Approaching Join Index

yet another one join algorithm

Mikhail Khludnev

principal engineer

Grid Dynamics

Lucidworks

Grid Dynamics

- Grid Dynamics is a Silicon Valley-based leading provider of scalable, next-generation commerce technology solutions
- Record of outperformance with Tier 1 retail clients
- Fortune 1000 client relationships

macy's

AMERICAN EAGLE OUTFITTERS

Integral Ad Science

KOHL'S

YAHOO!

x.commerce

Toys R us

ebay

Raley's
Family of Fine Stores
BEL AIR NOB HILL FOODS FOOD SOURCE

PayPal™

Ring Central

About Me

- principal engineer at Grid Dynamics
- spoke at few last LuceneRevolutions
- contributed BlockJoin query parser for Solr - {!parent}
- blogged about it at http://blog.griddynamics.com/
- tried to fix threads at DataImportHandler

http://google.com/+MikhailKhludnev

You are expected to know

- how Lucene **searches**/filters
- how it counts **facets**
- that there are **segments**
- what is **DocValues**
- why to **join**
- RDBMS joins: nested loop join, sort-merge join and hash join.

I’m expected to know

- **query-time** join
- **index-time** join
- yet another one join

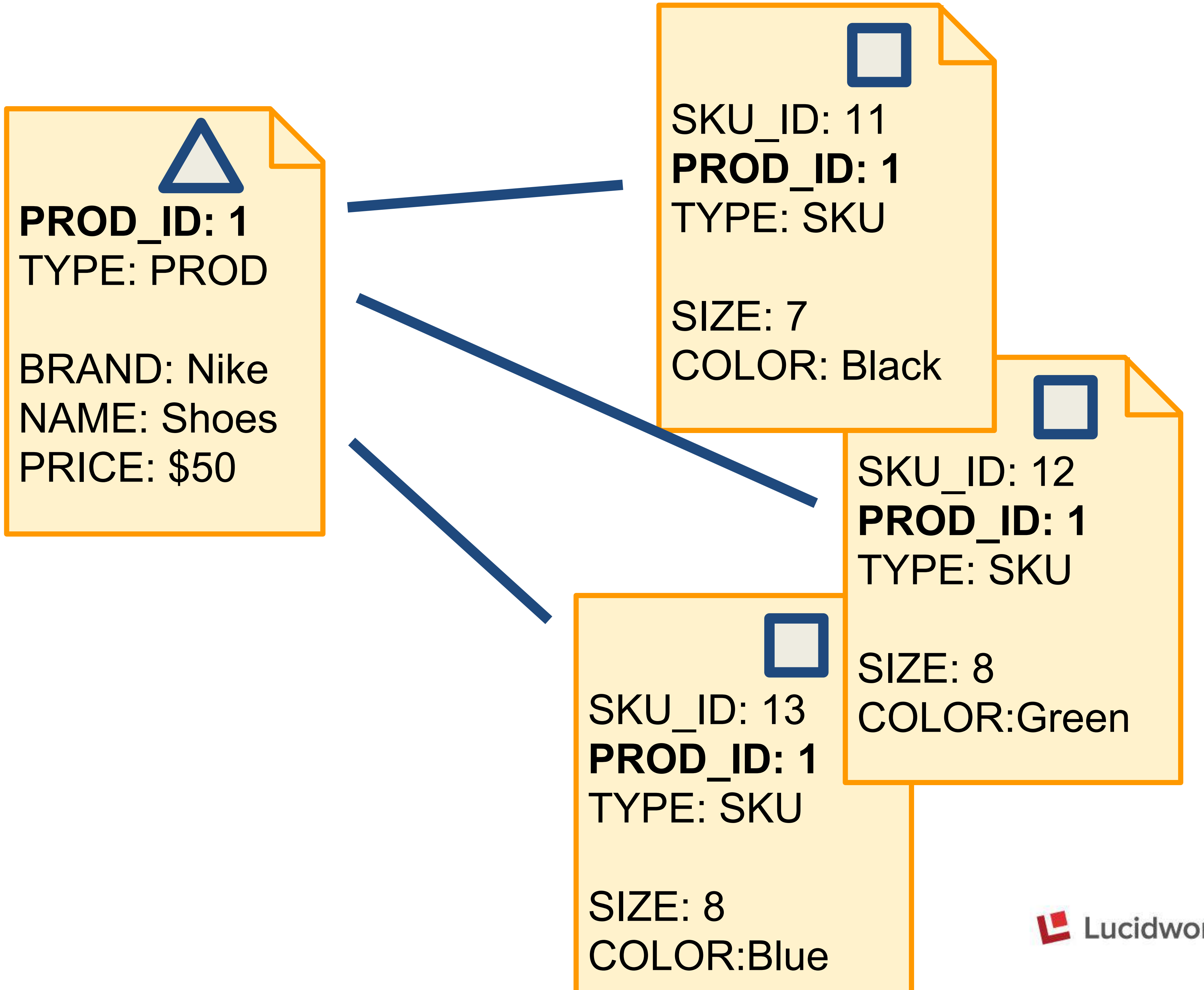
Lucene/Solr/Elastic Is Strong

- searching
  - filtering
- analytics
  - facets
