...	...	...

Join happens in O(1) as compared to O(logn) in RDBMS.

- ***Random Data Access***

```
MATCH (a:Person)-[e:Knows]->(b.Person)
WHERE a.name = 'alice'
RETURN b.age, e.since
```

SCAN a → FILTER a.name = `alice` → JOIN b → SCAN b.age, e.since

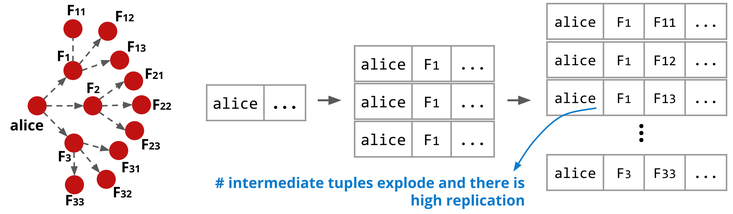
Reading properties is random

	PERSON.age
p4	54
p2	23
p3	45
p1	17

	KNOWS.since
...	...
e2	1992
e1	2015
e3	2003
e13	1999
e5	2009
...	...

p3 'alice' → e2,p1 | e3,p4 | e5,p2

- ***Prominence of many-to-many relationships in graph data***



- ***Partial Structuredness of graph data***
  - EDGE LABEL → SRC VERTEX LABEL and DEST VERTEX LABEL
  - EDGE LABEL → Cardinality of edges
  - EDGE/VERTEX LABEL → Properties keys

SINGLE-INDEXED EDGE PROPERTY PAGES

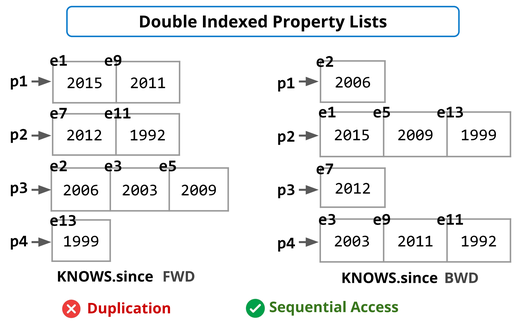
- ***Requirement:*** Store and access the properties of edges sequentially in the order edges appear in the adjacency lists.

Edge Columns

	KNOWS.since
...	
e1	2015
e3	2003
e13	1999
e5	2009
e9	2005
...	

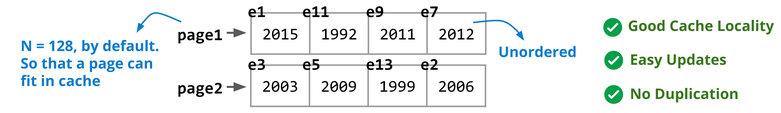
Random Access · Easy Updates · No Duplication

Double Indexed Property Lists



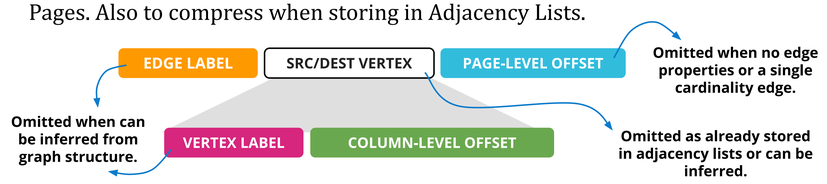
Duplication · Sequential Access

- ***Optimization:*** Group *n* single-directional property lists into a property page.



NEW ID SCHEME AND COMPRESSION

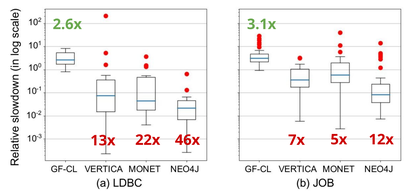
- ***Requirement:*** To support random access to properties in Columns & Property Pages. Also to compress when storing in Adjacency Lists.



EVALUATION

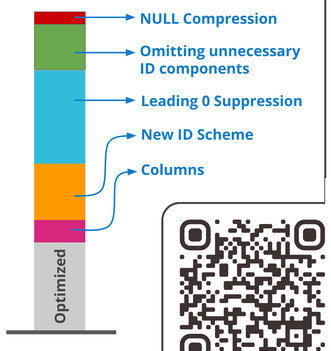
Performance

- ***Dataset:*** LDBC SNB (s.f. 100), IMdB dataset
- ***Queries:*** LDBC Interactive Short Reads (7), LDBC Interactive Complex (12) JOB Benchmark (33)

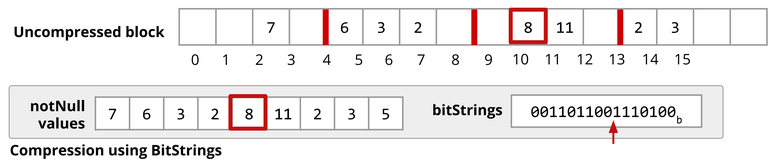


Relative to Vanilla GraphflowDB

Scalability

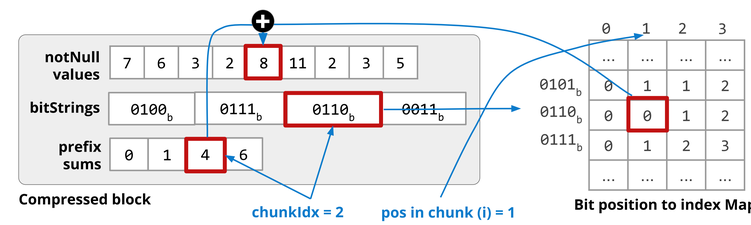


NULL COMPRESSION



- ***Requirement:*** Ability to randomly access from compressed block.

Null Compression with Jacobsons' Bit-vector Index



LIST-BASED PROCESSOR

- ***Block-based Processor:*** Not optimized for many-to-many joins

```
MATCH (a:PERSON) - [r1:FOLLOWS] → (b:PERSON)
      (b:PERSON) - [r2:FOLLOWS] → (c:PERSON)
WHERE a.age > 20
RETURN ...
```

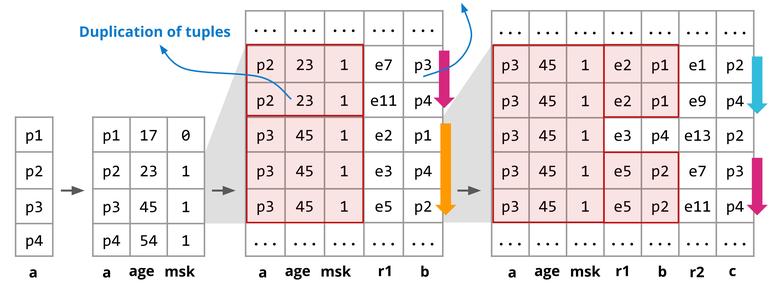
SCAN a → FILTER a.age > 20 → JOIN a□b → JOIN b□c

p1	e1,p2	e9,p4
p2	e7,p3	e11,p4
p3	e2,p1	e3,p4	e5,p2

Forward Adjacency Lists

Replication of AdjLists

Duplication of tuples



- ***Requirement :*** Avoid duplication of data and materializing of Adjacency lists
- ***Solution:***
  1/ Factorized Representation
  2/ List Groups
  - No Fixed Size
  - Do not materialize AdjLists
  - Flattened or Unflattened