Marcus Paradies, Michael Rudolf, Christof Bornhoevd, Wolfgang Lehner

GRATIN: Accelerating Graph Traversals in Main-Memory Column Stores

GRADES'14 Workshop

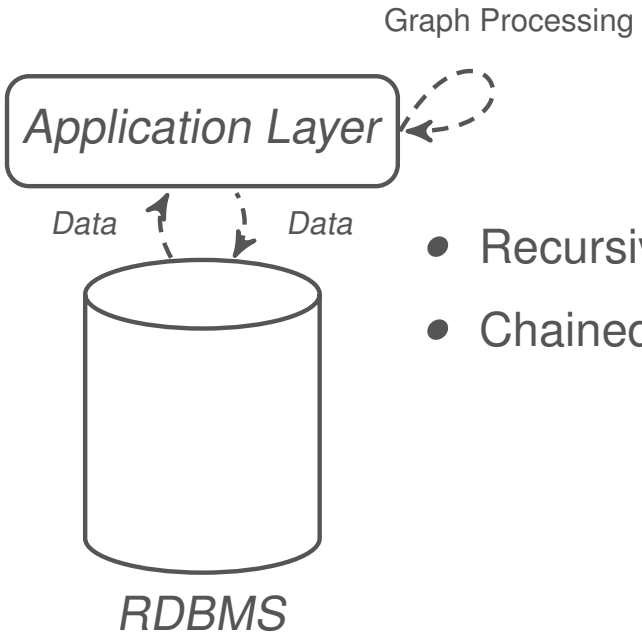
June 22, 2014

Graphs from an Enterprise View

Graphs from an Enterprise View

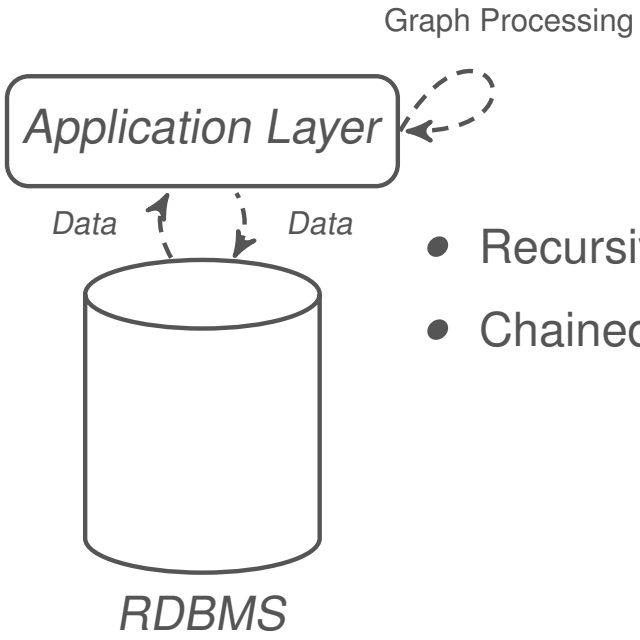
Relational

+ Application Logic



- Recursive Queries
- Chained Joins

Relational
+ Application Logic

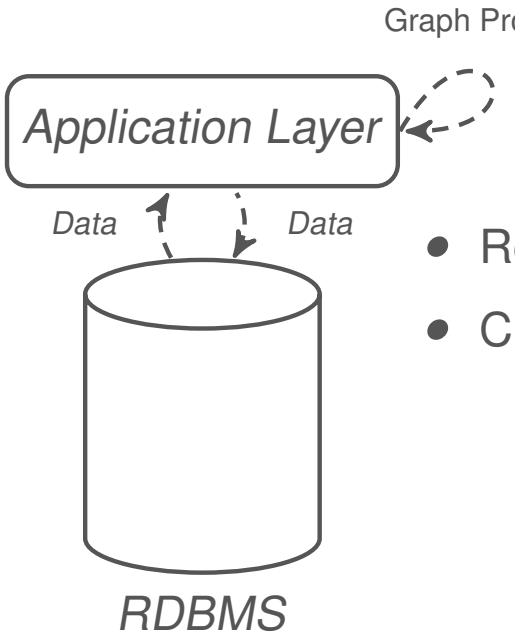


- Recursive Queries
- Chained Joins

✚ Data already in RDBMS
✖ SQL as interface
✖ Data transfer to application

Graphs from an Enterprise View

Relational + Application Logic



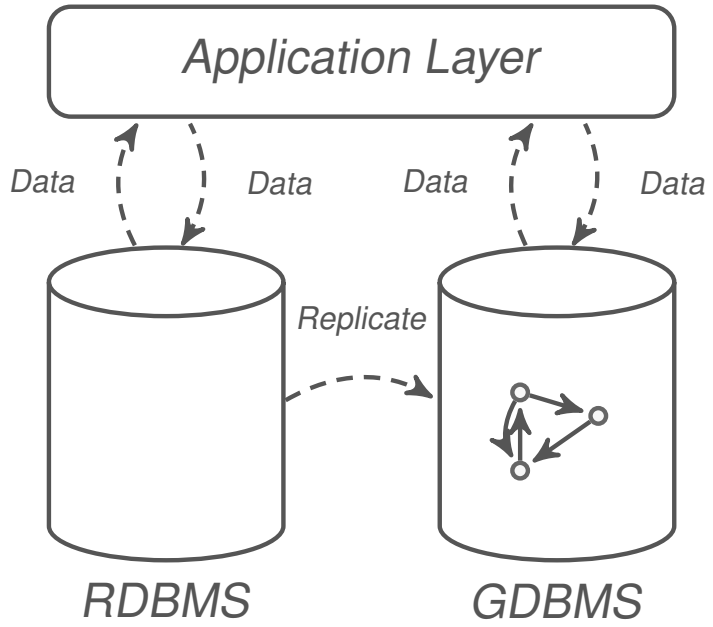
- Recursive Queries
- Chained Joins

✚ Data already in RDBMS

✖ SQL as interface

✖ Data transfer to application

Relational + Graph + Application Logic



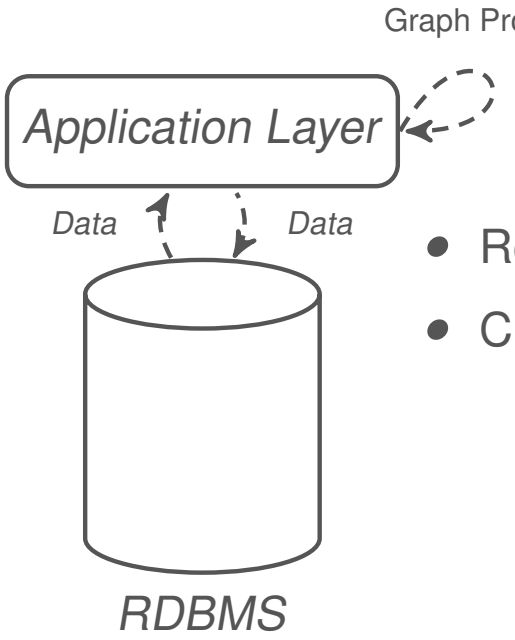
*dex

Neo4j

InfiniteGraph

Graphs from an Enterprise View

Relational + Application Logic



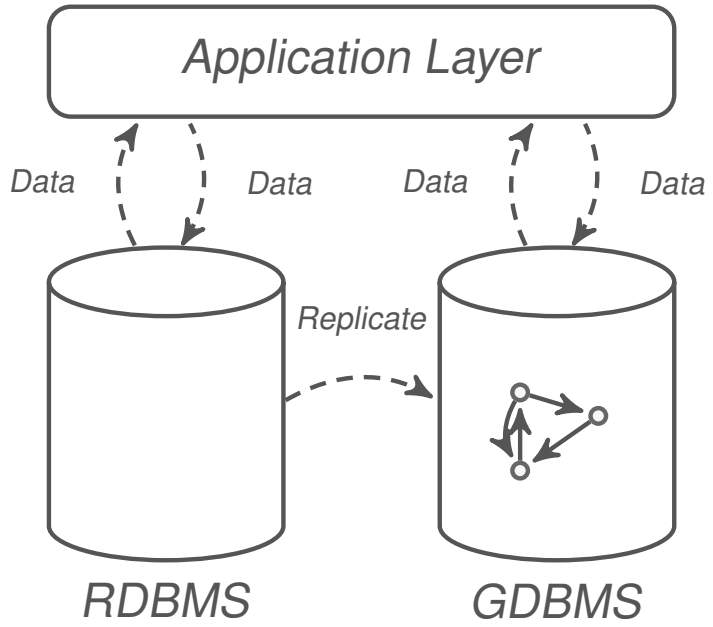
- Recursive Queries
- Chained Joins

✚ Data already in RDBMS

✖ SQL as interface