  - pivots
  - stats

There is a weakness

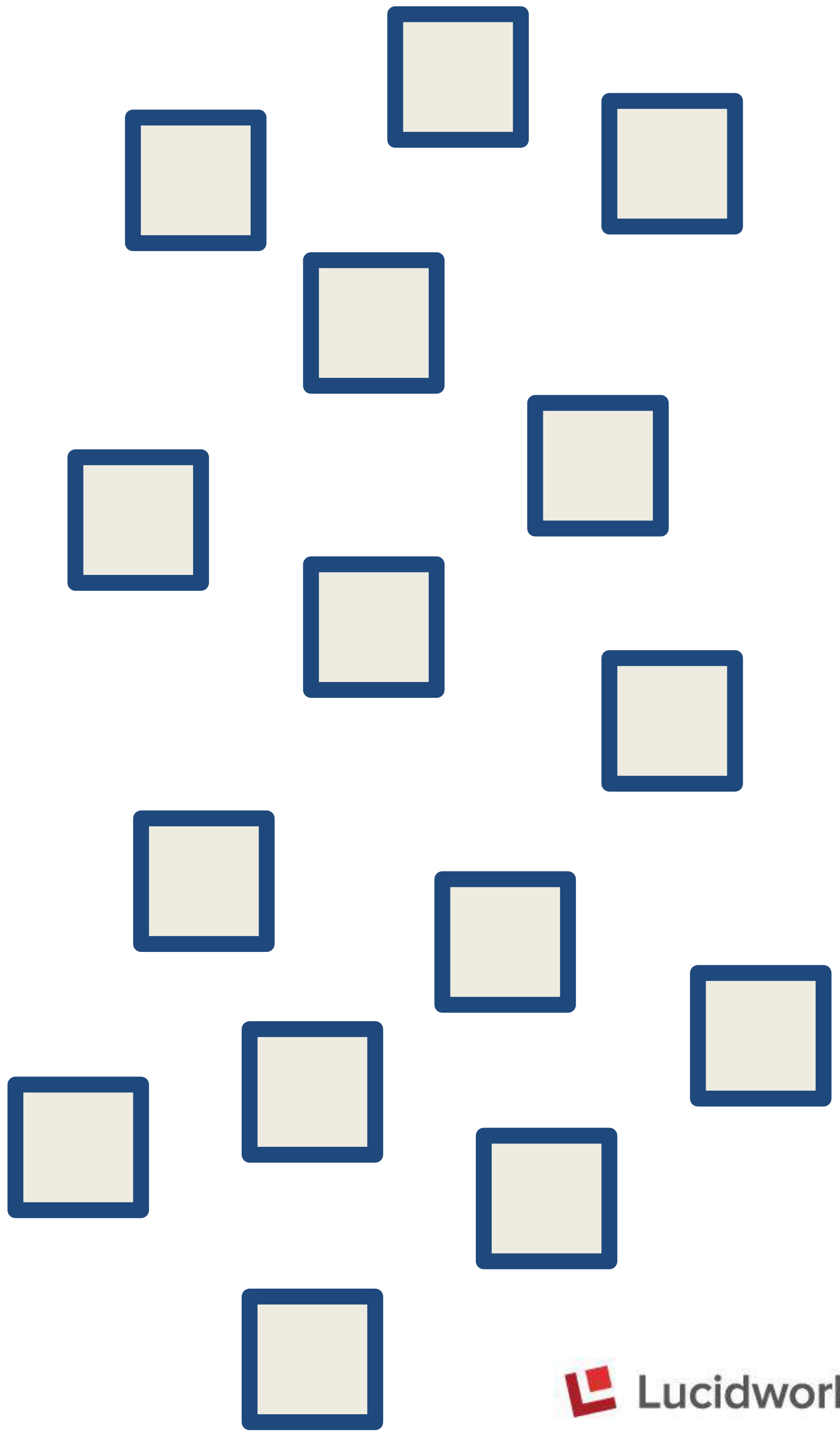
LUCENE/SOLR**REVOLUTION**

- robust joins
  - multiple entities
  - relations

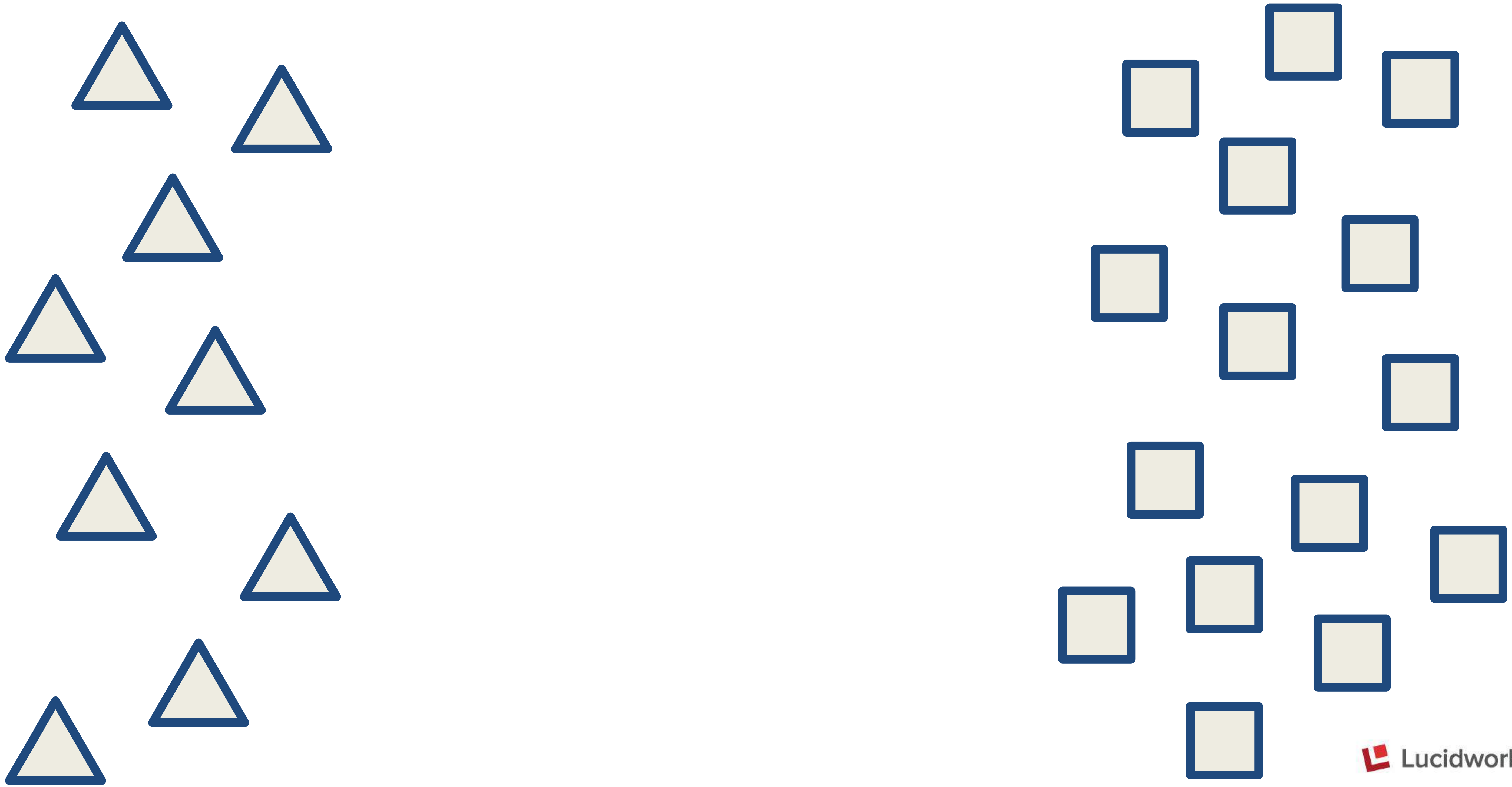


Lucidworks

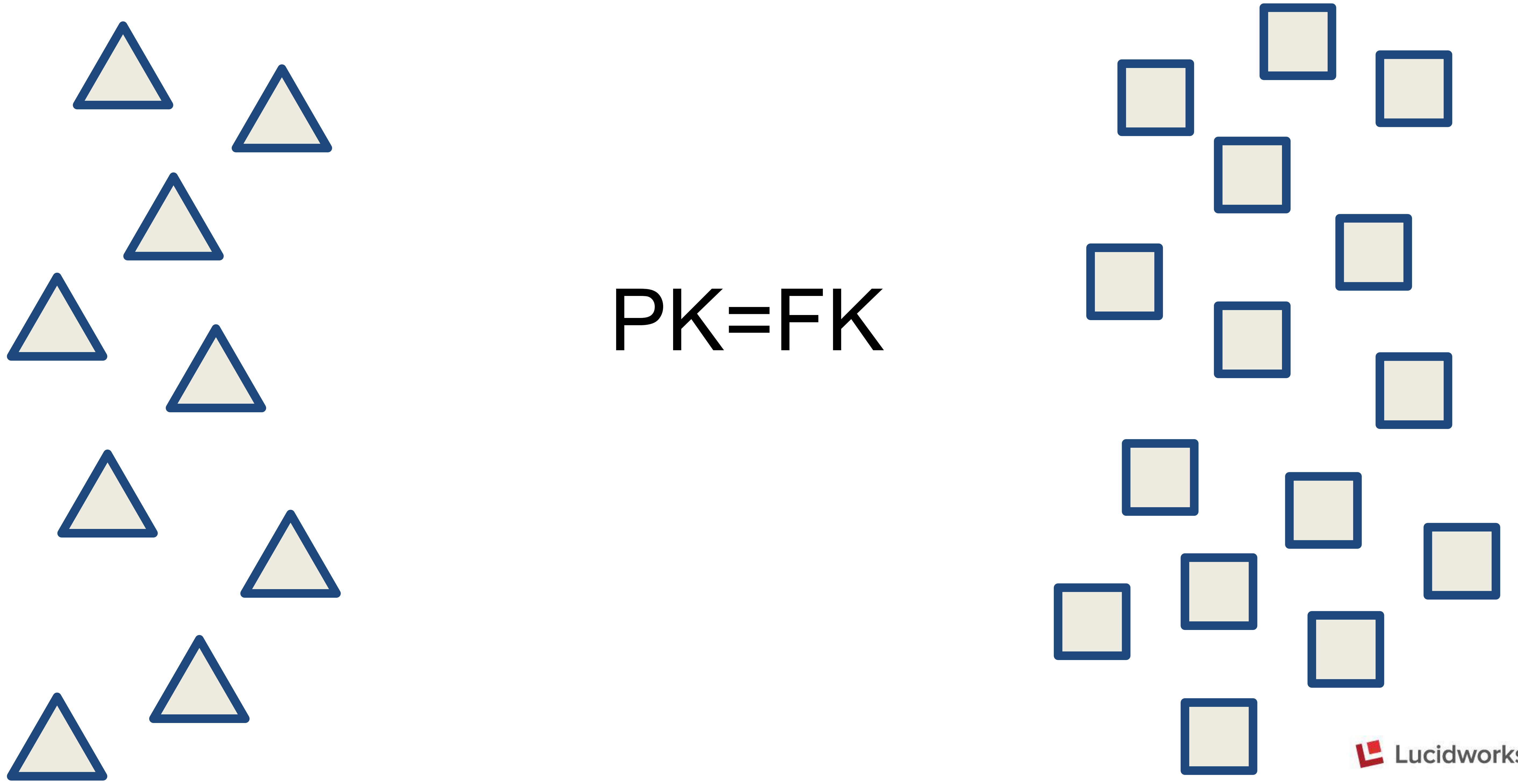
Joins in General



Joins in General

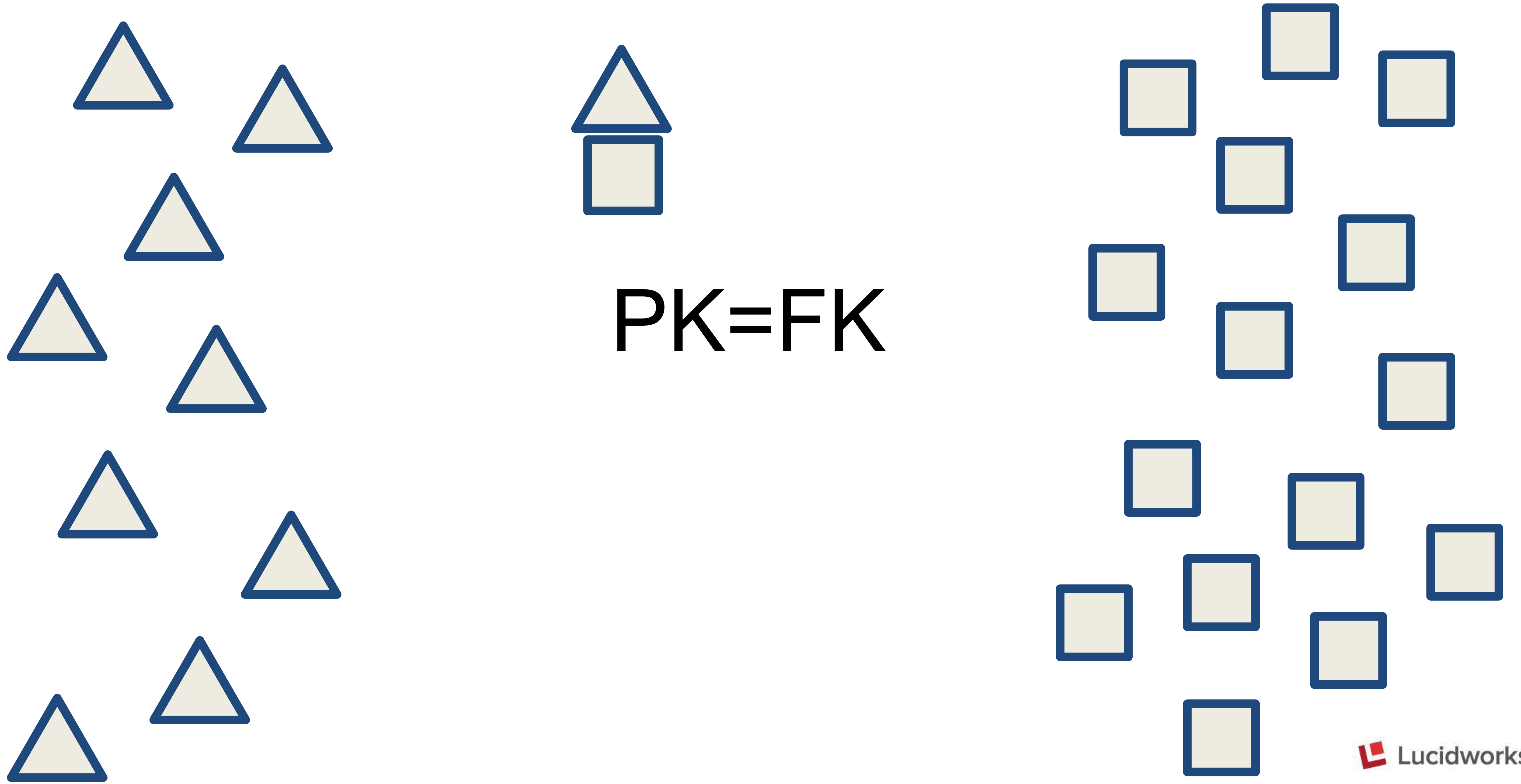


Joins in General



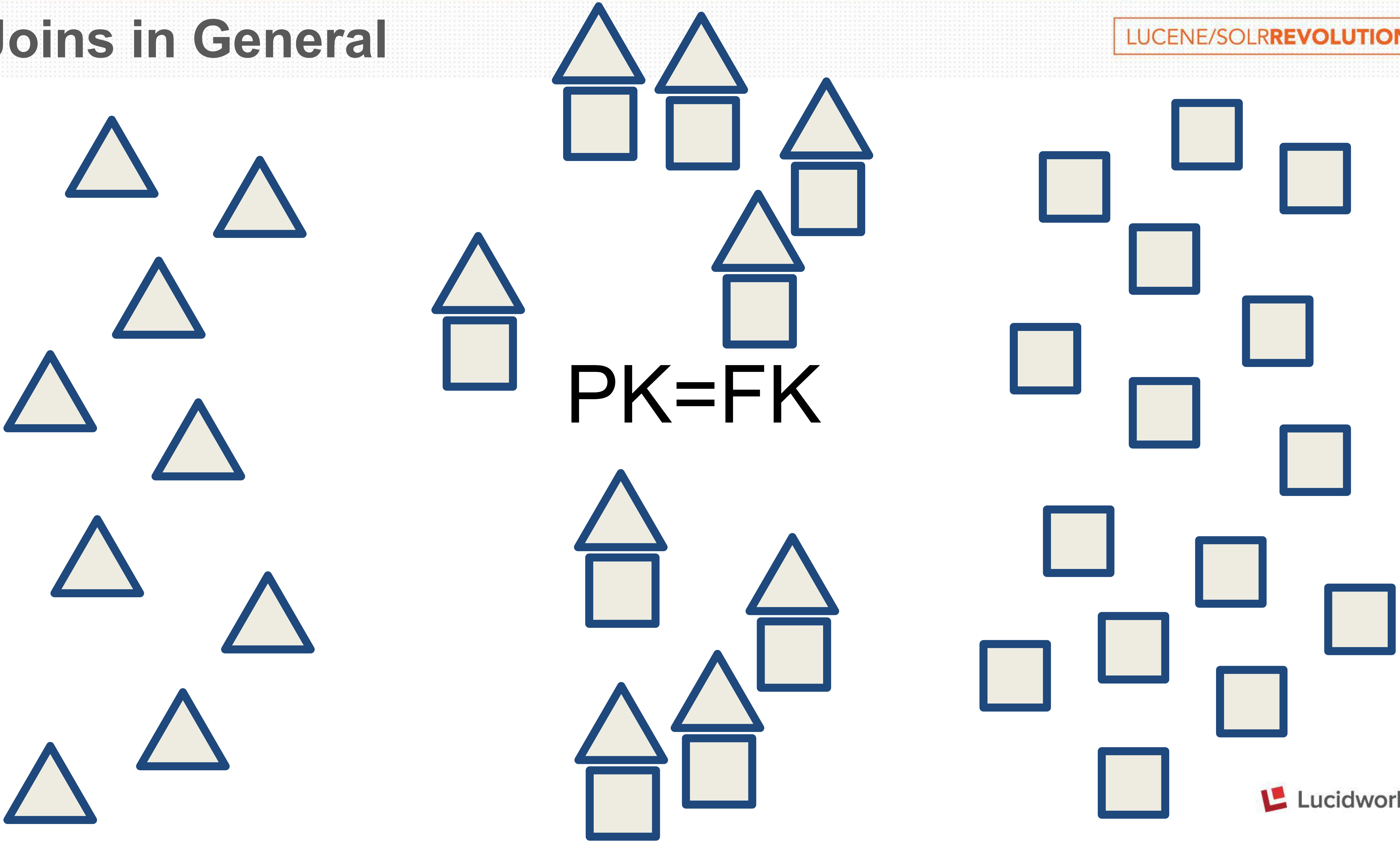
Joins in General

LUCENE/SOLR**REVOLUTION**



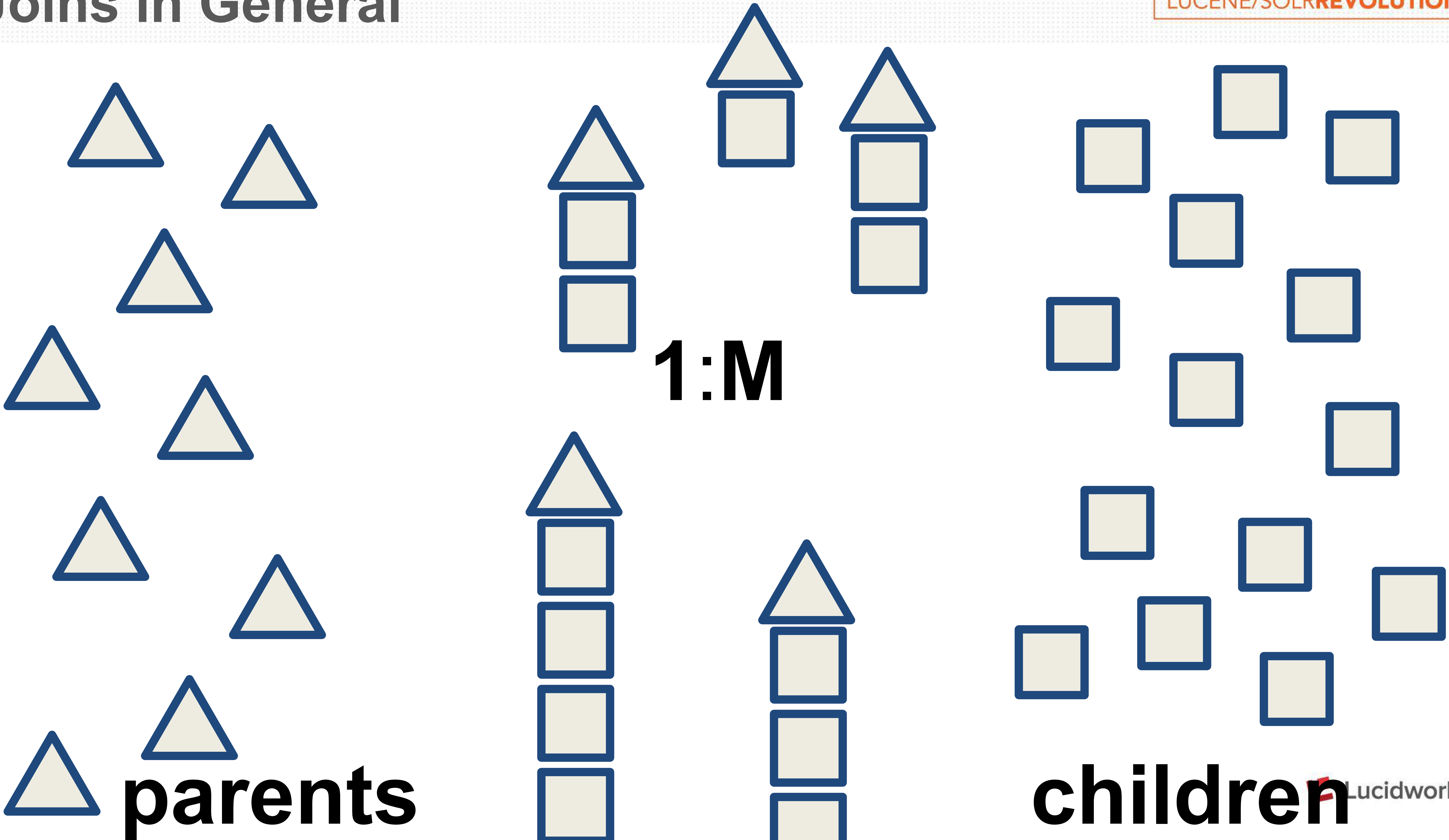
Lucidworks

Joins in General

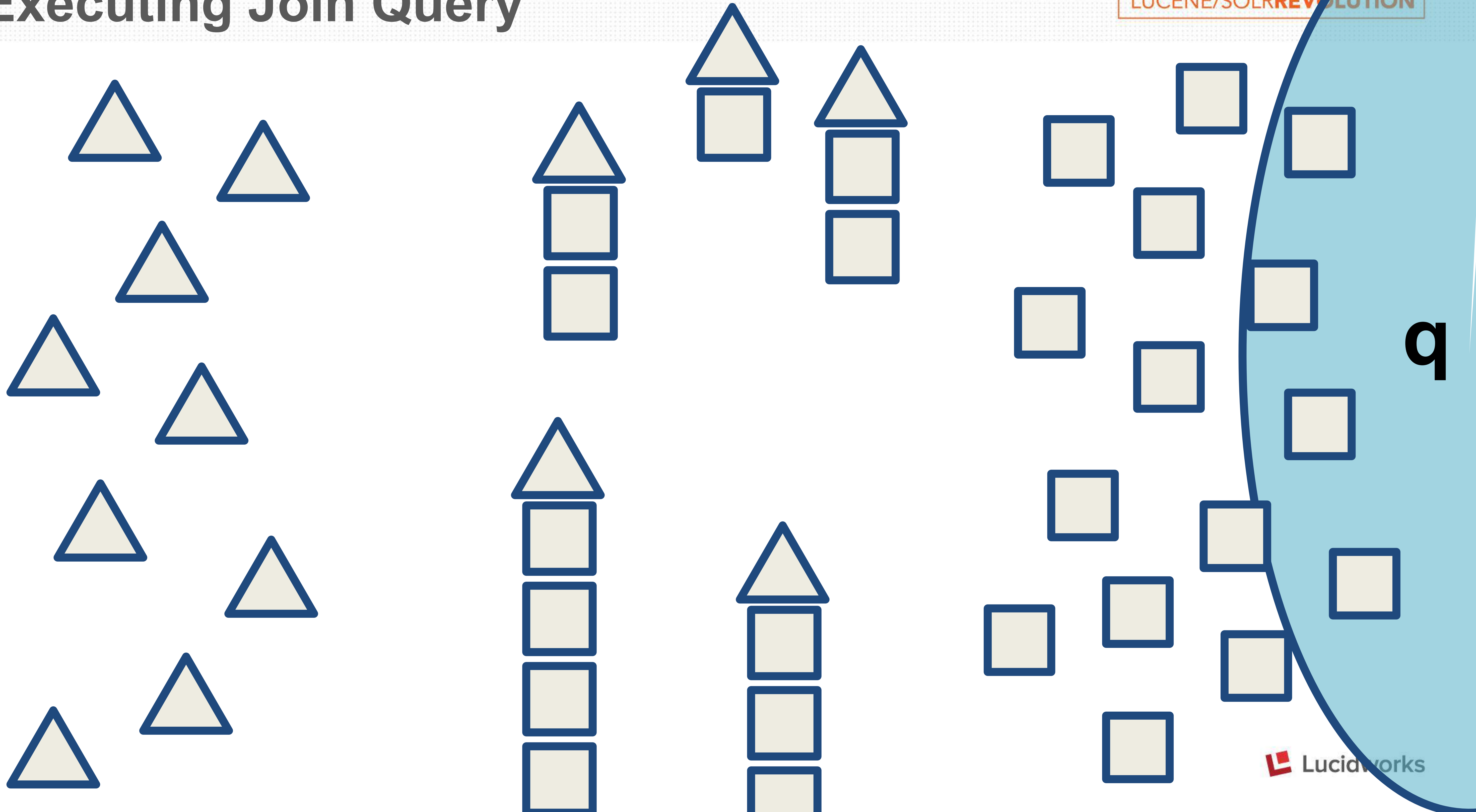


Joins in General

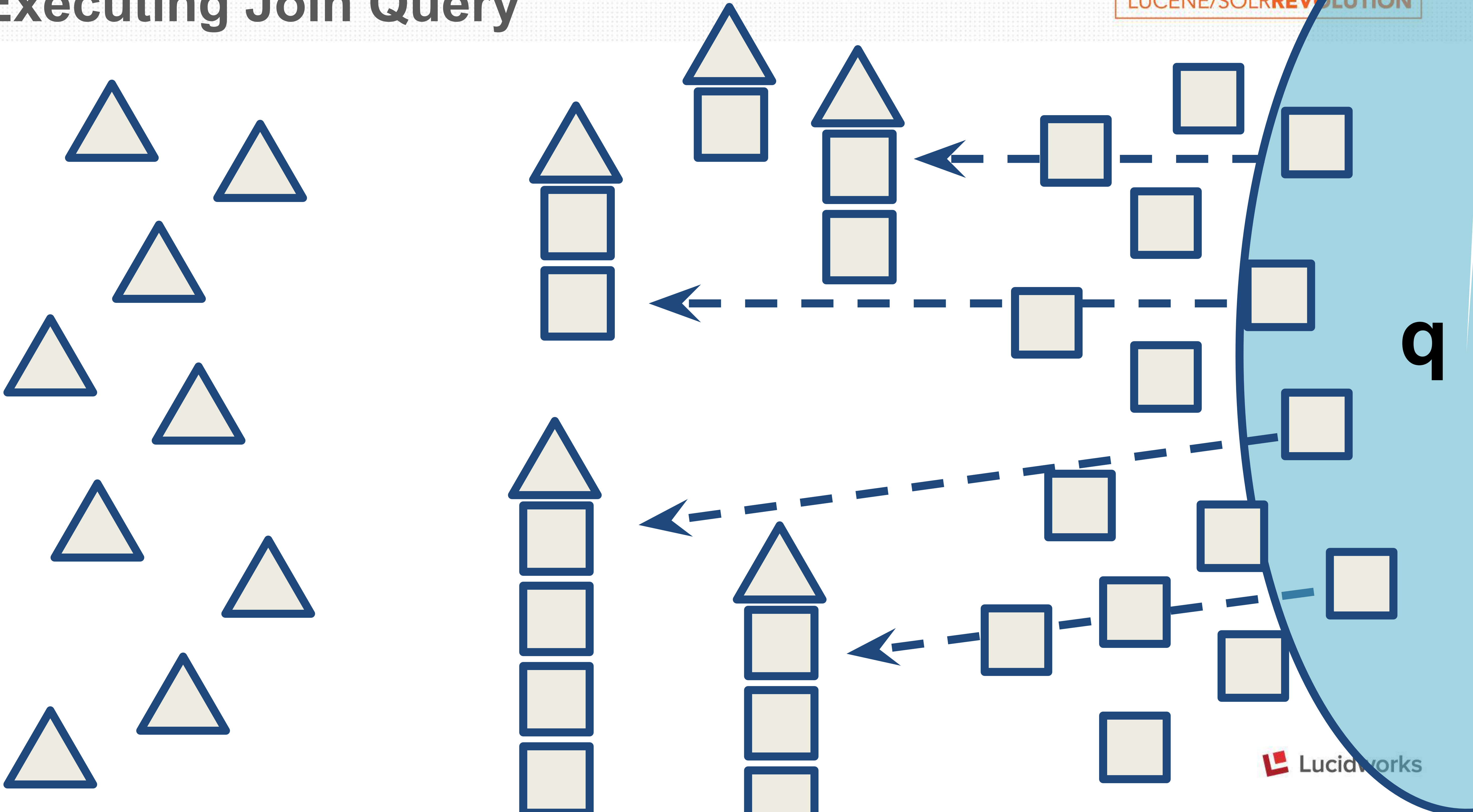