✖ Data transfer to application

Relational + Graph + Application Logic



*dex

Neo4j

InfiniteGraph

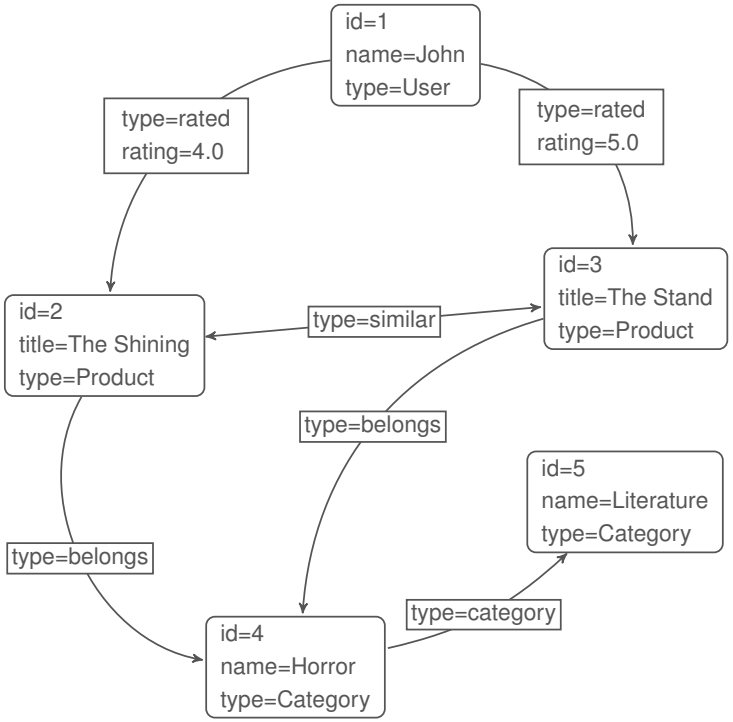
✚ Efficient processing in GDBMS

✖ Processing on replicated data

✖ No combination with relational data

Graph Representation

Graph Representation



Example graph

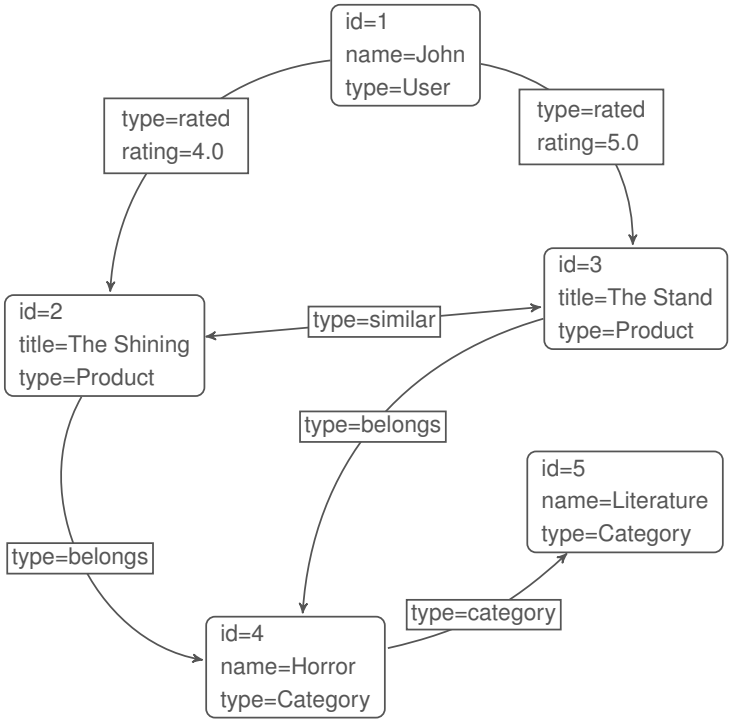
id	*type*	*name*	...	*title*
1	User	John	...	?
2	Product	?	...	The Shining
3	Product	?	...	The Stand
4	Category	Horror	...	?
5	Category	Literature	...	?

Vertex table

V_s	V_t	*type*	...	*rating*
3	2	similar	...	?
2	3	similar	...	?
2	4	belongs	...	?
3	4	belongs	...	?
1	3	rated	...	5.0
1	2	rated	...	4.0
4	5	category	...	?

Edge table

Graph Representation



Example graph

id	*type*	*name*	...	*title*
1	User	John	...	?
2	Product	?	...	The Shining
3	Product	?	...	The Stand
4	Category	Horror	...	?
5	Category	Literature	...	?

Vertex table

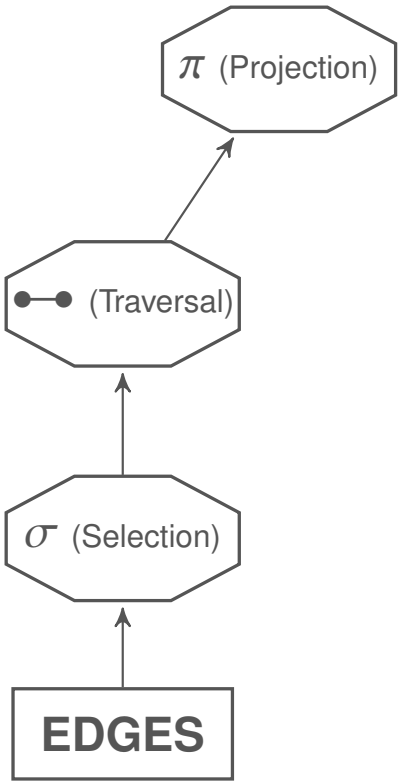
V_s	V_t	*type*	...	*rating*
3	2	similar	...	?
2	3	similar	...	?
2	4	belongs	...	?
3	4	belongs	...	?
1	3	rated	...	5.0
1	2	rated	...	4.0
4	5	category	...	?

Edge table

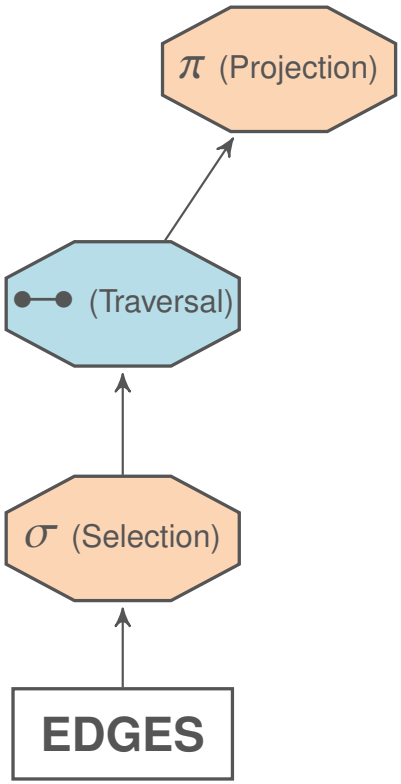
- Each vertex/edge represented as a single record in universal tables
- Support for transactions and compression
- Combination with other data models (spatial, text, temporal)

Query Execution

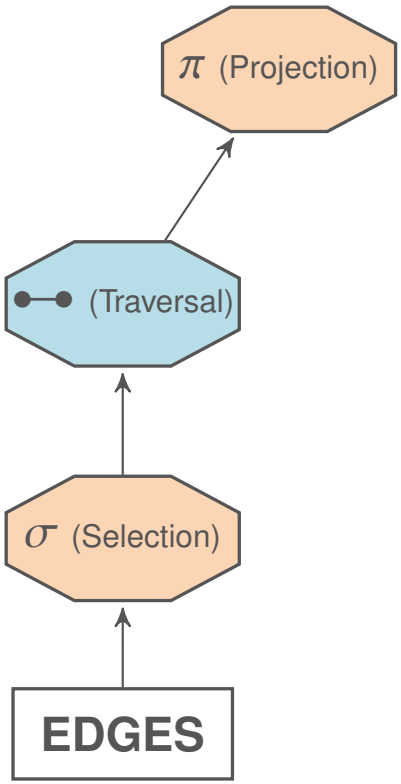
Query Execution



Query Execution

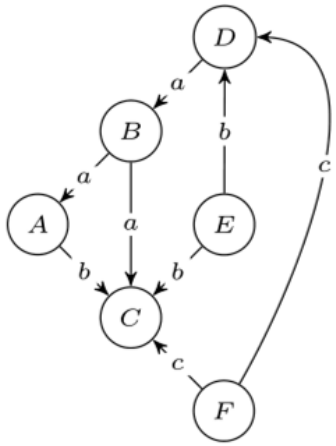


Query Execution



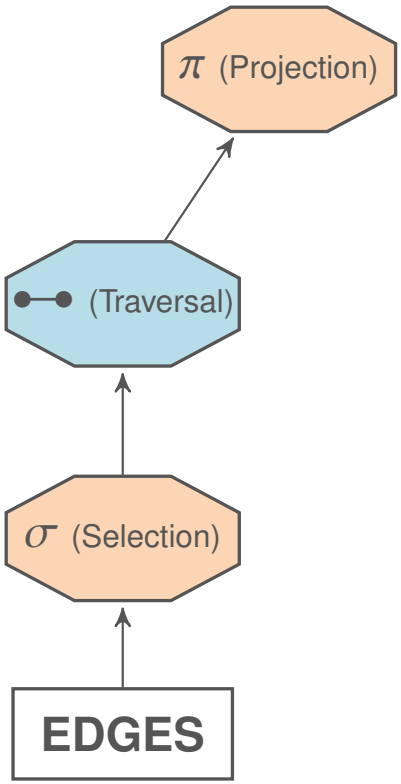
Graph Traversals

Example query: `{ id:D }-a->(1,*)`



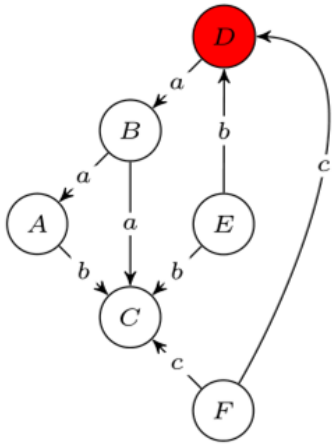
V_{source}	V_{target}	E_{type}
D	B	a
F	C	c
B	A	a
A	C	b
E	C	b
E	D	b
B	C	a
F	D	c

Query Execution



Graph Traversals

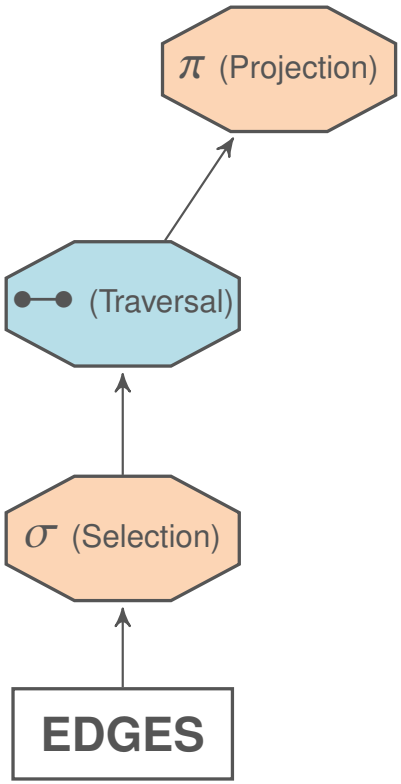
Example query: `{ id:D }-a->(1,*)`



Scan

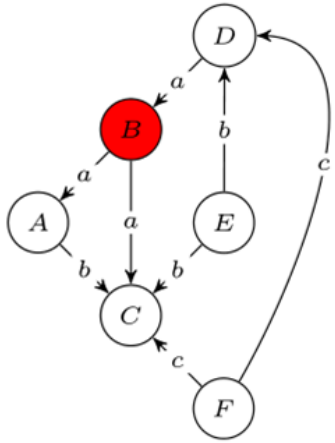
V_{source}	V_{target}	E_{type}
D	B	a
F	C	c
B	A	a
A	C	b
E	C	b
E	D	b
B	C	a
F	D	c

Query Execution



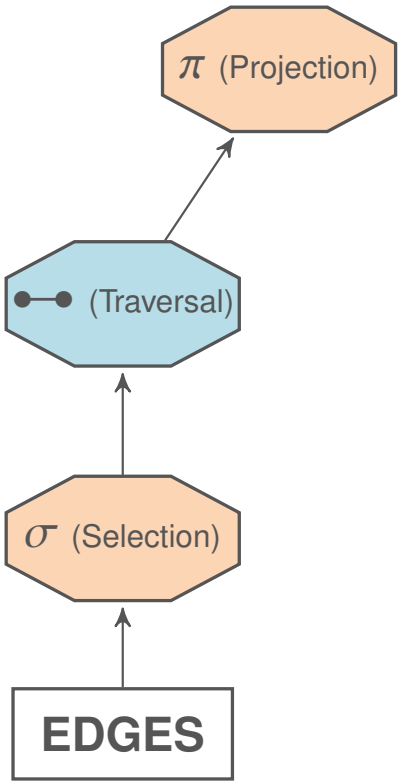
Graph Traversals

Example query: `{ id:D }-a->(1,*)`



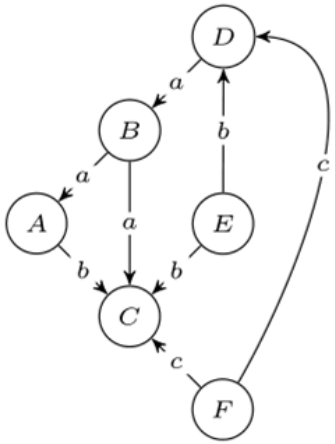
V_{source}	V_{target}	E_{type}
D	B	a
F	C	c
B	A	a
A	C	b
E	C	b
E	D	b
B	C	a
F	D	c