LUCENE/SOLR**REVOLUTION**



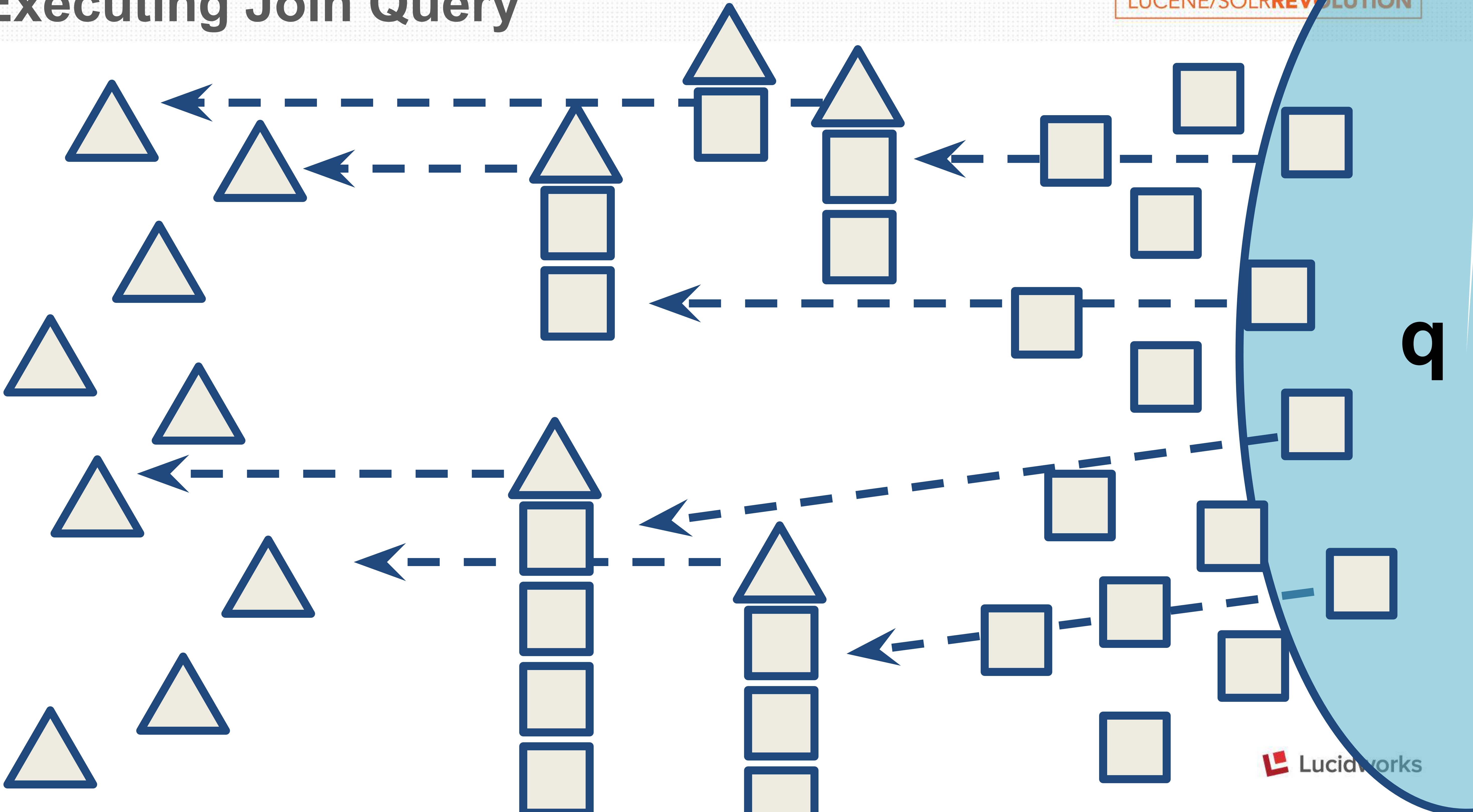
Executing Join Query



Executing Join Query

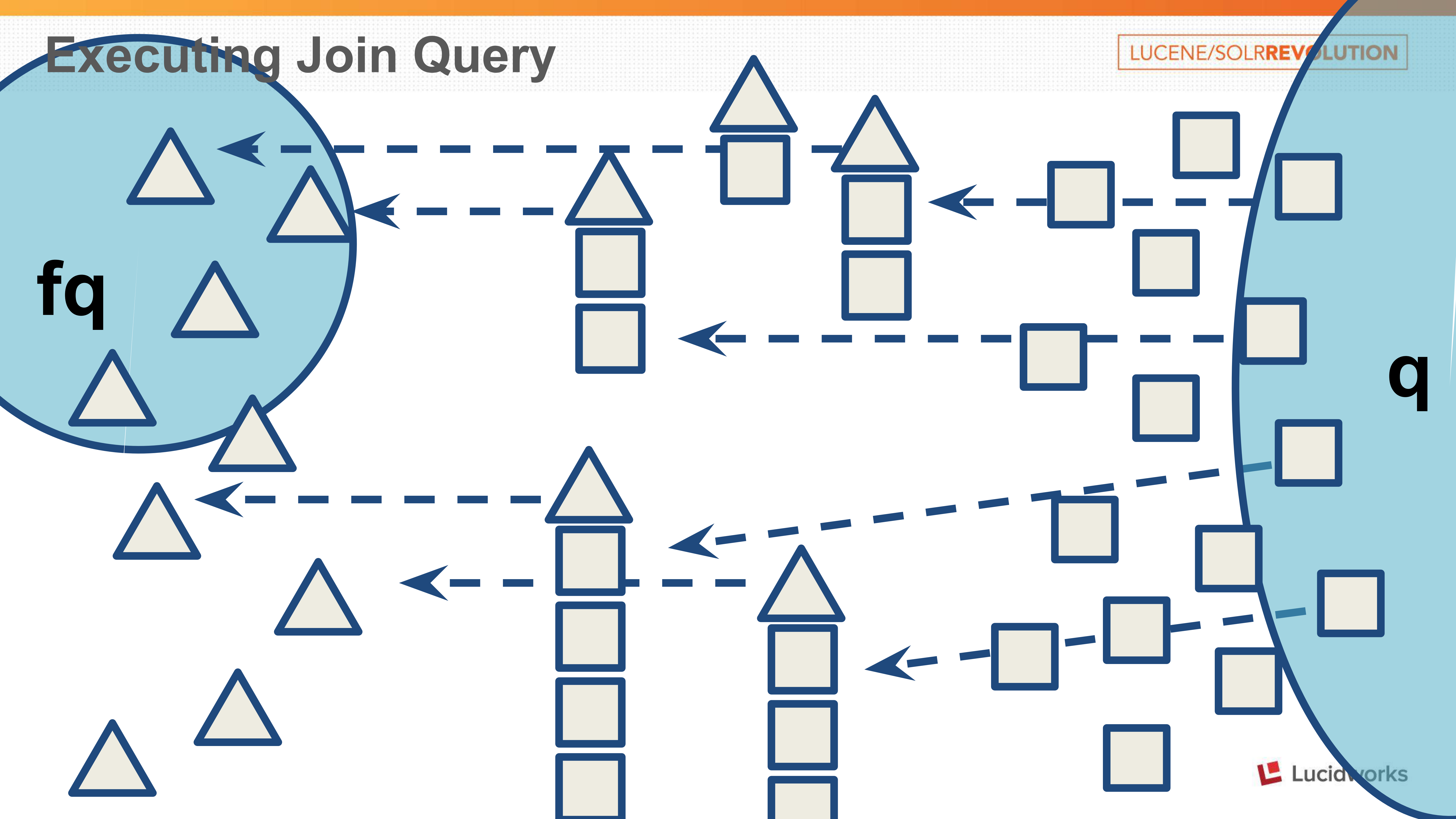


Executing Join Query



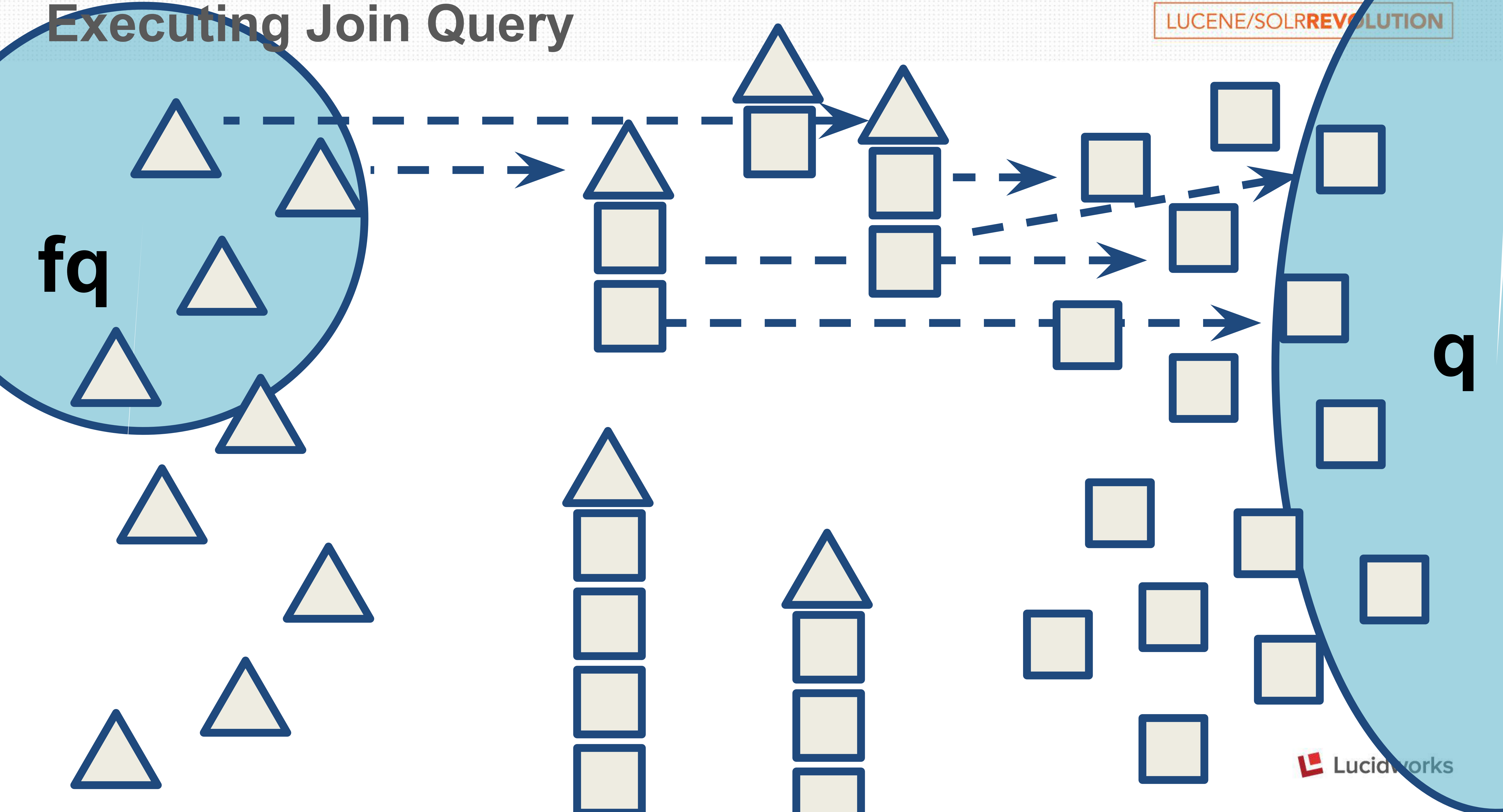
Executing Join Query

LUCENE/SOLR**REVOLUTION**



Lucidworks

Executing Join Query



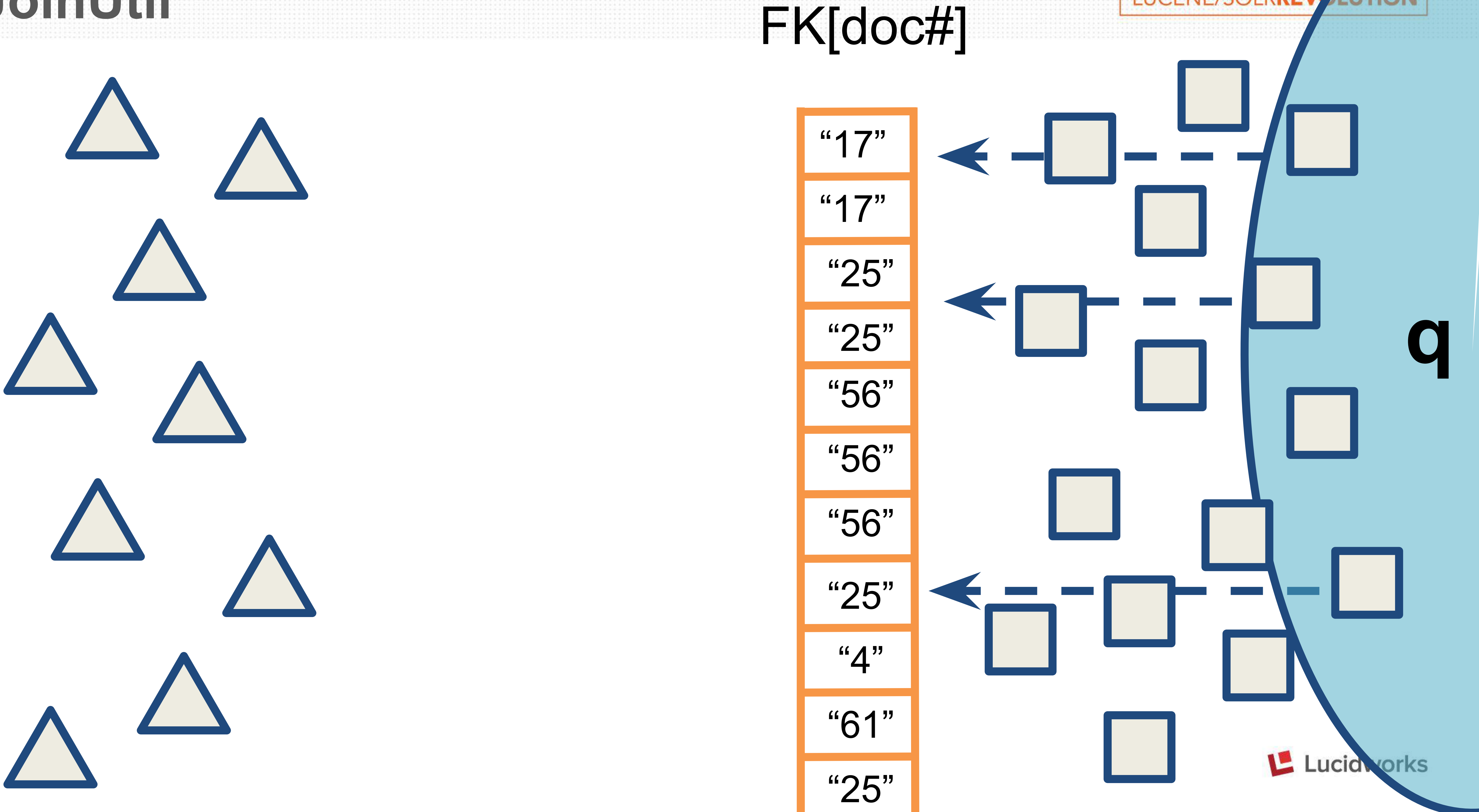
Join in General

LUCENE/SOLR**REVOLUTION**

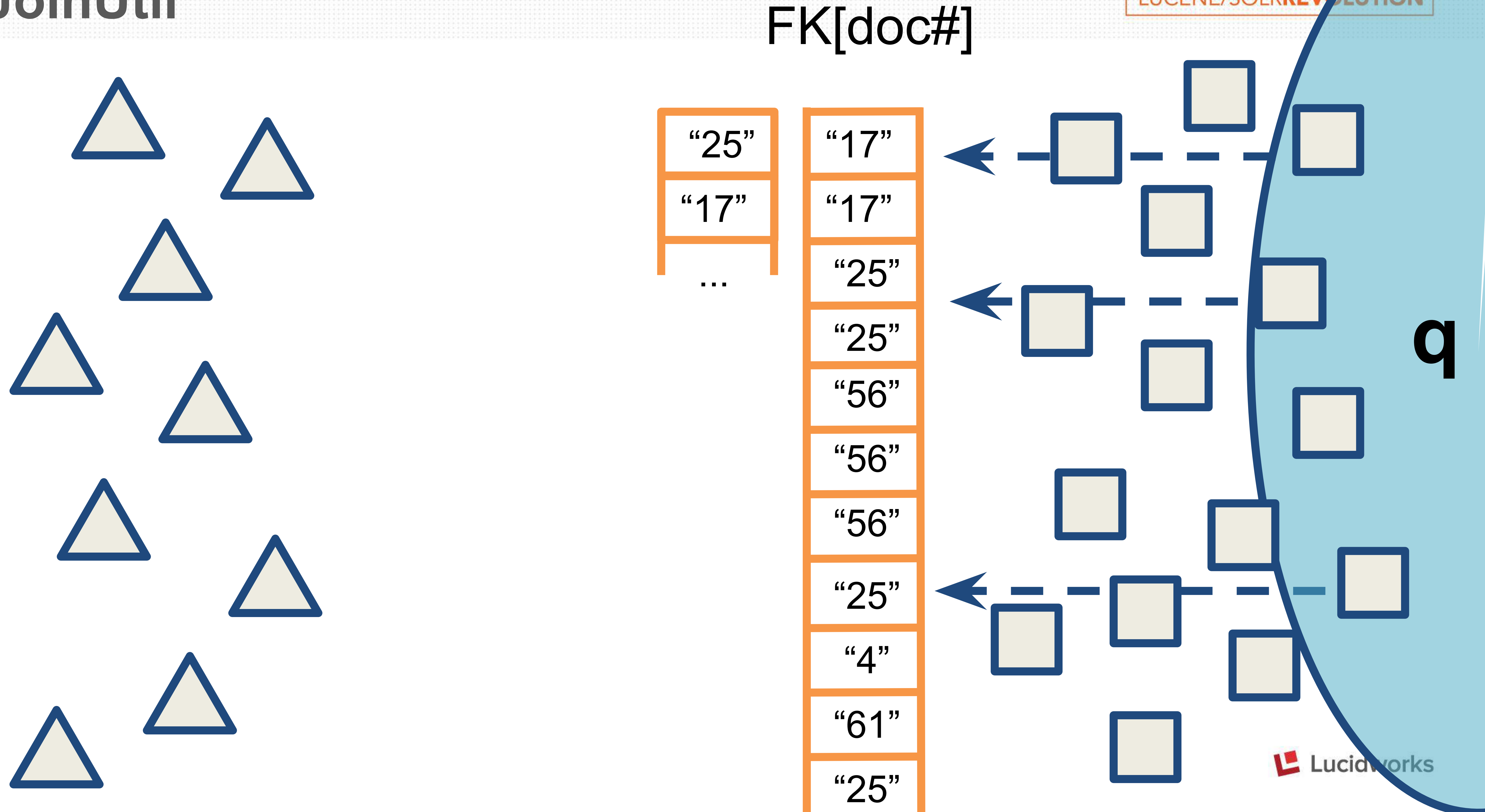
parents ∩ join-relation ∩ children

Lucidworks

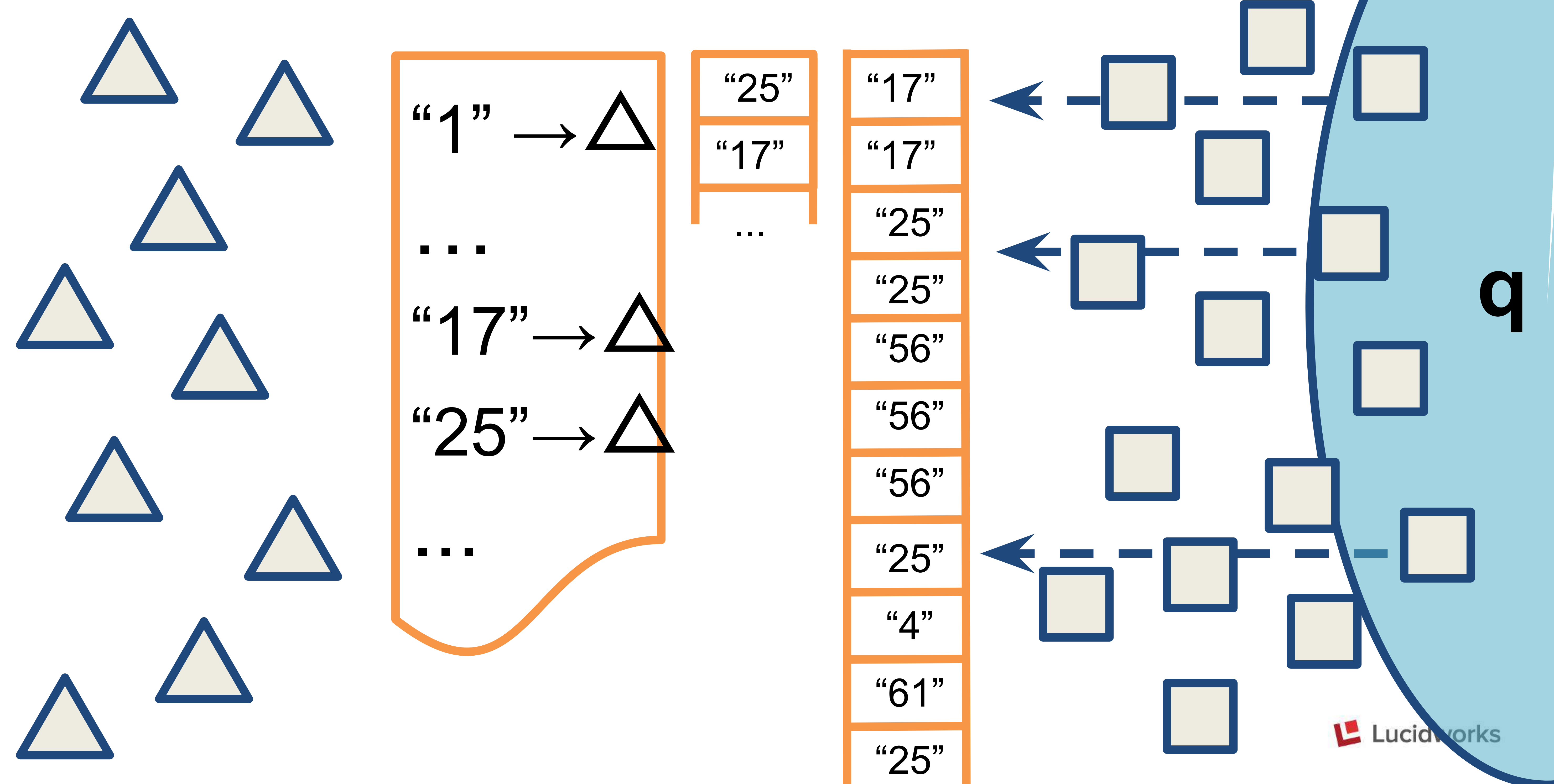
JoinUtil



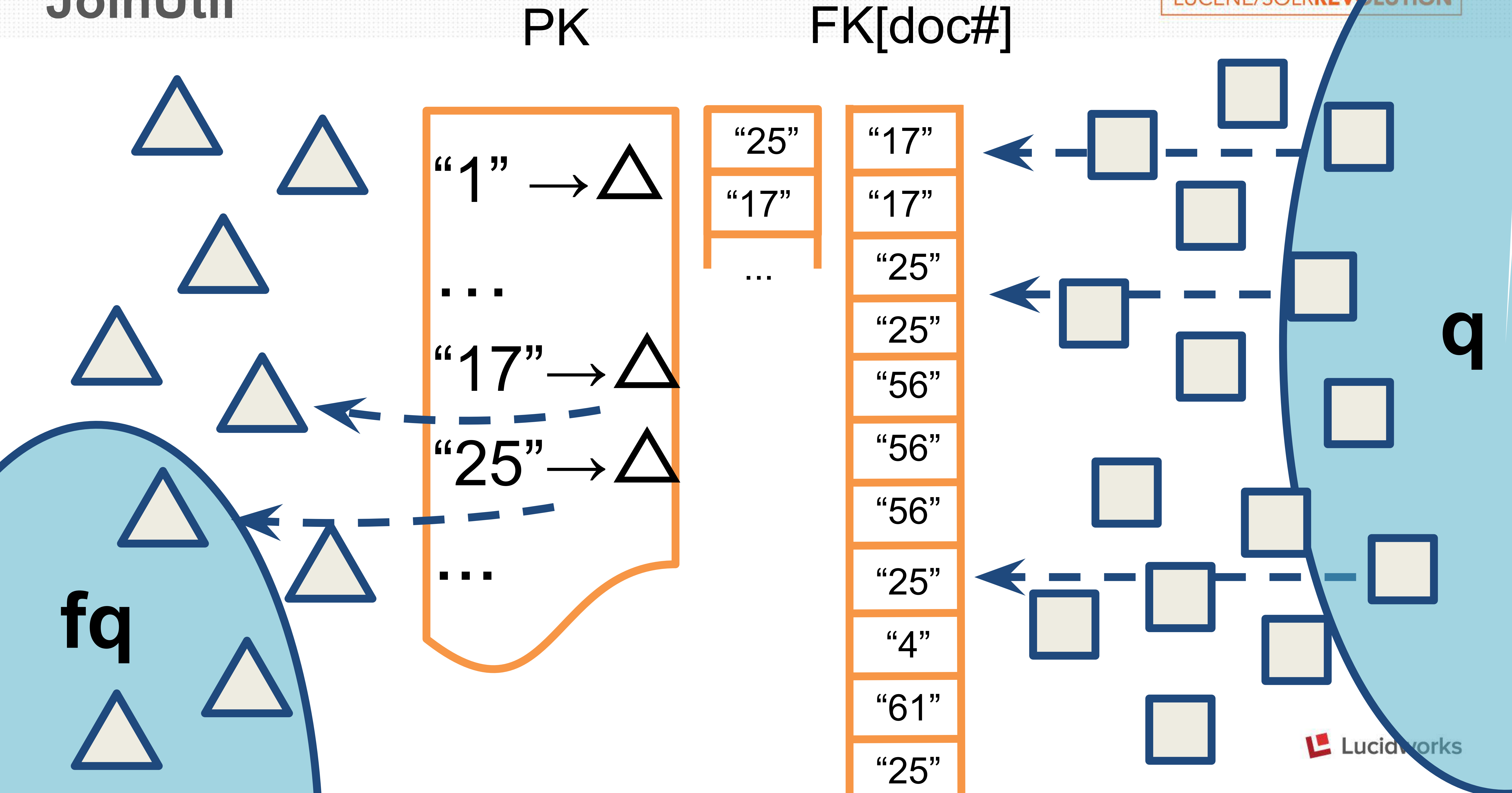
JoinUtil



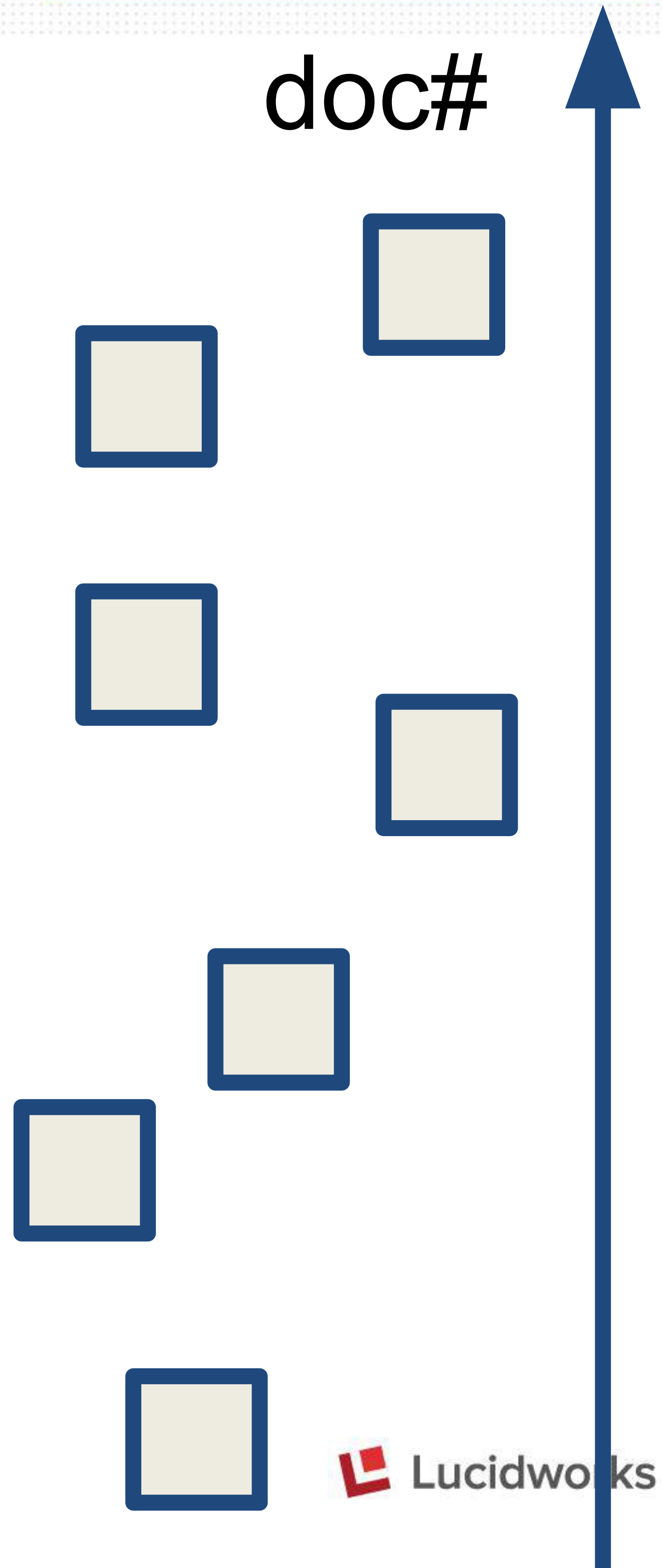