Query Execution



Graph Traversals

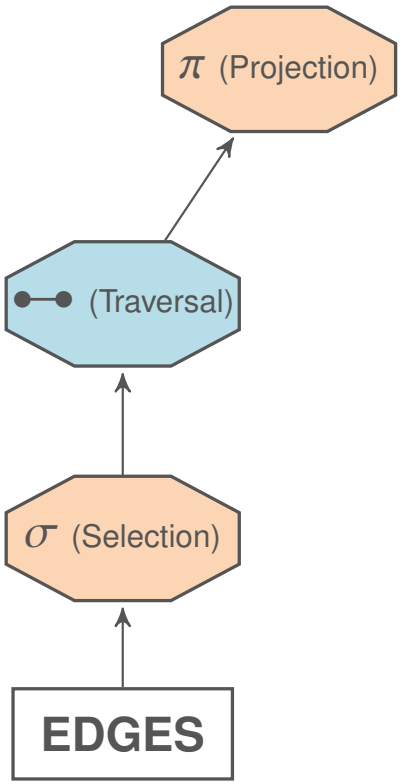
Example query: `{ id:D }-a->(1,*)`



Scan

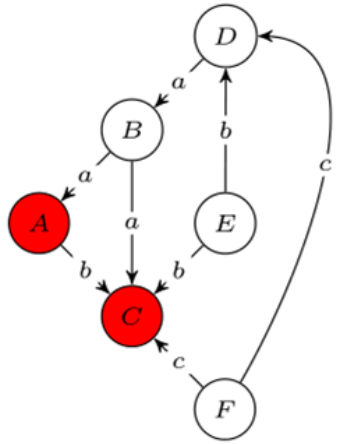
V_{source}	V_{target}	E_{type}
D	B	a
F	C	c
B	A	a
A	C	b
E	C	b
E	D	b
B	C	a
F	D	c

Query Execution



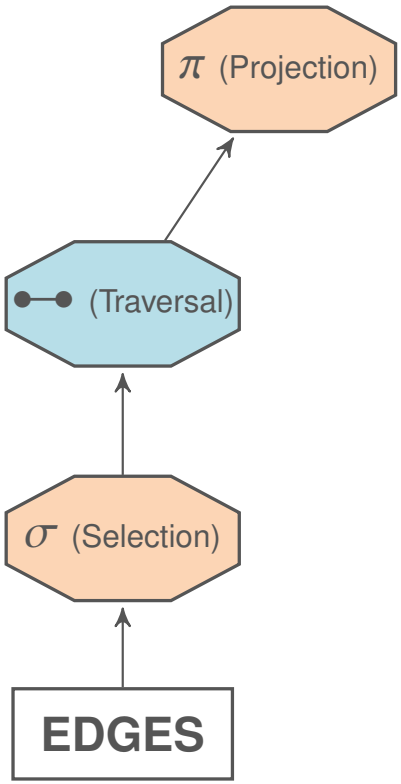
Graph Traversals

Example query: `{ id:D }-a->(1,*)`



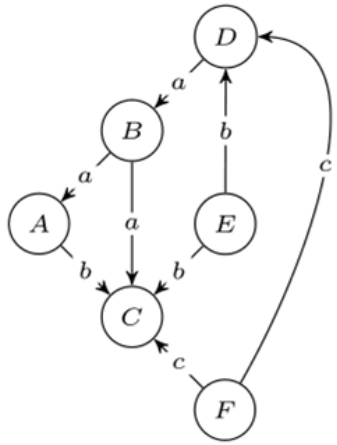
V_{source}	V_{target}	E_{type}
D	B	a
F	C	c
B	A	a
A	C	b
E	C	b
E	D	b
B	C	a
F	D	c

Query Execution



Graph Traversals

Example query: `{ id:D }-a->(1,*)`

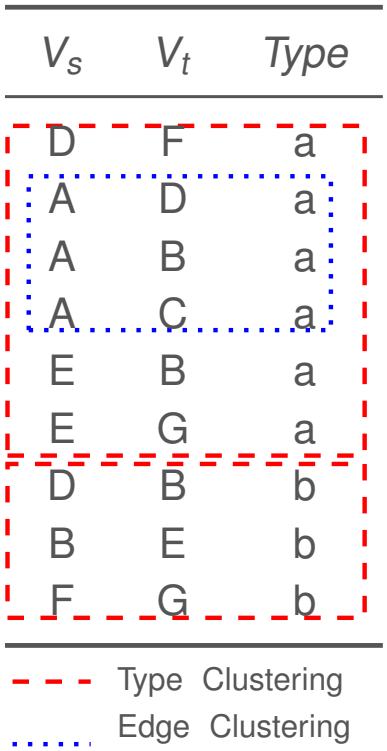


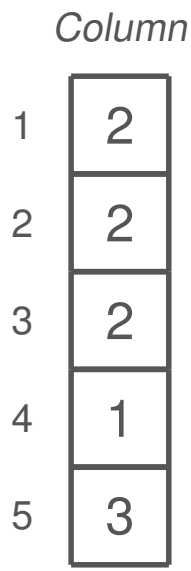
Scan

V_{source}	V_{target}	E_{type}
D	B	a
F	C	c
B	A	a
A	C	b
E	C	b
E	D	b
B	C	a
F	D	c

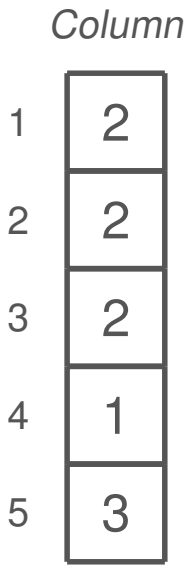
Edge Clustering

- Clustering by edge type preserves subgraph meaning
- Clustering by edge source preserves vertex neighborhood
- Increases spatial locality in memory
- Allows reducing scan to range in column