JoinUtil



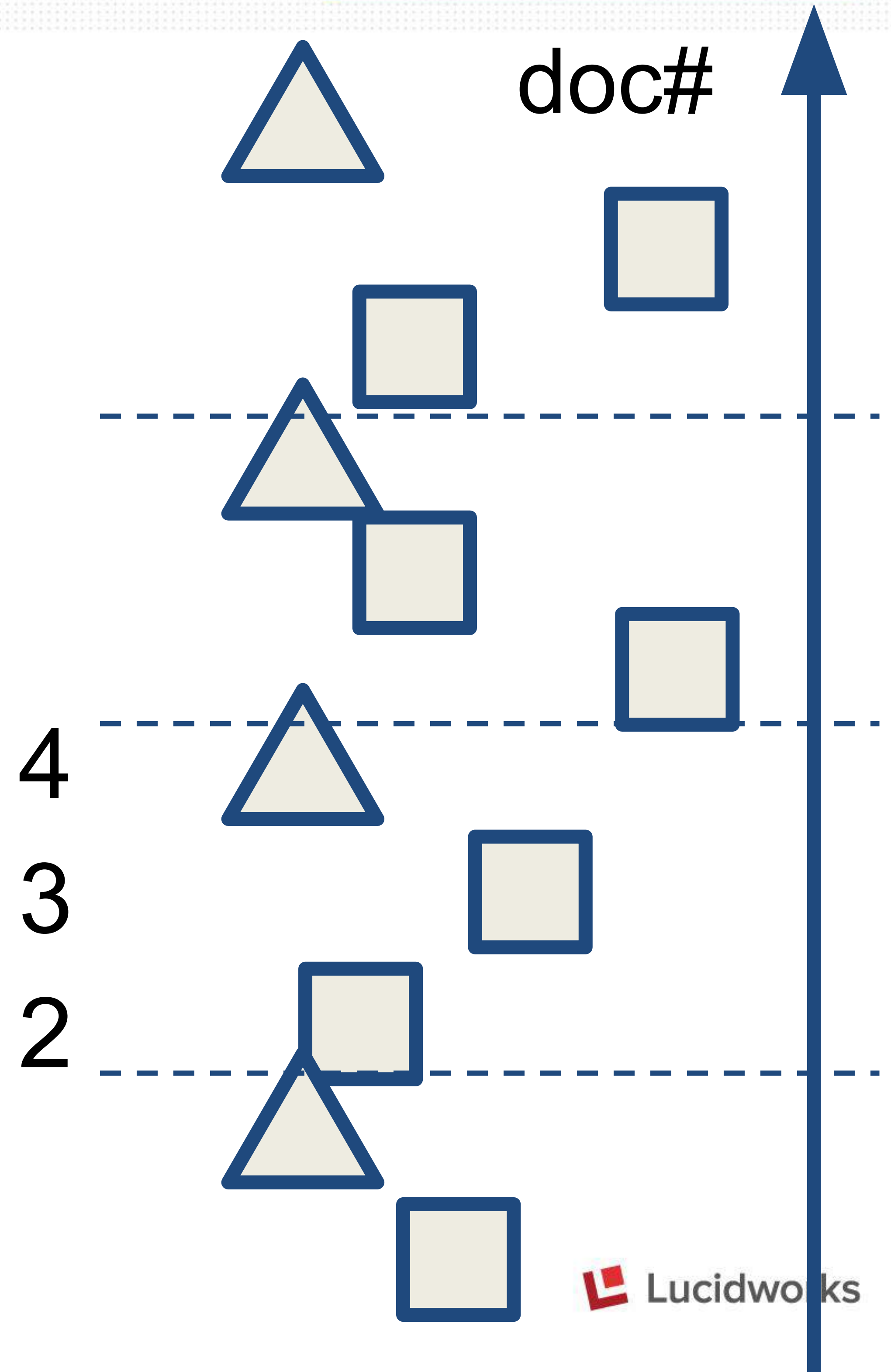
JoinUtil



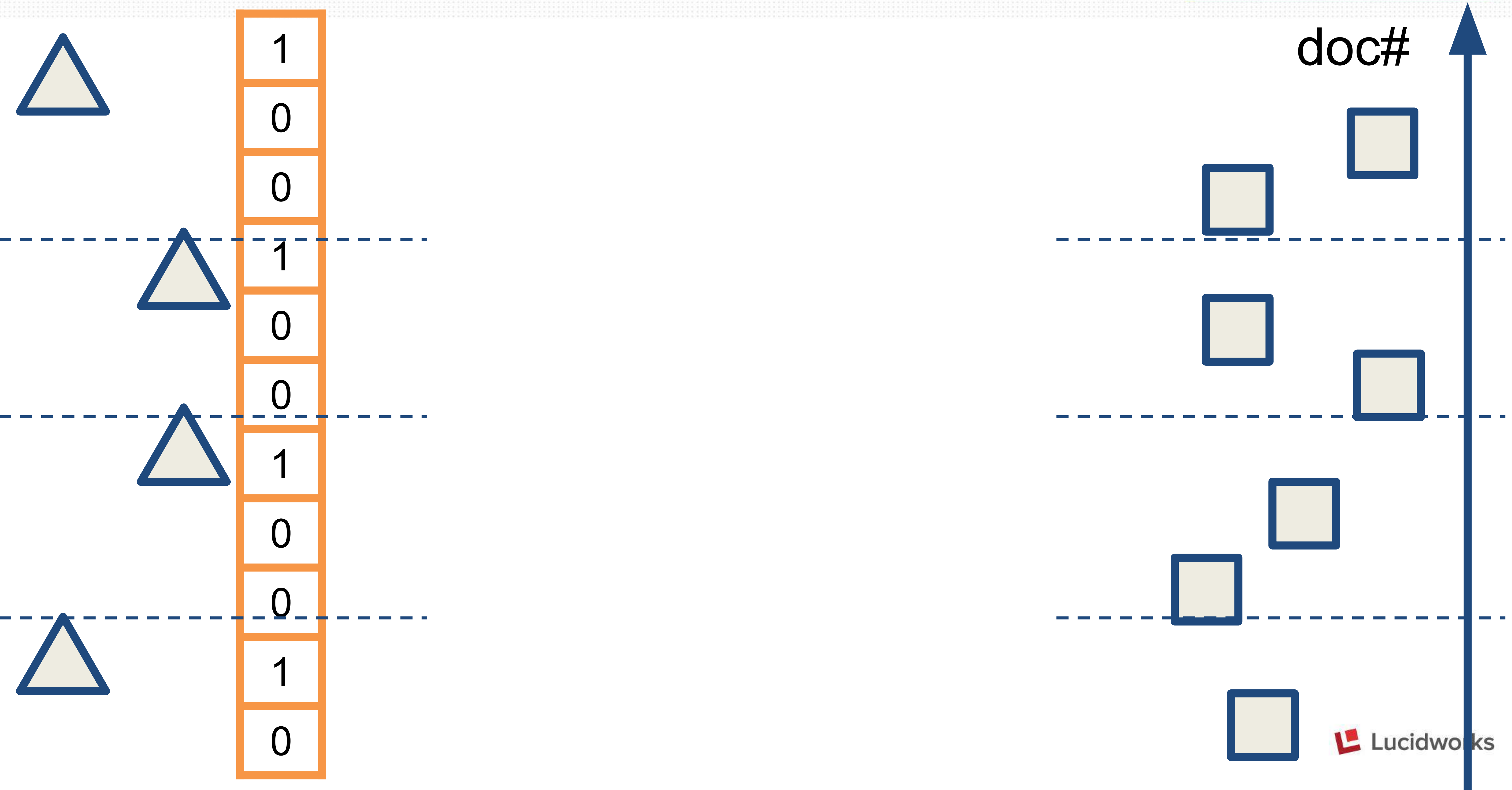
Block Join



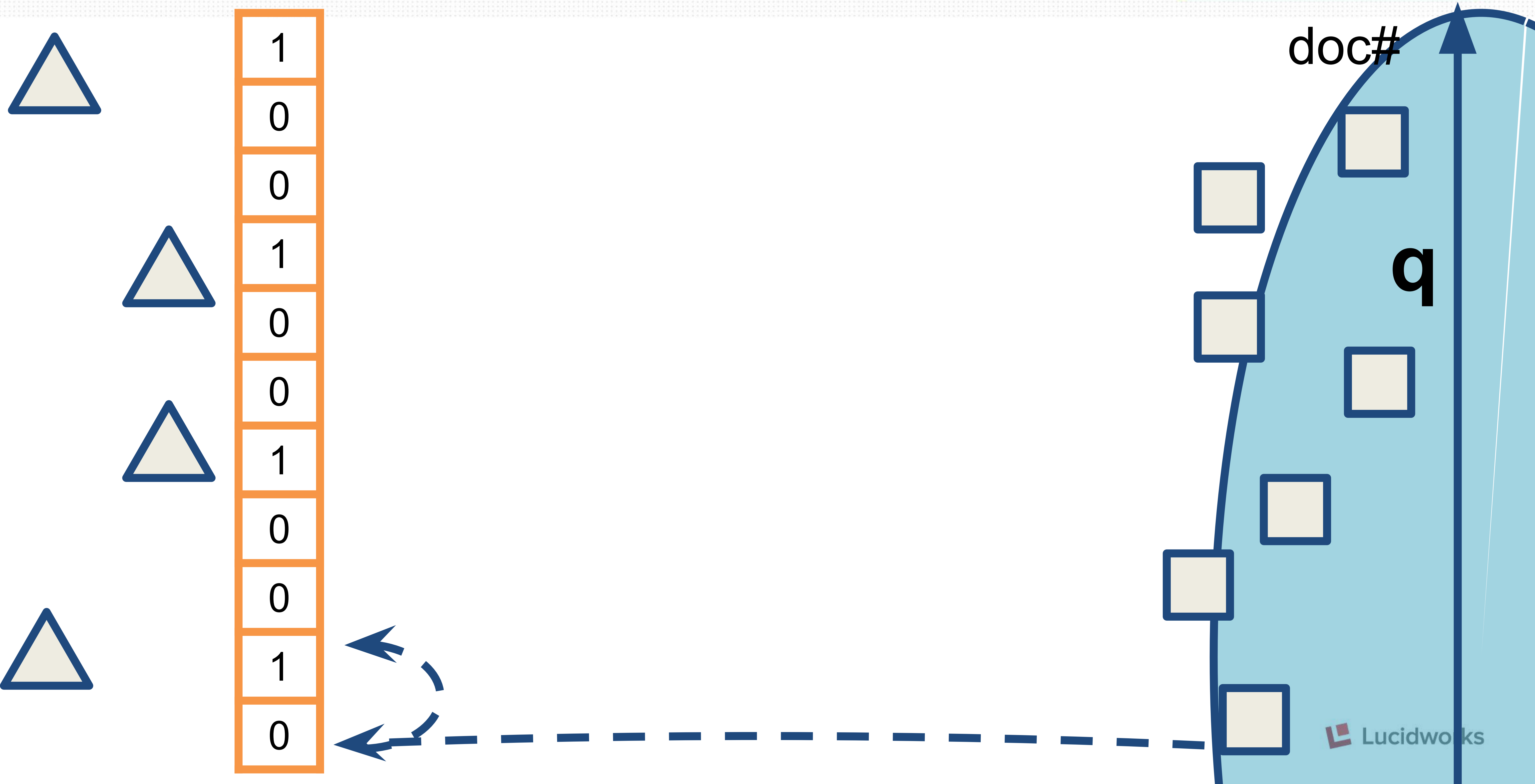
Block Join



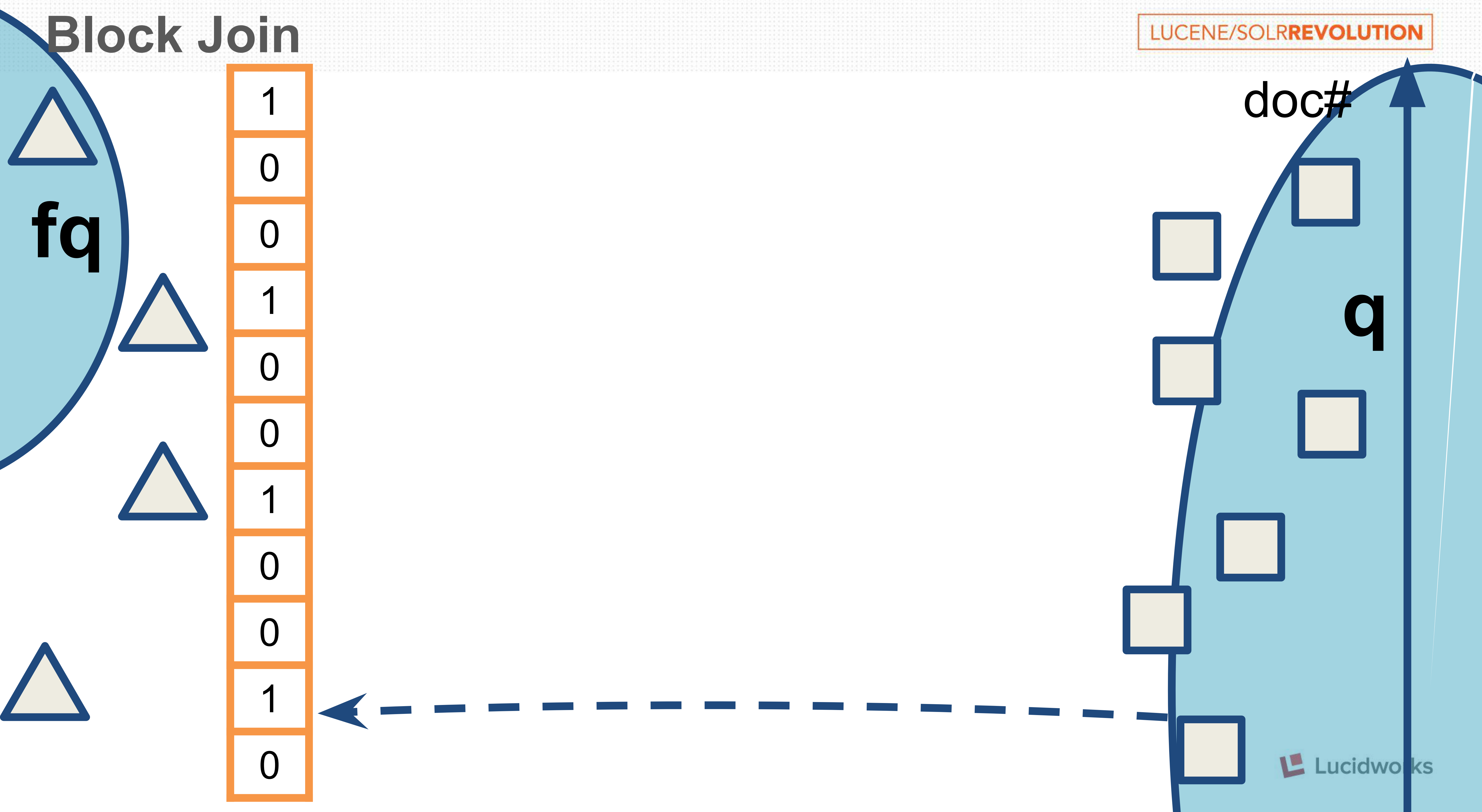
Block Join



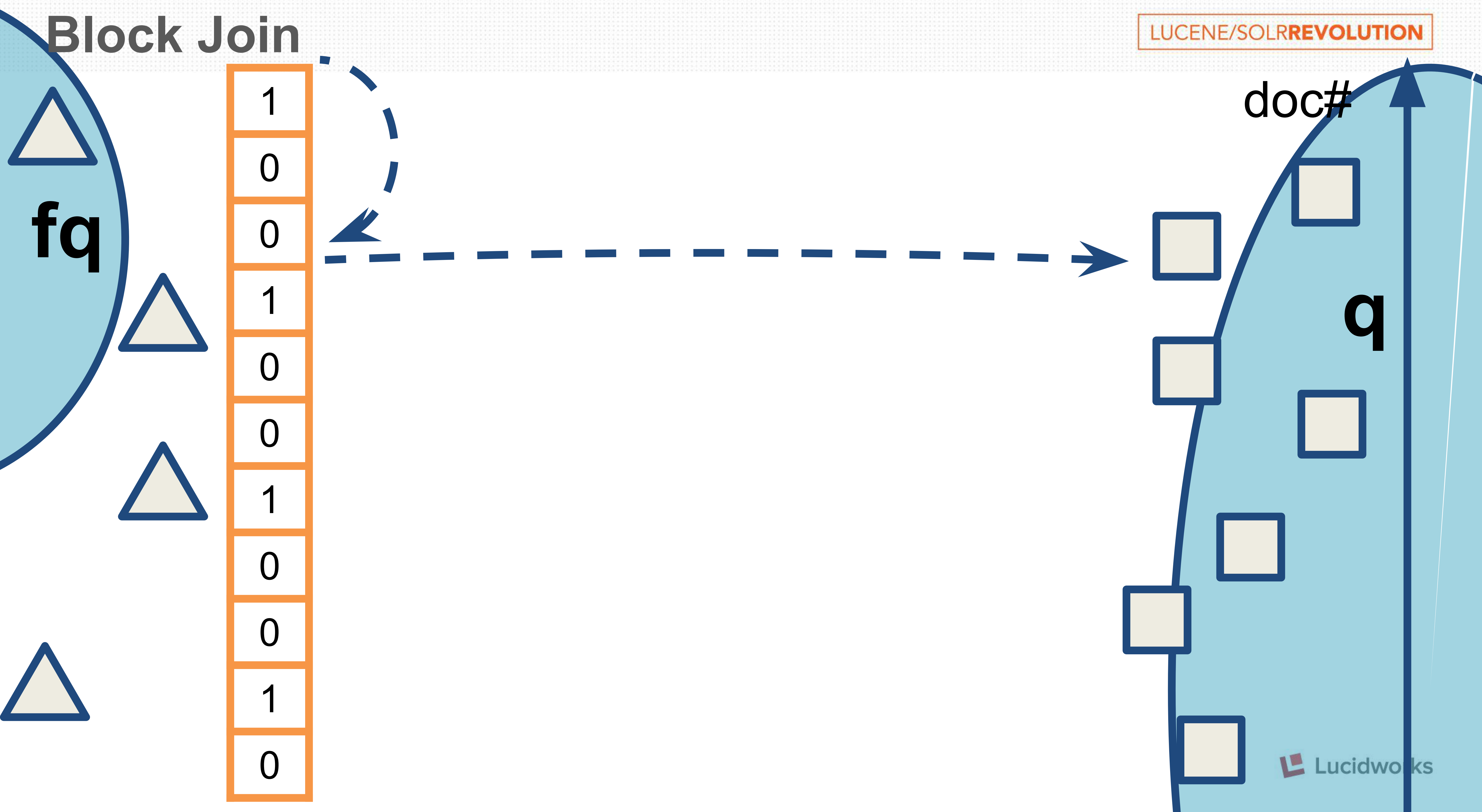
Block Join



Block Join



Block Join



Comparison

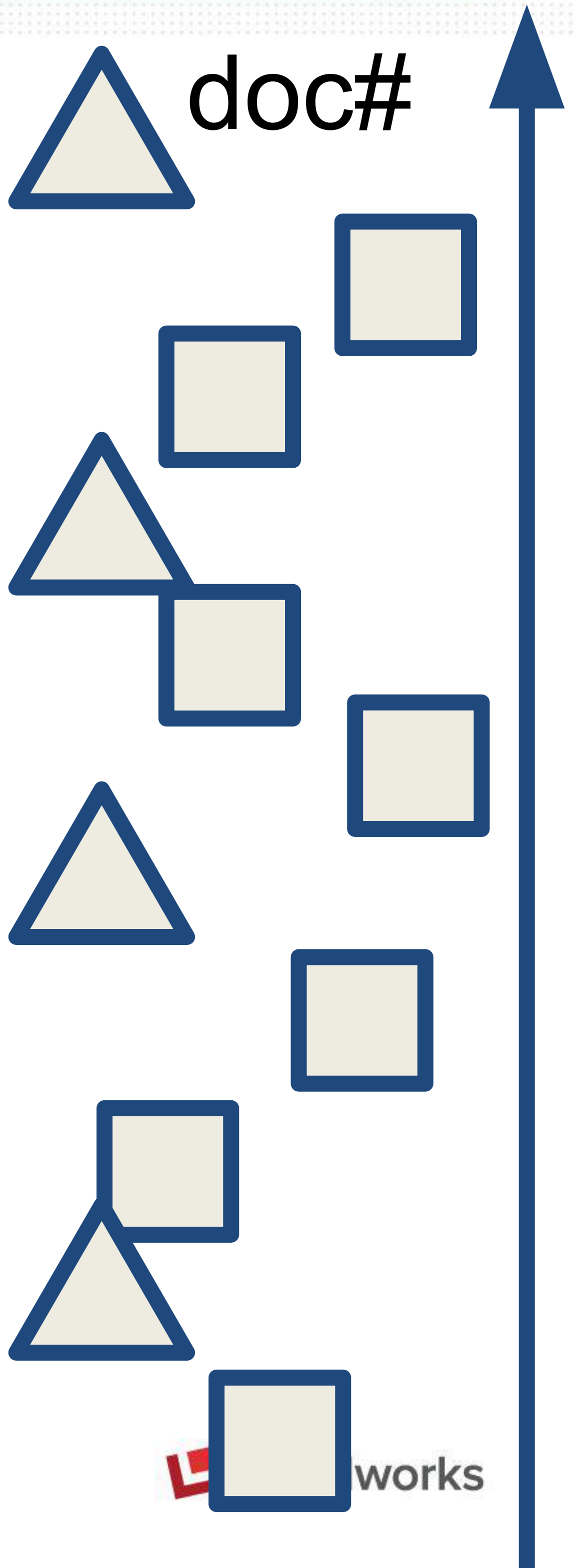
	JoinUtil	BlockJoin
searching	slow	fast
reindexing		

Comparison

	JoinUtil	BlockJoin
searching	slow	fast
reindexing	fast	slow

Comparison

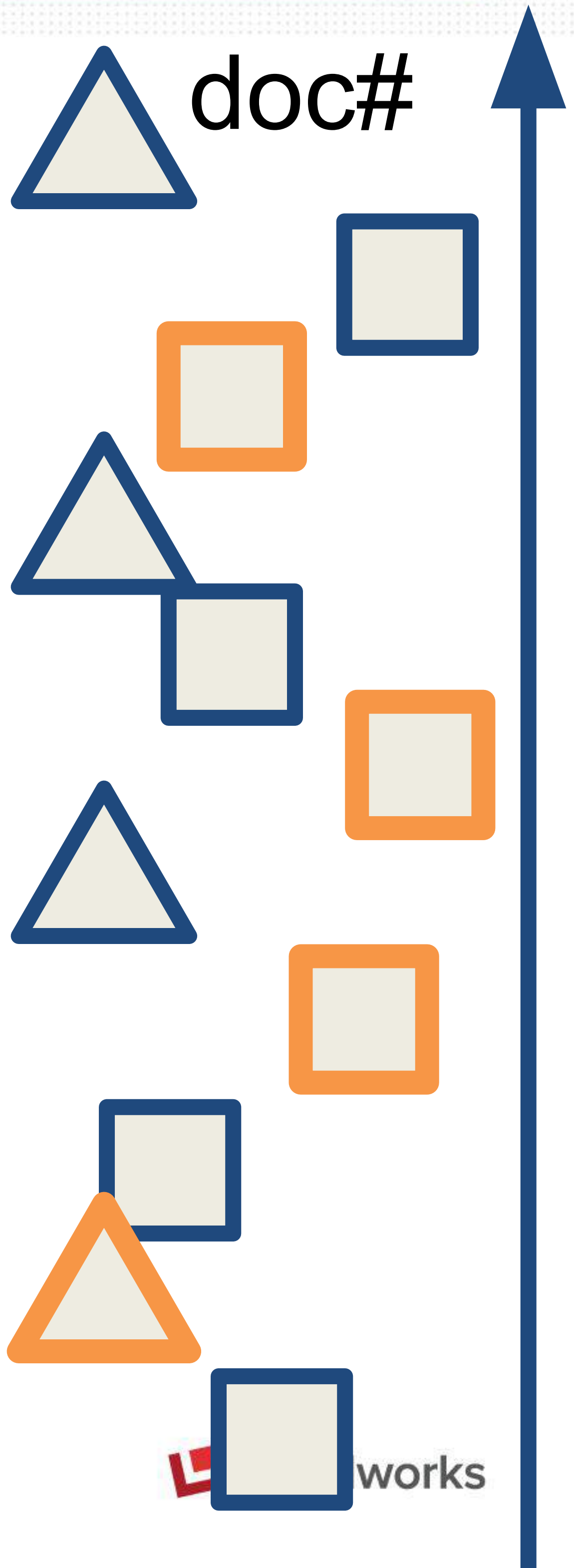
	JoinUtil	BlockJoin
searching	slow	fast
reindexing	fast	slow



Comparison

LUCENE/SOLR**REVOLUTION**

	JoinUtil	BlockJoin
searching	**slow**	**fast**
reindexing	**fast**	**slow**

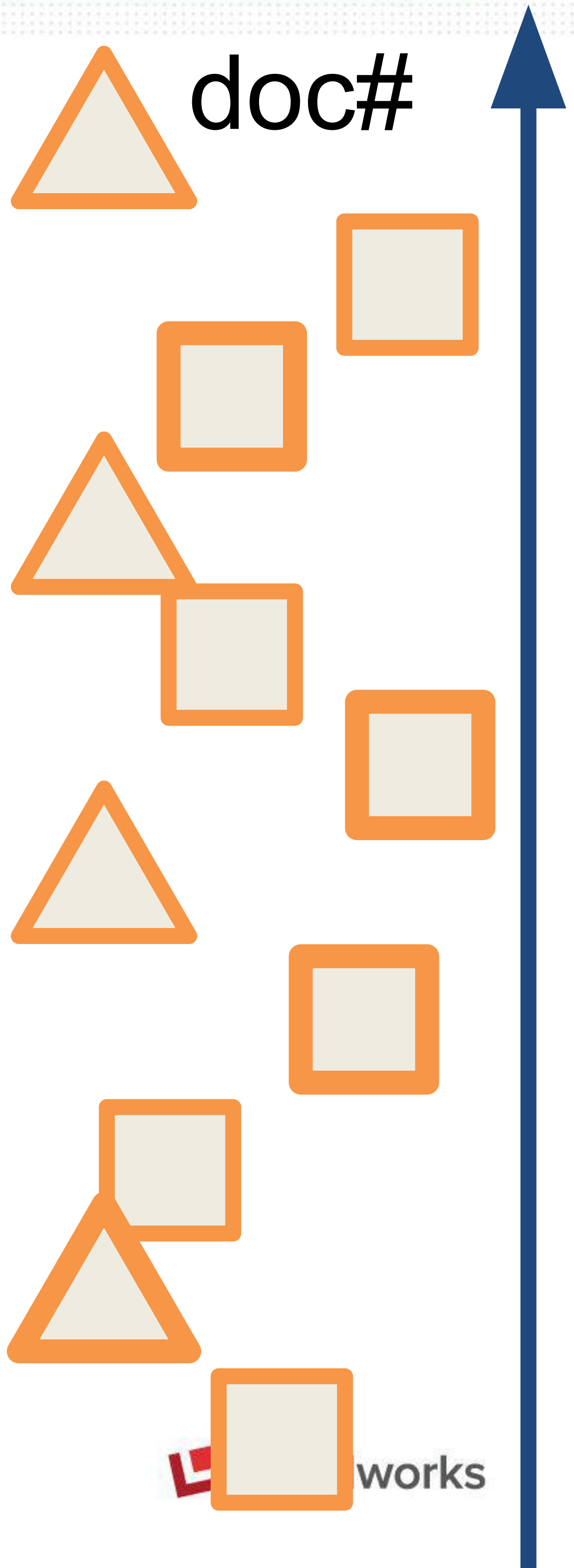


works

Comparison

LUCENE/SOLRREVOLUTION

	JoinUtil	BlockJoin
searching	slow	fast
reindexing	fast	slow

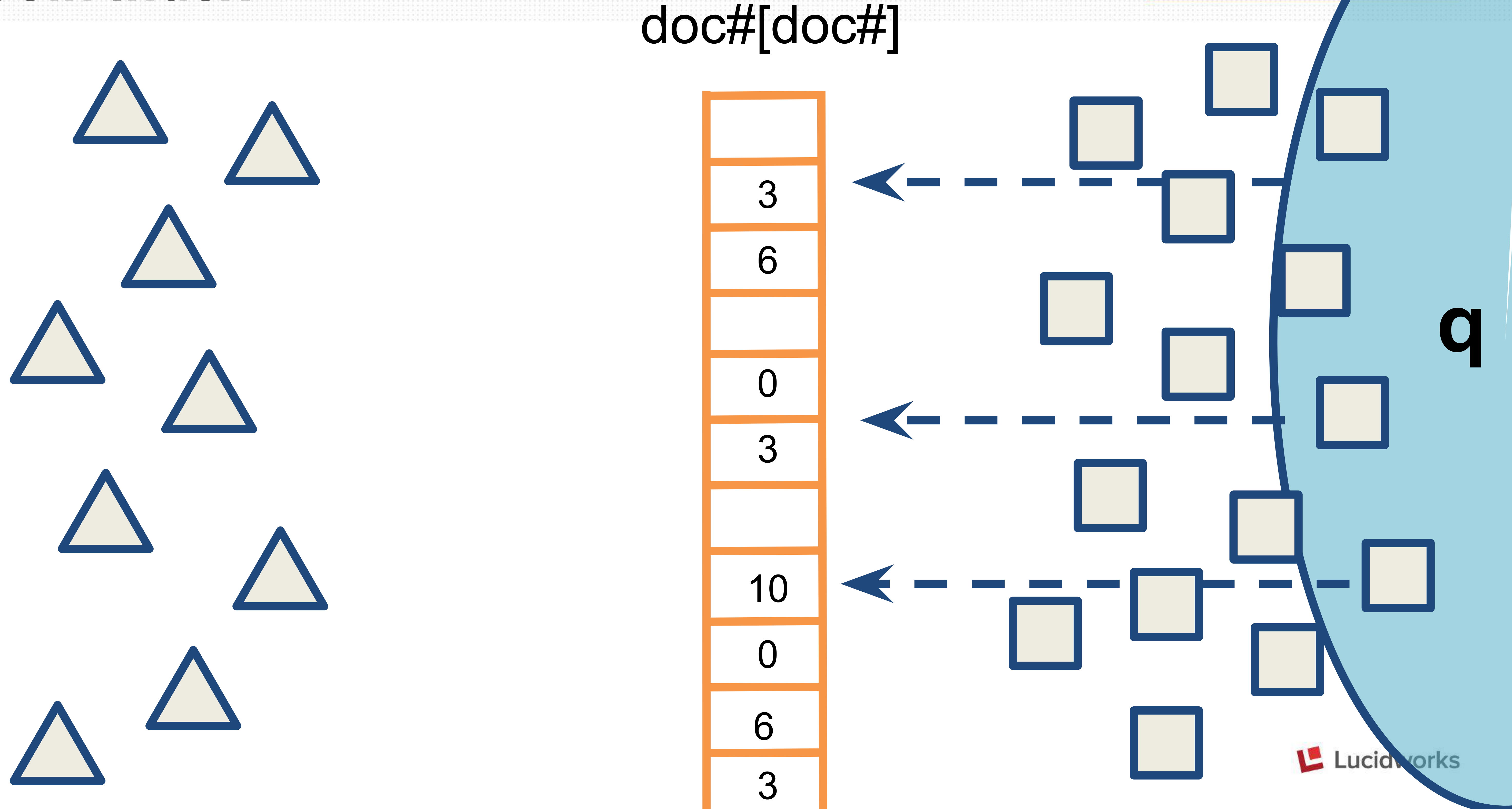


works

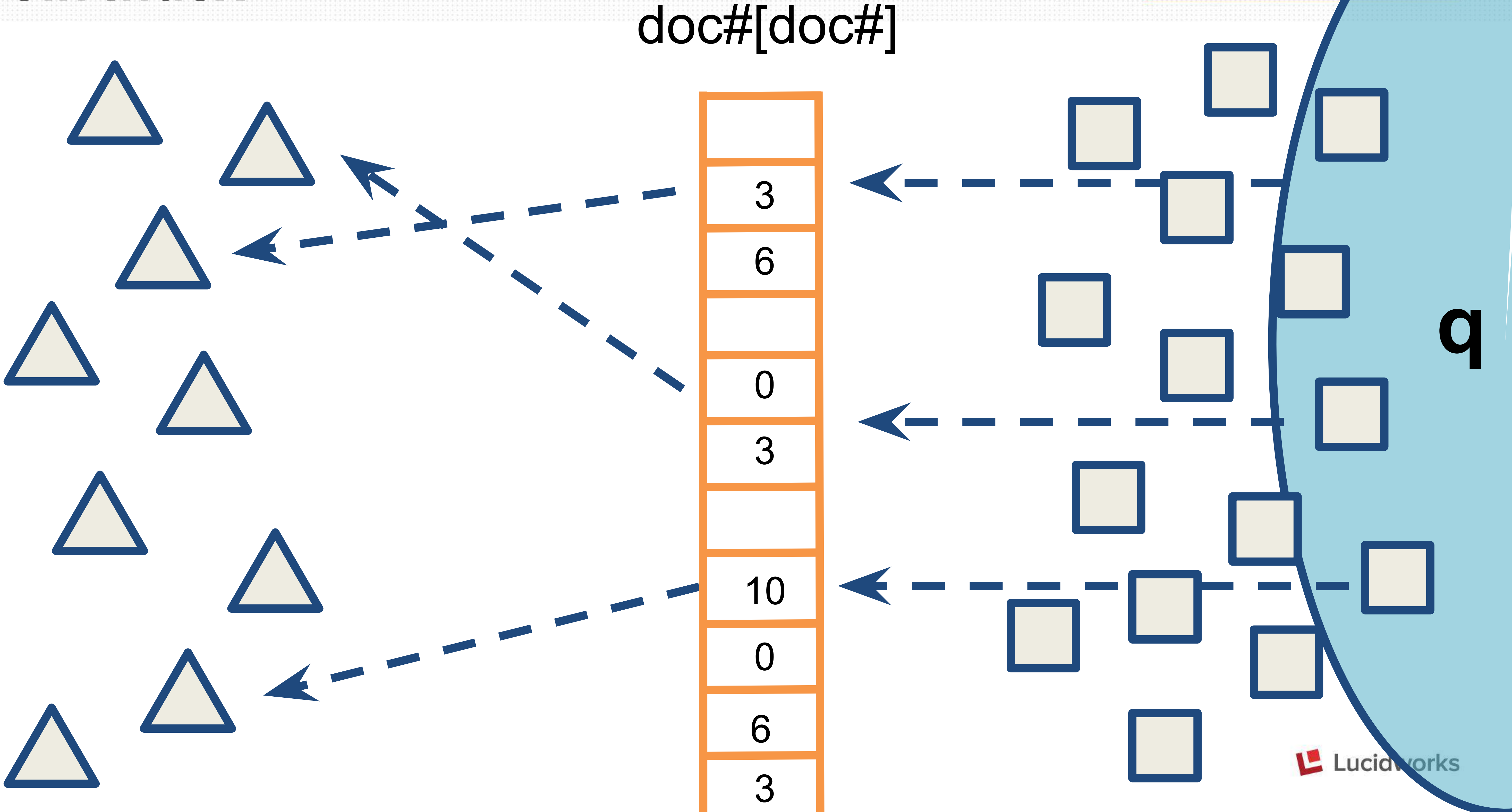
Comparison

	JoinUtil		BlockJoin
searching	slow	< ? <	fast
reindexing	fast	> ? >	slow

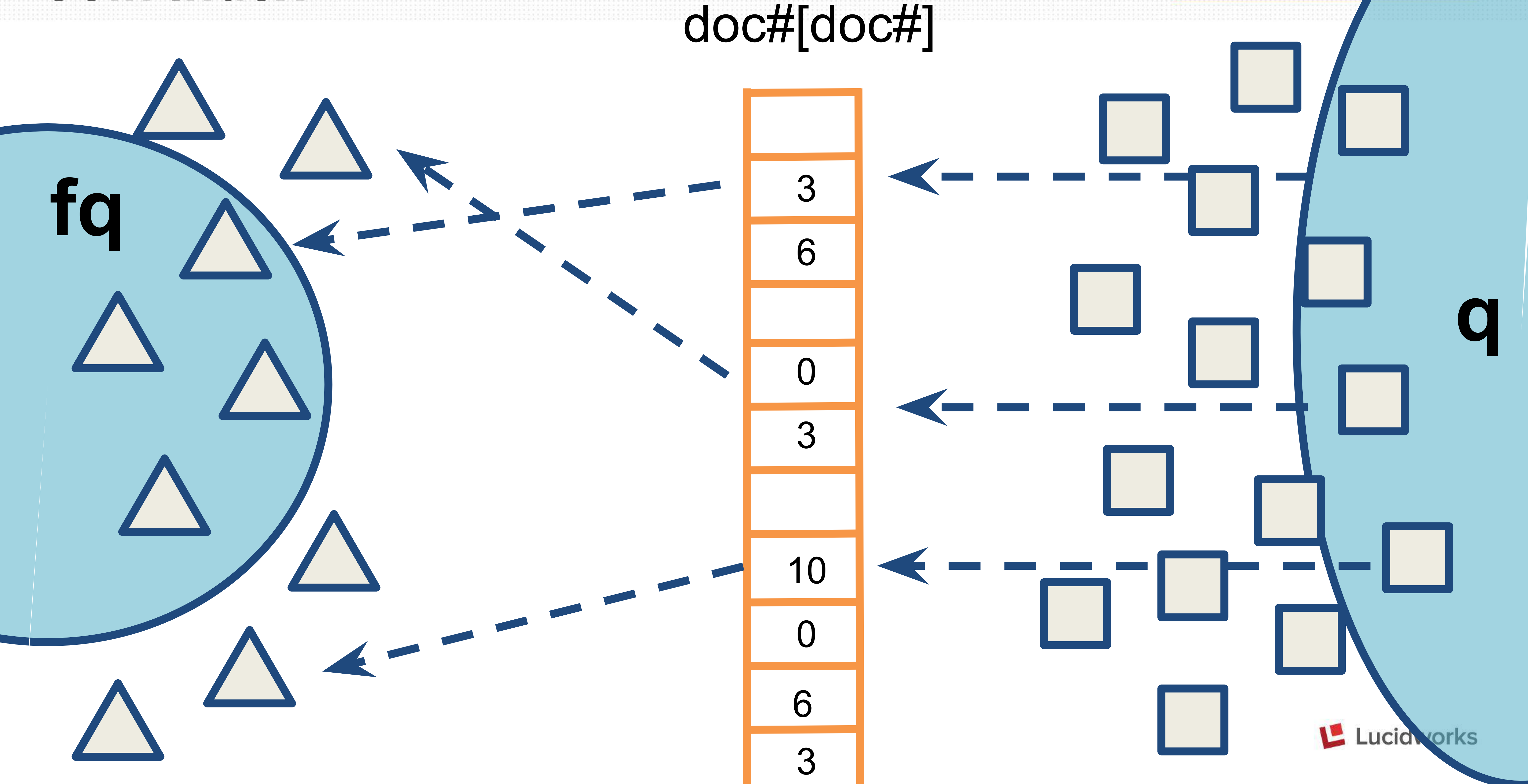
Join Index



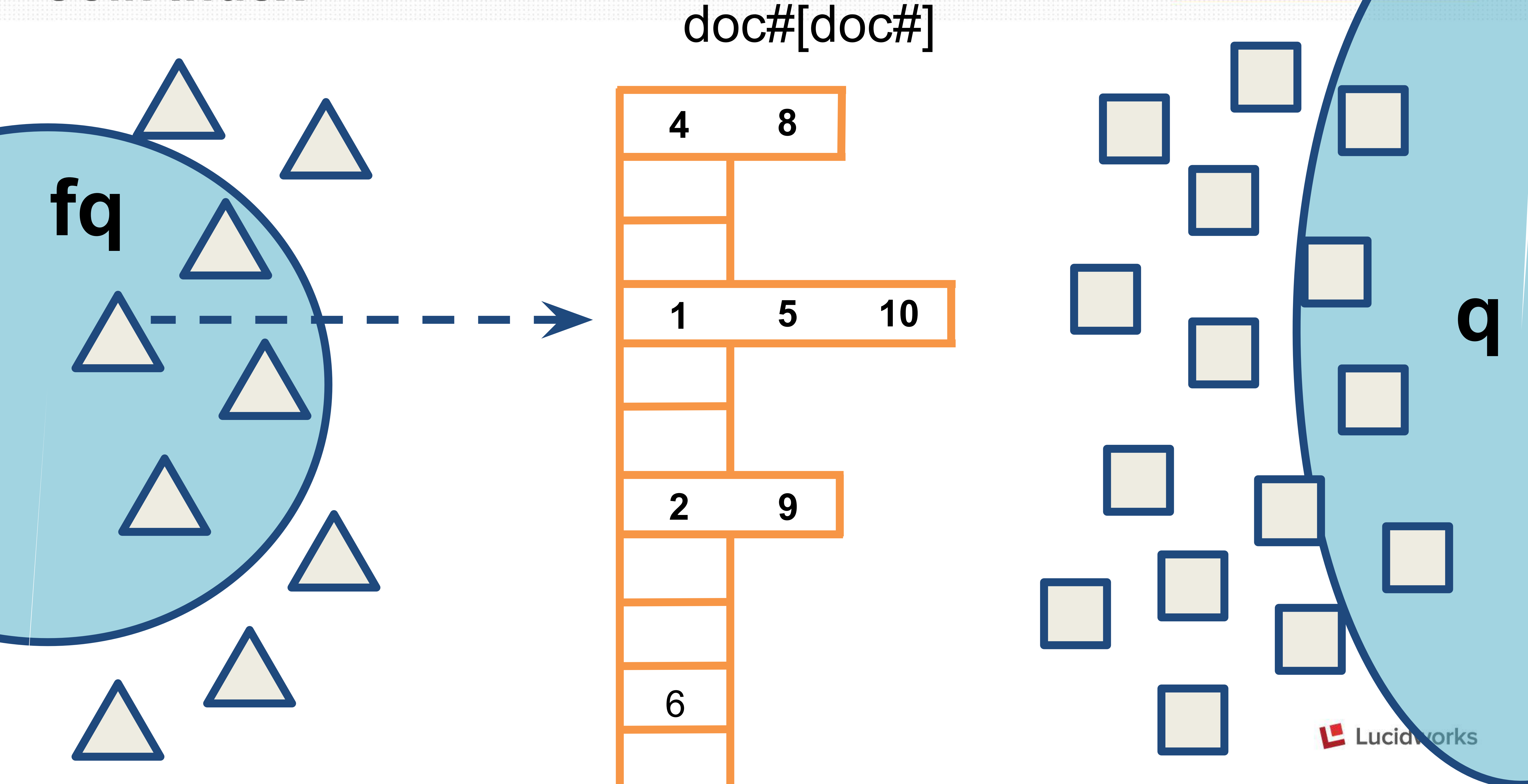
Join Index



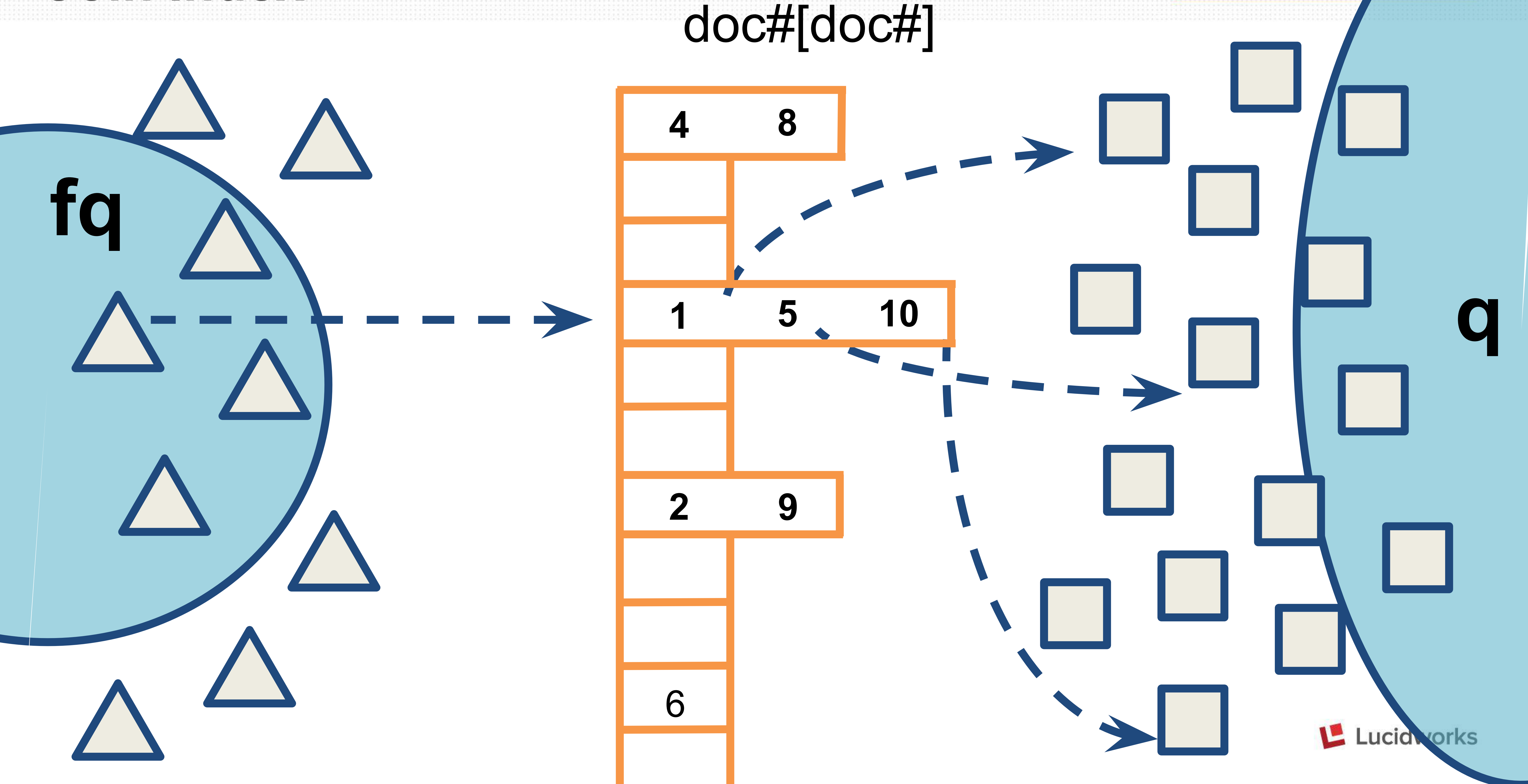
Join Index



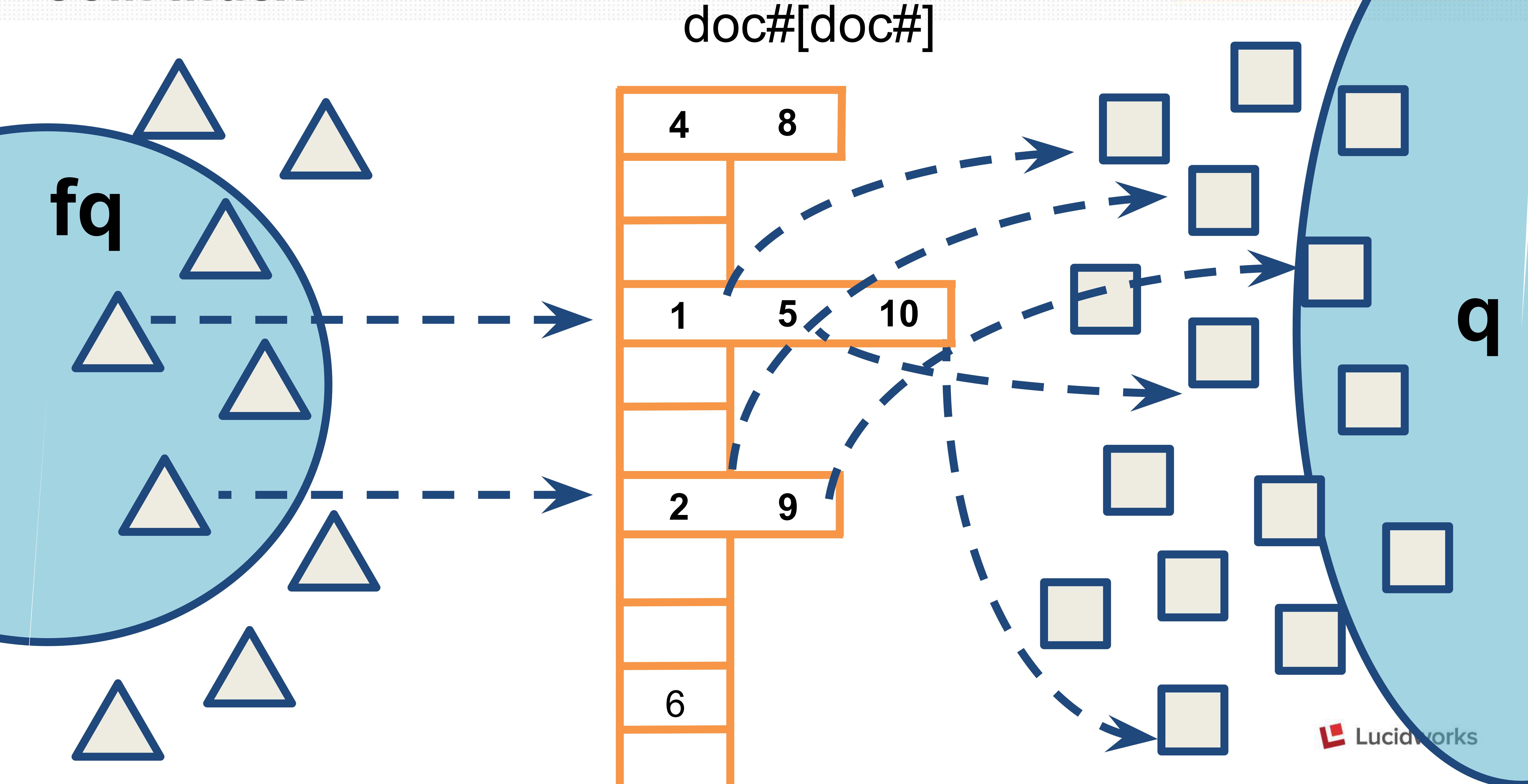
Join Index



Join Index

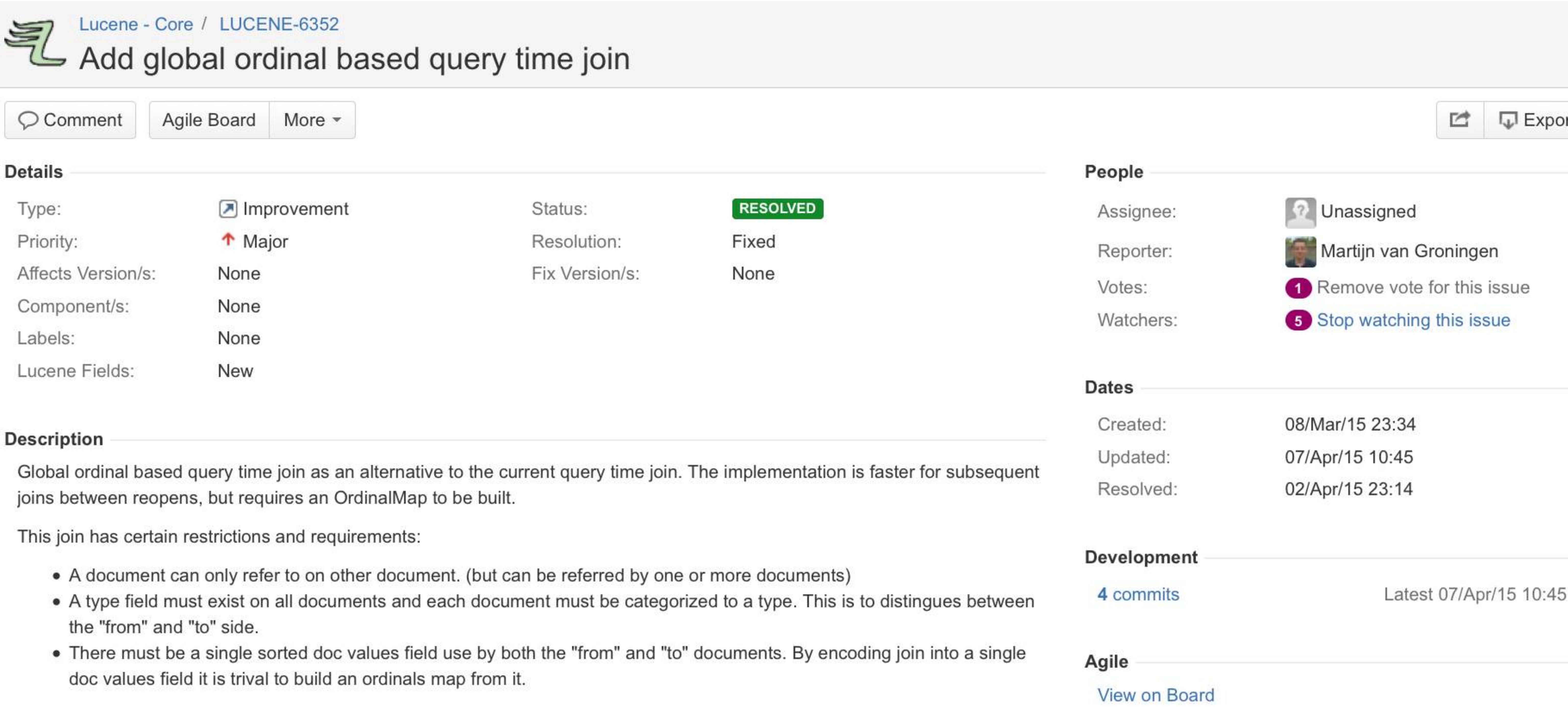


Join Index



meanwhile… in LUCENE-6352

LUCENE/SOLRREVOLUTION



Lucene - Core / LUCENE-6352

Add global ordinal based query time join

Details

Type:	Improvement	Status:	RESOLVED
Priority:	Major	Resolution:	Fixed
Affects Version/s:	None	Fix Version/s:	None
Component/s:	None		
Labels:	None		
Lucene Fields:	New		

People