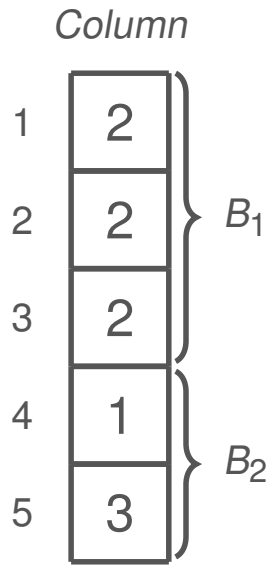


- GRATIN replaces full column scans by block scans



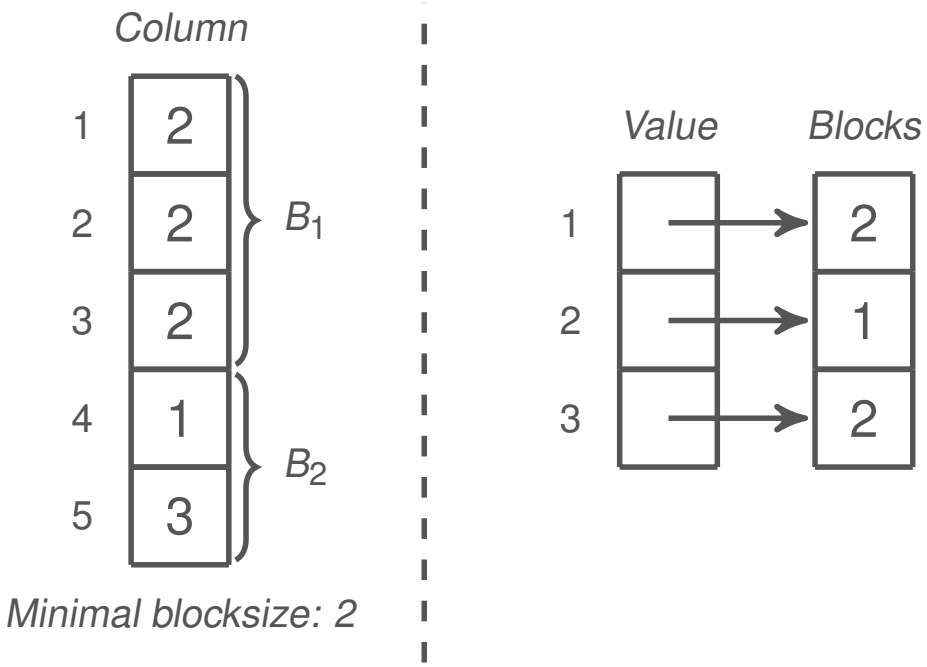
Minimal blocksize: 2

- GRATIN replaces full column scans by block scans
- GRATIN indexes column with variable block size

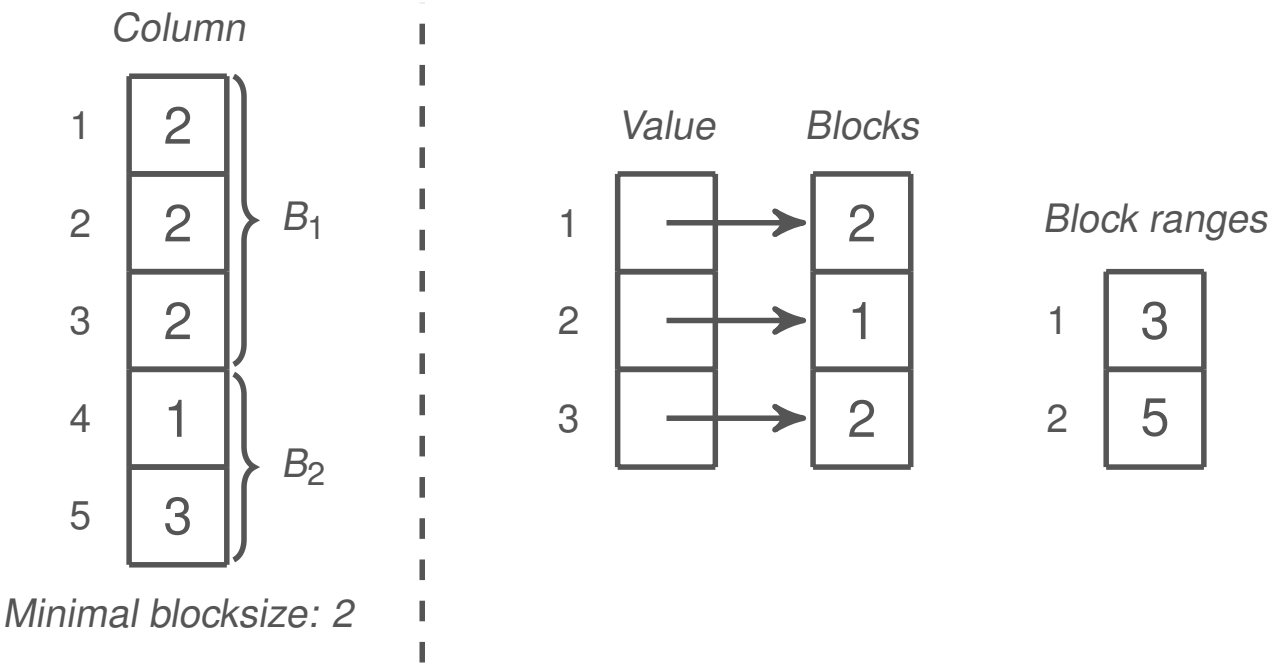


Minimal blocksize: 2

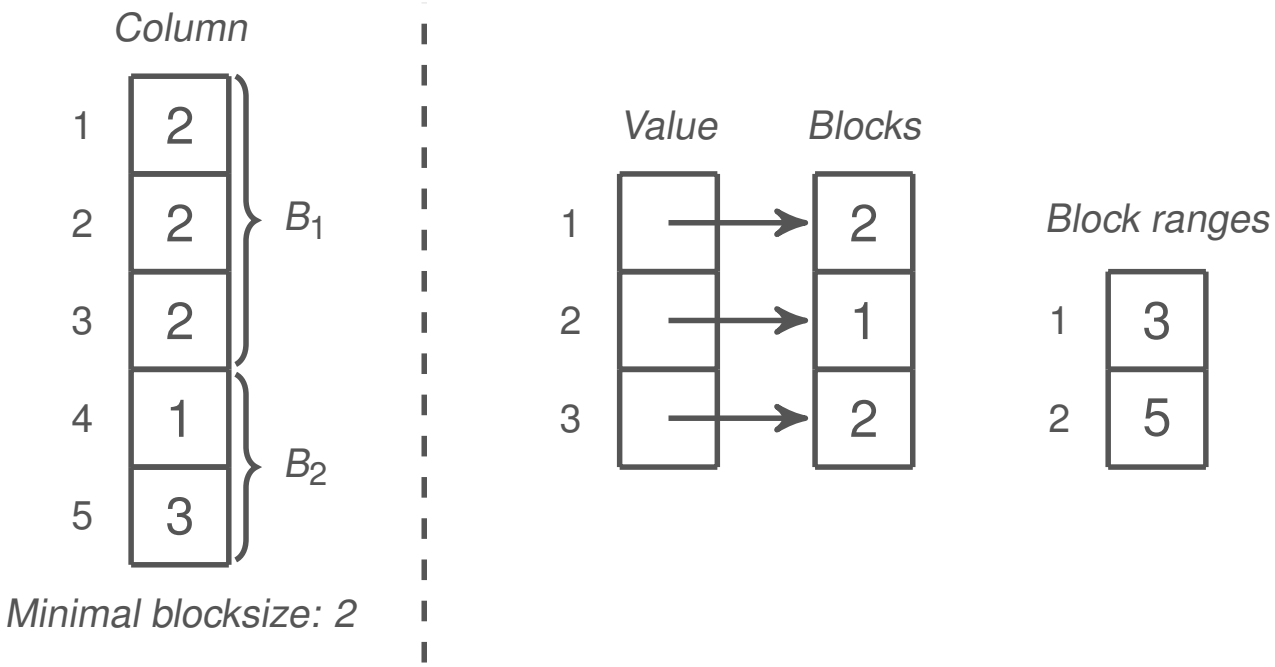
- GRATIN replaces full column scans by block scans
- GRATIN indexes column with variable block size