Assignee:	Unassigned
Reporter:	Martijn van Groningen
Votes:	1 Remove vote for this issue
Watchers:	5 Stop watching this issue

Dates

Created:	08/Mar/15 23:34
Updated:	07/Apr/15 10:45
Resolved:	02/Apr/15 23:14

Development

4 commits Latest 07/Apr/15 10:45

Agile

View on Board

Description

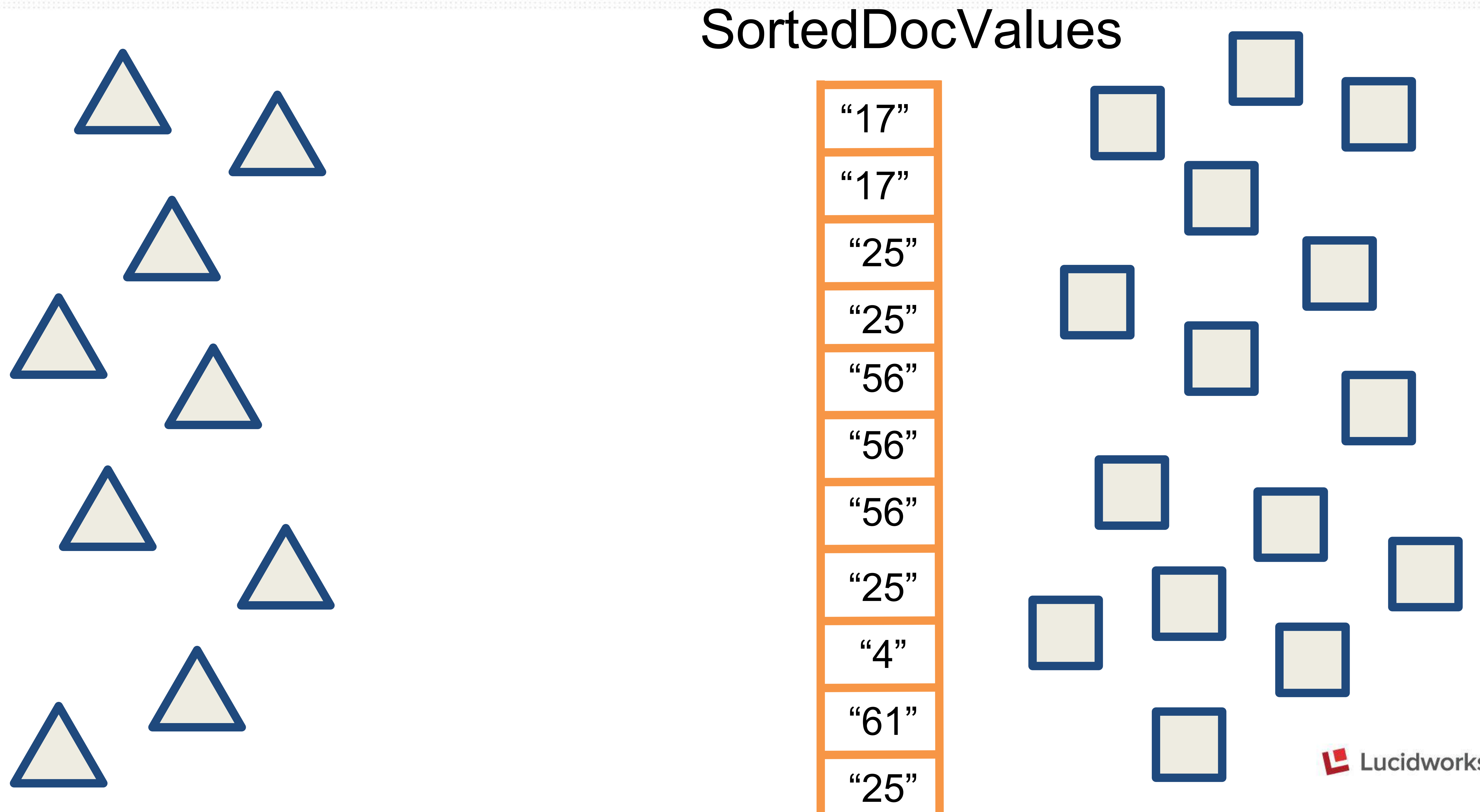
Global ordinal based query time join as an alternative to the current query time join. The implementation is faster for subsequent joins between reopens, but requires an OrdinalMap to be built.

This join has certain restrictions and requirements:

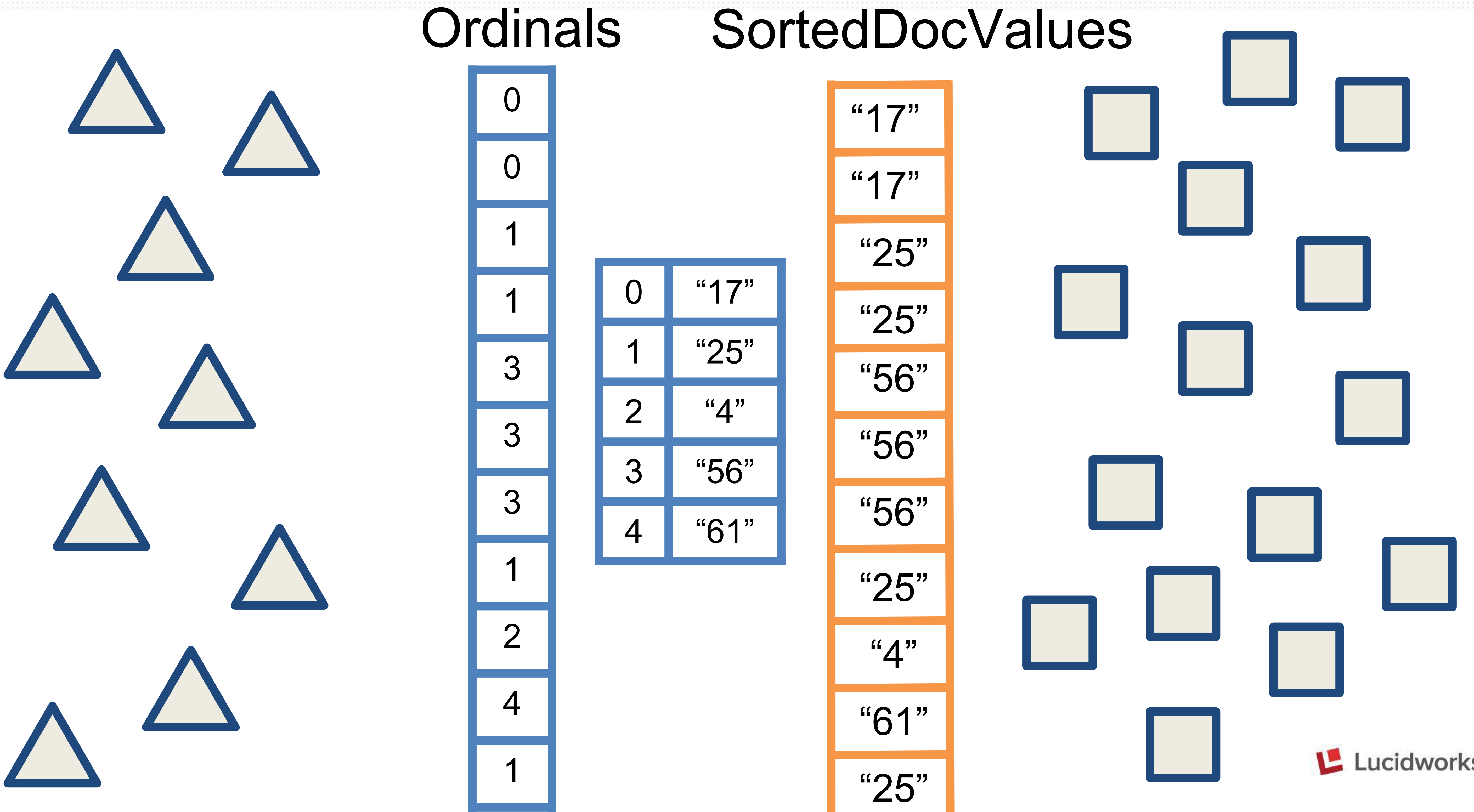
- A document can only refer to on other document. (but can be referred by one or more documents)
- A type field must exist on all documents and each document must be categorized to a type. This is to distingues between the "from" and "to" side.
- There must be a single sorted doc values field use by both the "from" and "to" documents. By encoding join into a single doc values field it is trival to build an ordinals map from it.

Lucidworks

GlobalOrdinalsQuery

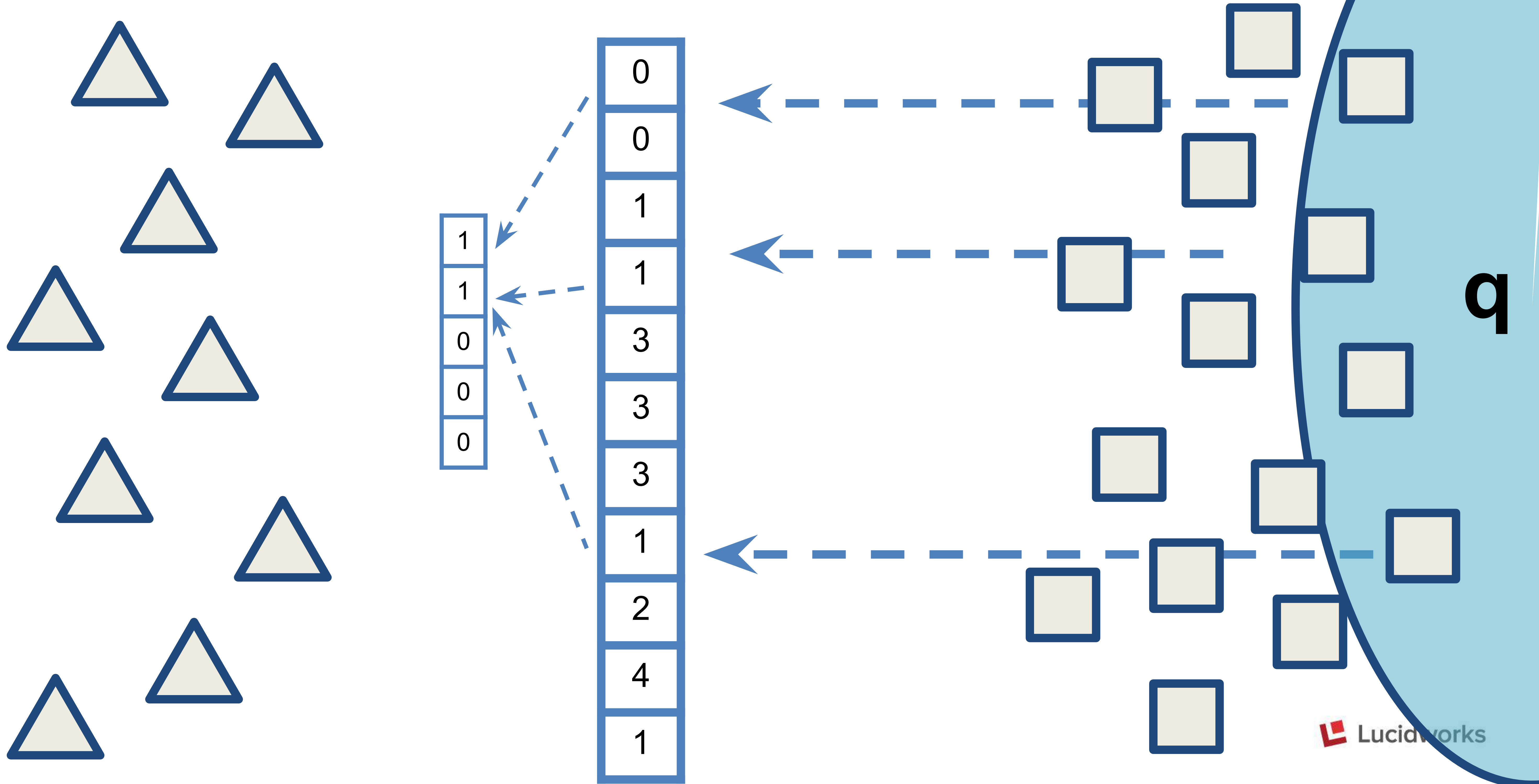


GlobalOrdinalsQuery

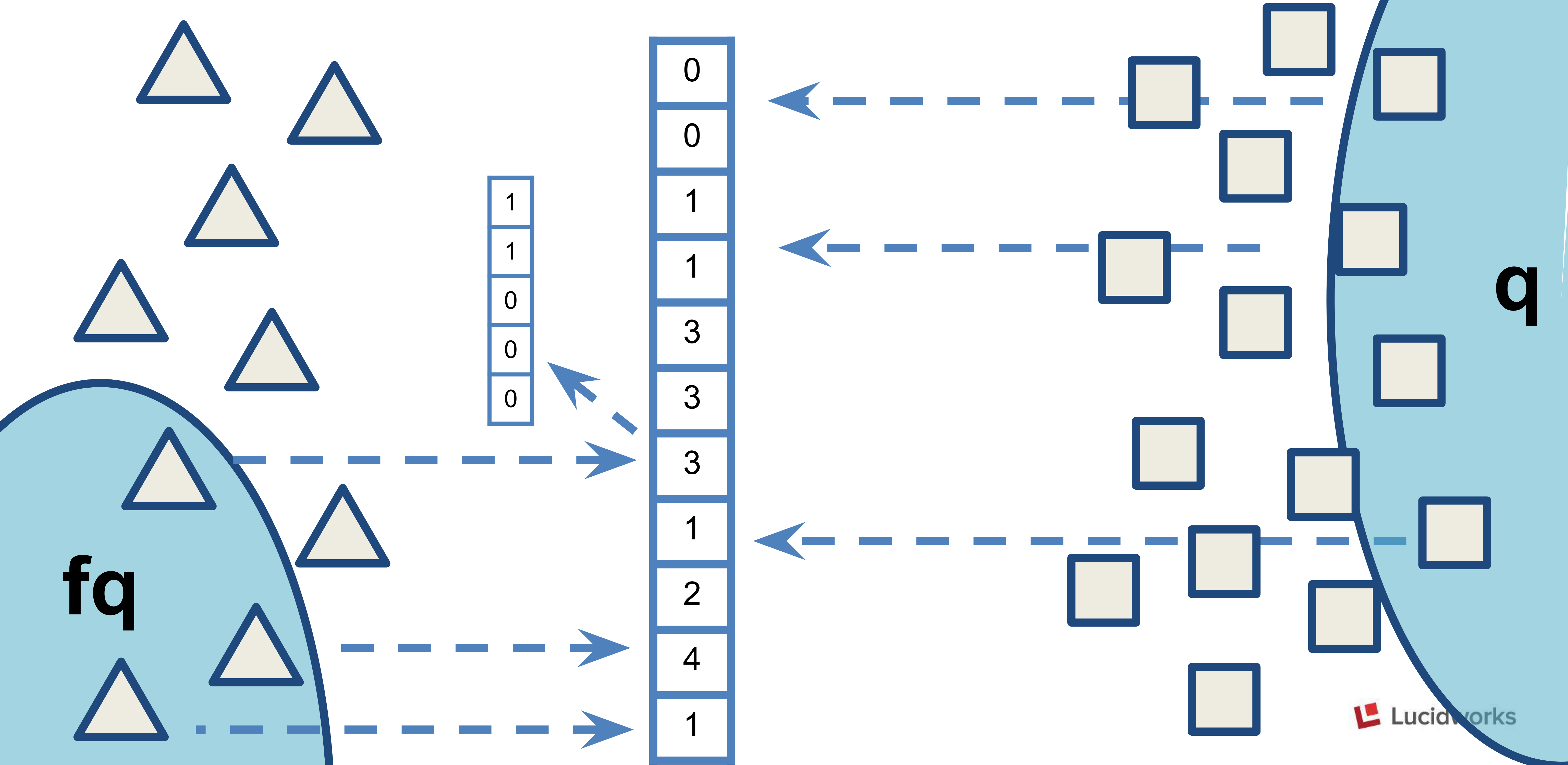


0	"17"
1	"25"
2	"4"
3	"56"
4	"61"

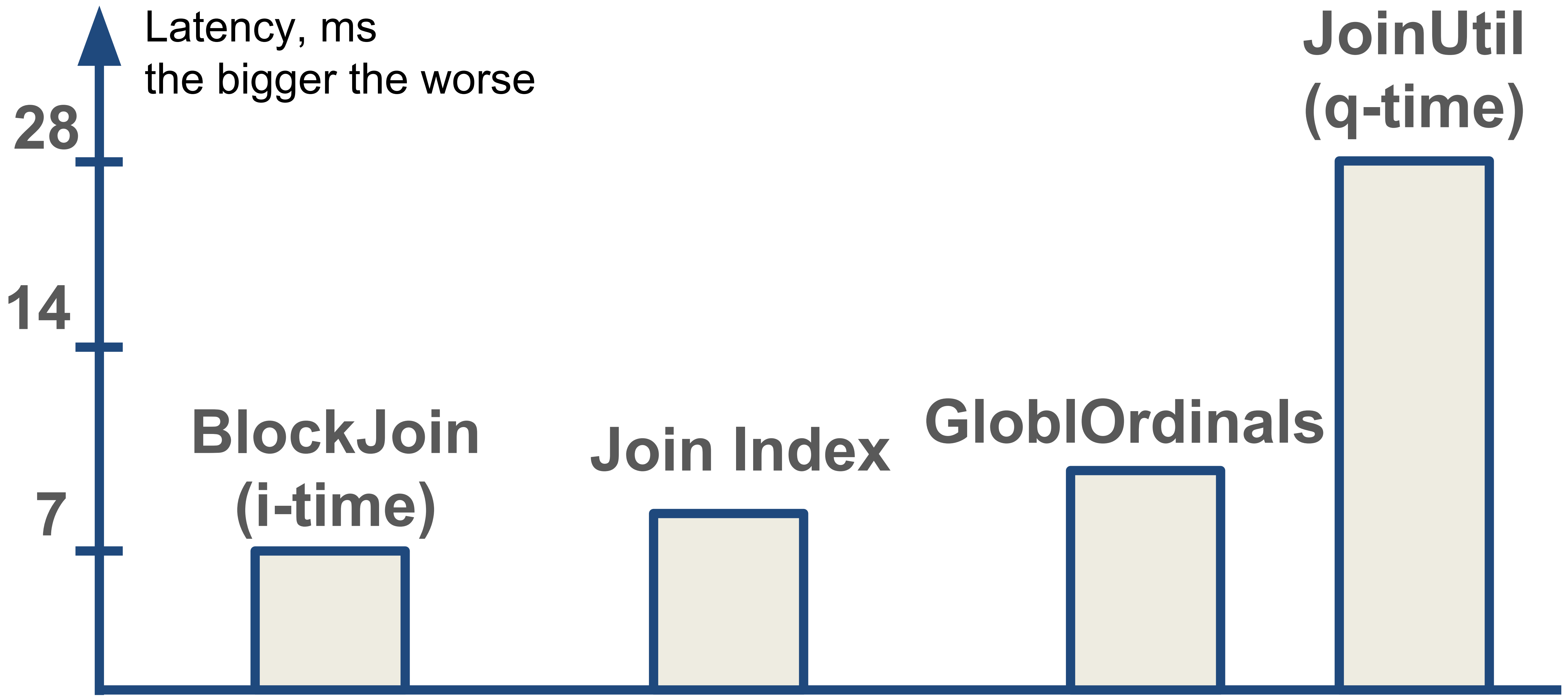
GlobalOrdinalsQuery



GlobalOrdinalsQuery



Benchmarking 2.9 M docs

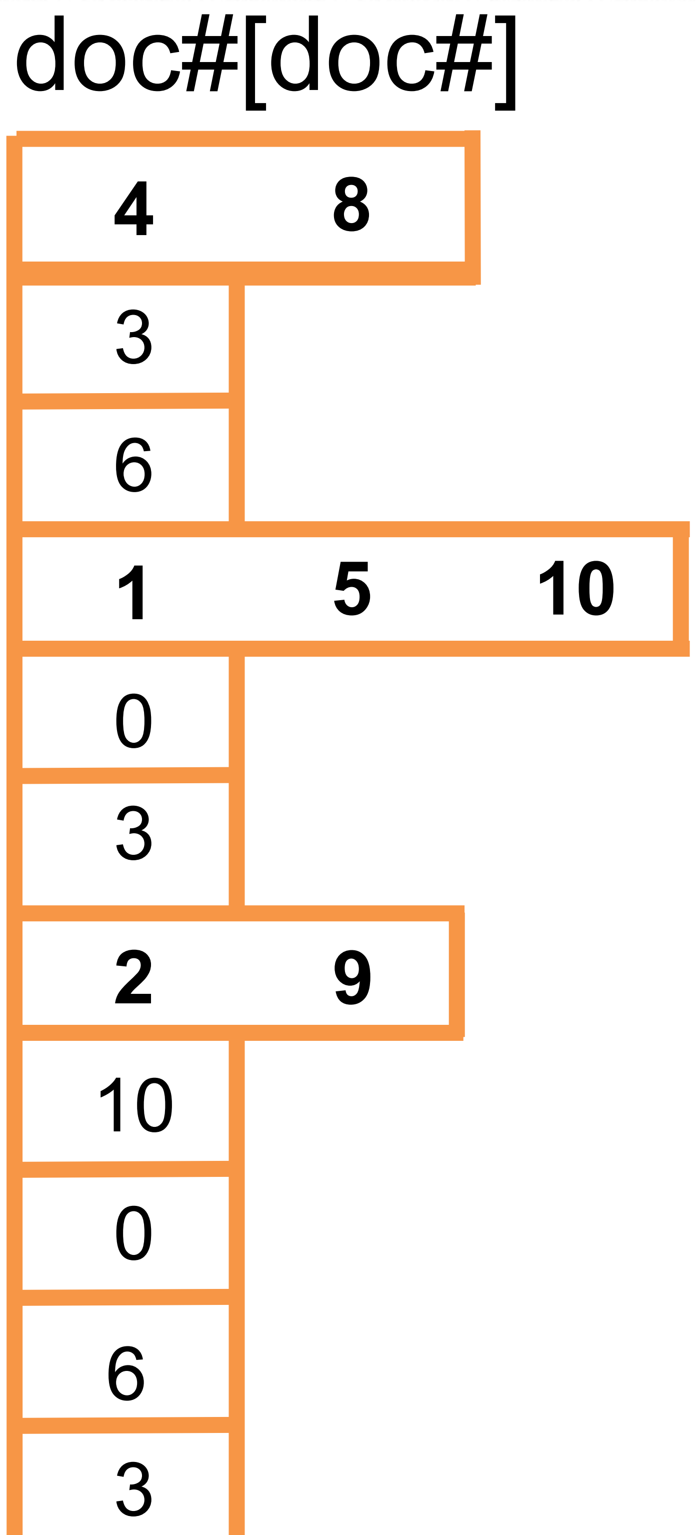


https://github.com/m-khl/lucene-solr/tree/dvjoin-benchmark-5-1

•

	JoinUtil	JoinIndex	Global Ordinals		Block Join
searching	slow	**fast**	**fast**	<	faster anyway
reindexing	fast	**uber slow**	**fast**		slow

Indexing is still a problem



Further Plans 2.0

- incremental join-index update
- ~~perhaps just calculate and cache it~~
- ~~or put to dedicated index~~
- join in both directions
- ~~calculate optimal execution plan of segments enumeration~~
- edge case for benchmark

Summary

LUCENE/SOLR**REVOLUTION**

- Joins in General
- JoinUtil vs Block-join vs GlobalOrdinals
- updatable DocValues
- opportunities for improving query-time joins:
  - ~~eliminate term enum~~
  - ~~choose lower cardinality side for enumeration~~
  - GlobalOrdinalsJoin

Lucidworks

References

LUCENE/SOLR**REVOLUTION**

- Searching relational content with Lucene's BlockJoinQuery
  http://blog.mikemccandless.com/
- Solr Experience: search parent-child relations. Part I
  Solr block-join support
  http://blog.griddynamics.com/

- https://wiki.apache.org/solr/Join
- http://www.slideshare.net/martijnvg/document-relations
- SOLR-6234 - {!scorejoin }
- LUCENE-6352

- Updatable DocValues Under the Hood
  http://shaierera.blogspot.com/

- Subject: How to openIfChanged the most recent merge?

  at: java-dev@lucene.apache.org

Lucidworks

Thanks for Joining us!

LUCENE/SOLR**REVOLUTION**

https://goo.gl/hjsYZW

Lucidworks

Off scope

Joins' Zoo in Lucene

True Joins

- **query-time** join
  - JoinUtil
  - ~~{!join }~~
  - {!scorejoin } - SOLR-6234
- **index-time** join aka block-join {!parent}

Joins' Zoo in Lucene

LUCENE/SOLR**REVOLUTION**

True Joins

- **query-time** join
  - JoinUtil
  - ~~{!join }~~,
  - {!scorejoin } - SOLR-6234

- **index-time** join aka block-join {!parent}

Workarounds

- term positions/SpanQueries
- FieldCollapsing/Grouping
- term decoration
  - spatial

- multivalue fields

Lucidworks

Joins' Zoo in Lucene

True Joins

- **query-time** join
  - JoinUtil
  - ~~{!join }~~,
  - {!scorejoin } - SOLR-6234
- **index-time** join aka block-join {!parent}

Workarounds

- term positions/SpanQueries
- FieldCollapsing/Grouping
- term decoration
  - spatial
- $\cancel{\text{multivalue fields}}$

Two phase update problem

Subject: How to openIfChanged the most recent merge?

at: java-dev@lucene.apache.org

LUCENE/SOLR**REVOLUTION**

JoinUtil

- query-time
- indexing is fast
- searching is slow, why?
  - expensive term enum
  - single enumeration order

BlockJoin

- index-time
- reindexing whole block is as expensive as mandatory
- searching is darn fast, **however**
  - can’t reorder child docs

Lucidworks

store ref = segment#, doc#
put ref to previous and current segment in DV
when add new segment, join IDs with previous segments
- for parent, just ref to all children docnums
- for children, add plain field refsToSeg:seg#
-

when score parents on some segment
- buffer them with the link refs, then
- intersect buffered link refs with children query on previous segments
- search all segments for refsToSeg:seg#, intersect with children query, obtain perent ref from DV intersect with buffered