- GRATIN replaces full column scans by block scans
- GRATIN indexes column with variable block size



- GRATIN replaces full column scans by block scans
- GRATIN indexes column with variable block size



- GRATIN replaces full column scans by block scans
- GRATIN indexes column with variable block size
- Allows efficient handling of vertices with high outdegree

Experiments on Static Graphs

ID	$\|V\|$	$\|E\|$	$\bar{d}_{out}$
Amazon	0.4 M	3.3 M	16.8
California-Roads	1.9 M	2.7 M	2.8

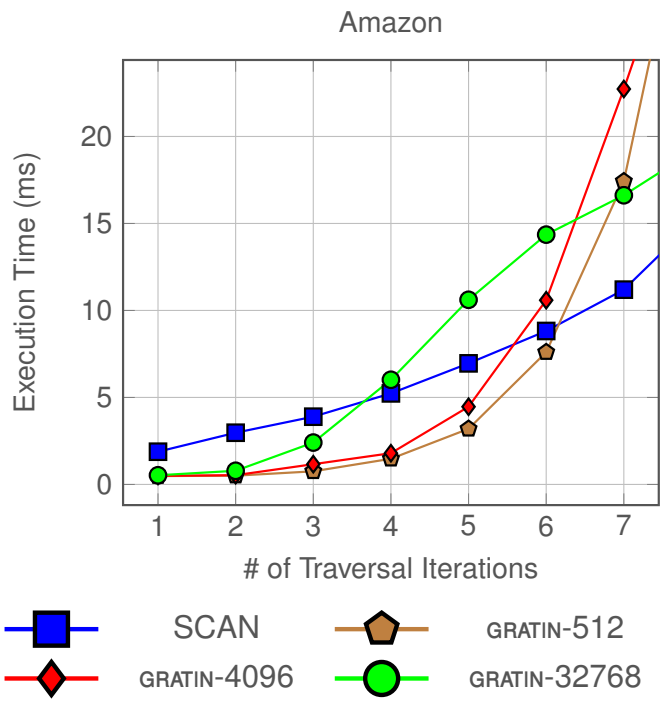


Figure: Comparison for different block sizes.

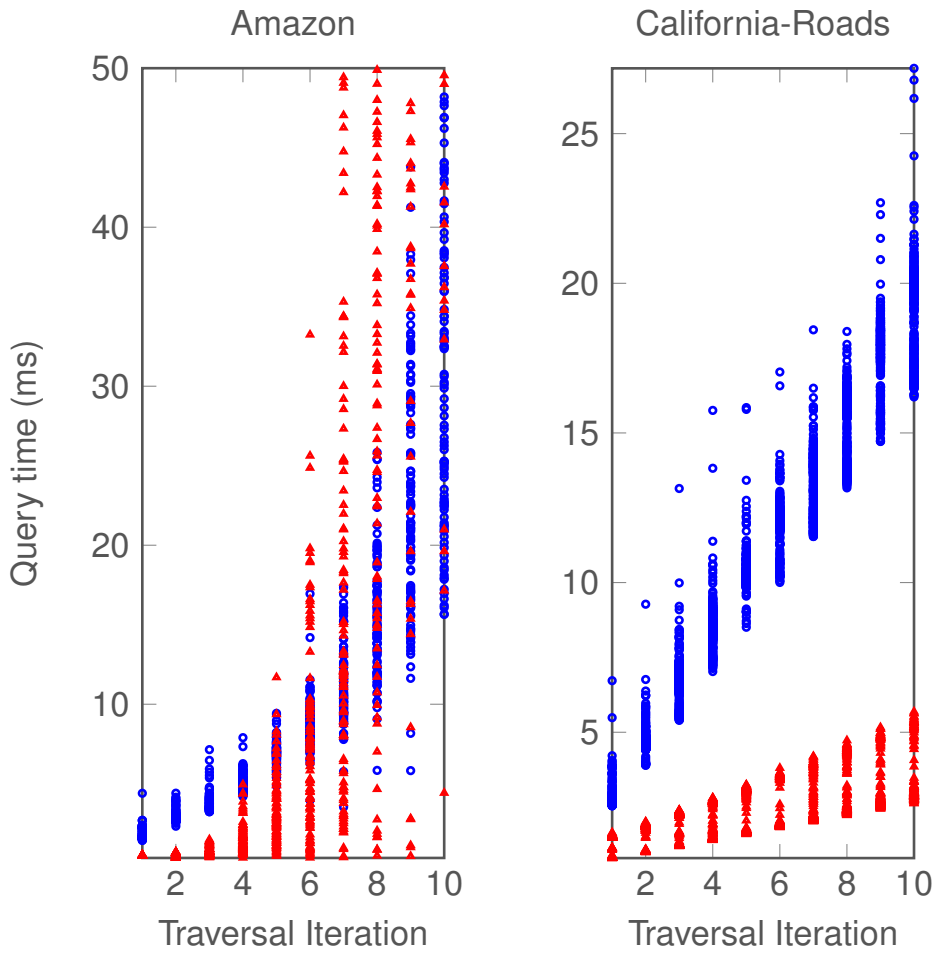
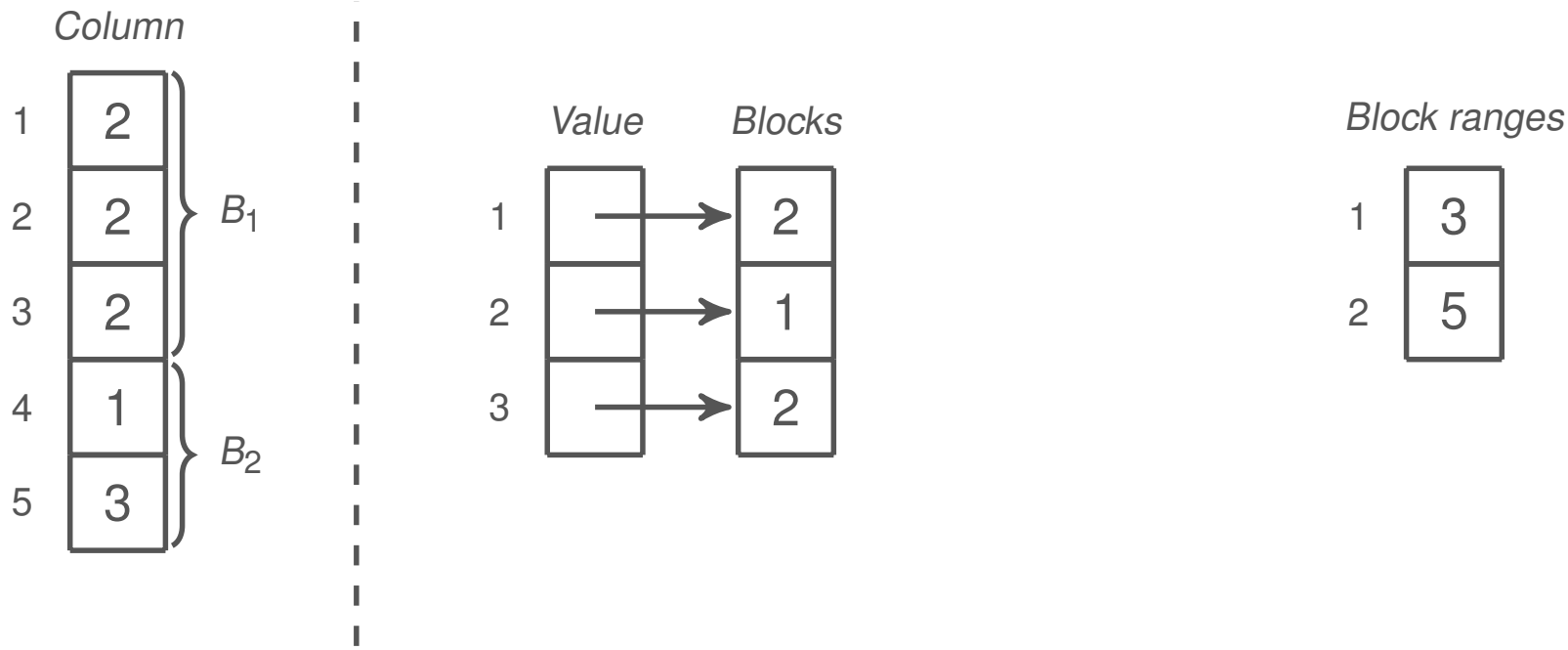


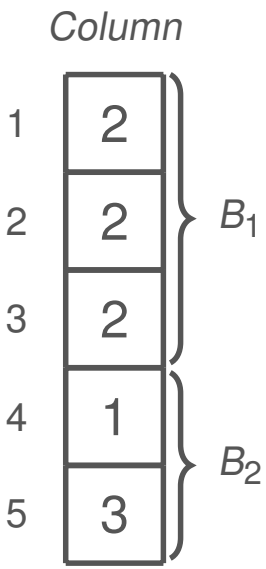
Figure: Query time for scan-based traversal (○) and GRATIN-based traversal (△).

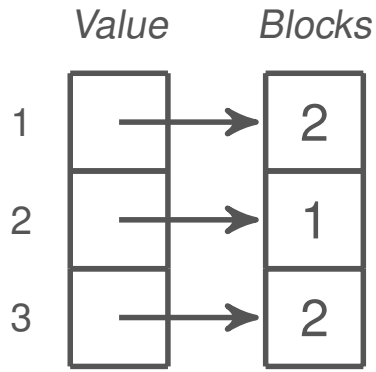
Handling Updates



Minimal blocksize: 2

Handling Updates





Health factor

- $h_{B_1} = 1.0$
- $h_{B_2} = 1.0$
- $h_{B_3} = 1.0$

Block ranges

1	3
2	5

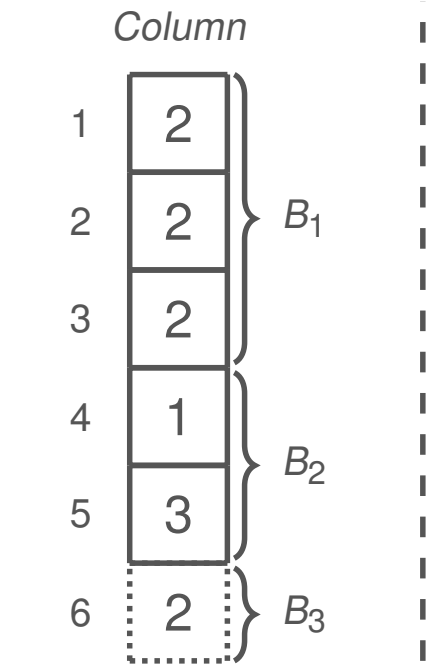
Minimal blocksize: 2

GRATIN health

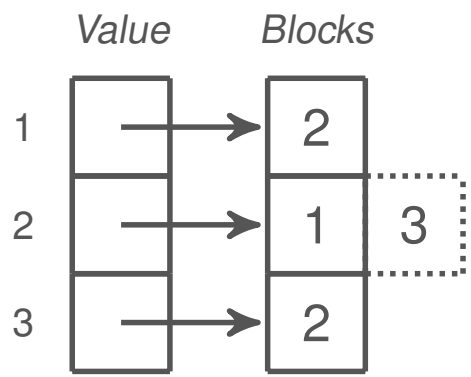
- $h = 1.0$

- Health factor describes viability of GRATIN
- If global health factor below threshold → rebuild index

Handling Updates



Minimal blocksize: 2



Health factor

- $h_{B_1} = 1.0$
- $h_{B_2} = 1.0$
- $h_{B_3} = 1.0$

Block ranges

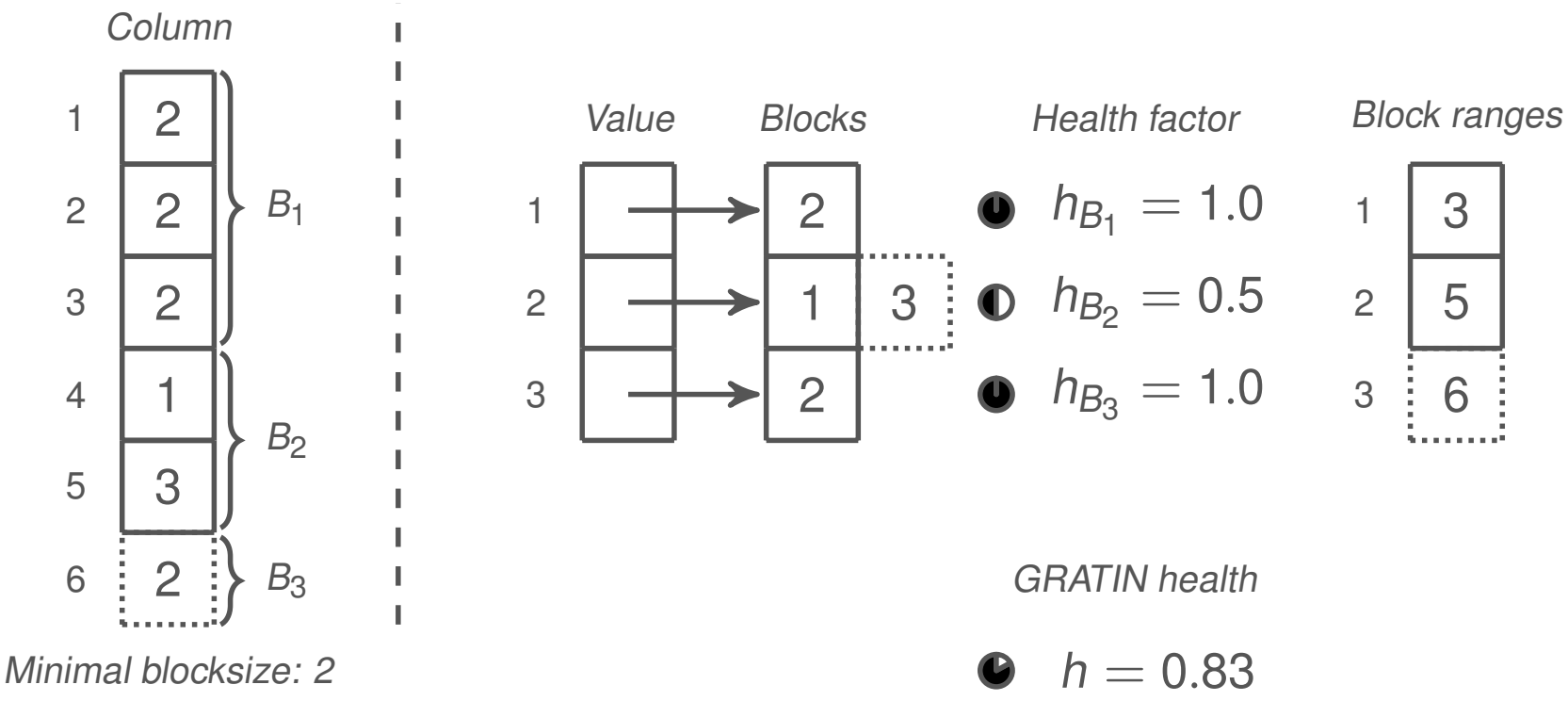
1	3
2	5
3	6

GRATIN health

- $h = 1.0$

- Health factor describes viability of GRATIN
- If global health factor below threshold → rebuild index
- GRATIN allows updates in constant time (append-only)

Handling Updates



- Health factor describes viability of GRATIN
- If global health factor below threshold → rebuild index
- GRATIN allows updates in constant time (append-only)

Experiments on Dynamic Graphs

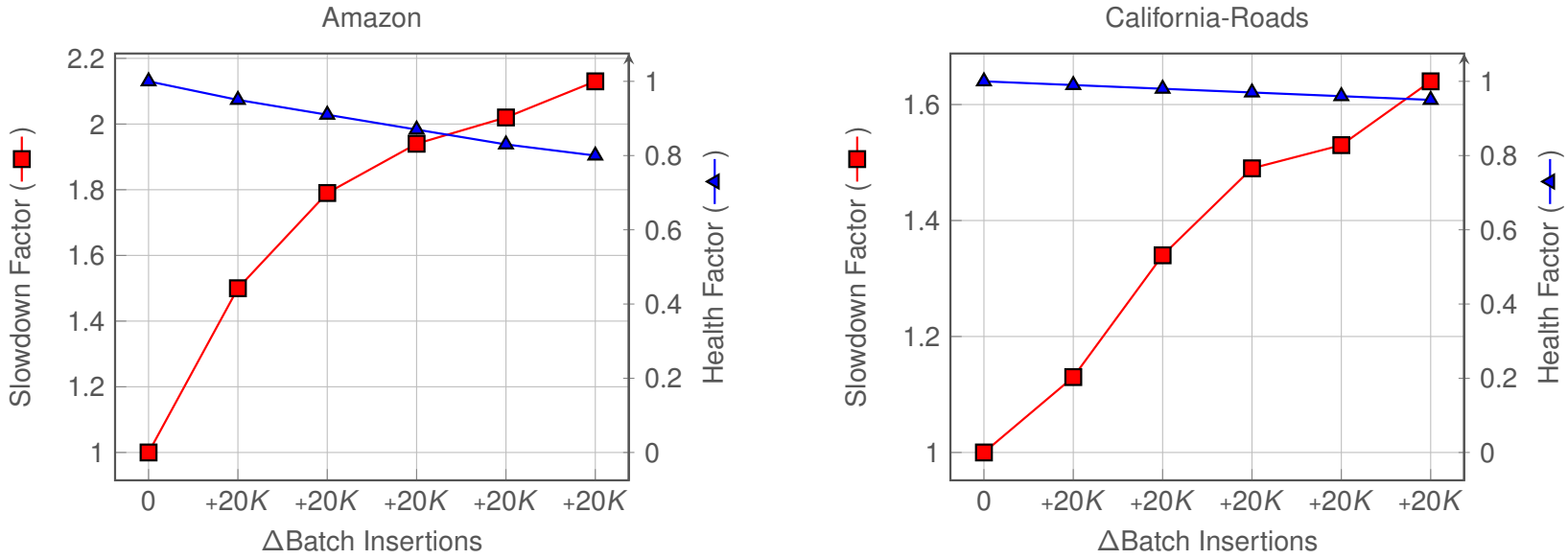
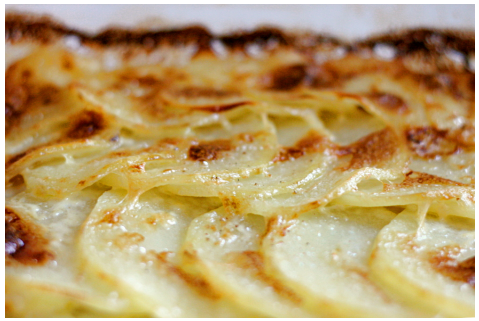
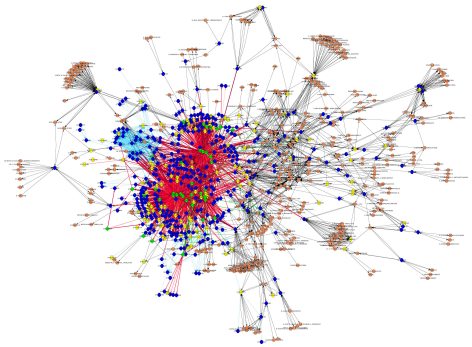


Figure: Query time for GRATIN-based traversal on dynamic graphs. Slowdown factor describes the relative execution time in multiples of the execution time on a static graph.

Summary

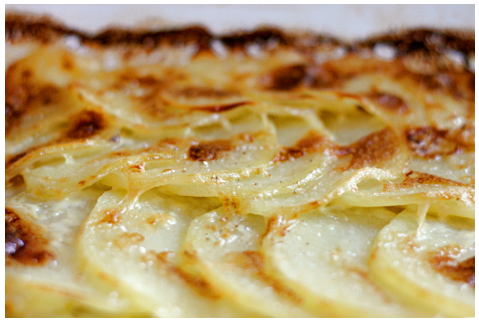
- Tight integration of traversal operator into main-memory column store
- GRATIN is a lightweight secondary index structure
- Handles dynamic graphs in predictable time
- Experiments show a diverse spectrum of performance improvements
- Performance of GRATIN depends on graph topology

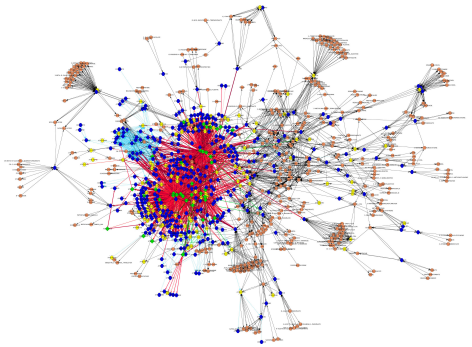




Summary

- Tight integration of traversal operator into main-memory column store
- GRATIN is a lightweight secondary index structure
- Handles dynamic graphs in predictable time
- Experiments show a diverse spectrum of performance improvements
- Performance of GRATIN depends on graph topology





If you want to know more, join me for a chat at my poster!

1 Backup Slides