RNT GROUP

50 оттенков Join в Apache Lucene

Михаил Хлуднев, RNT Group

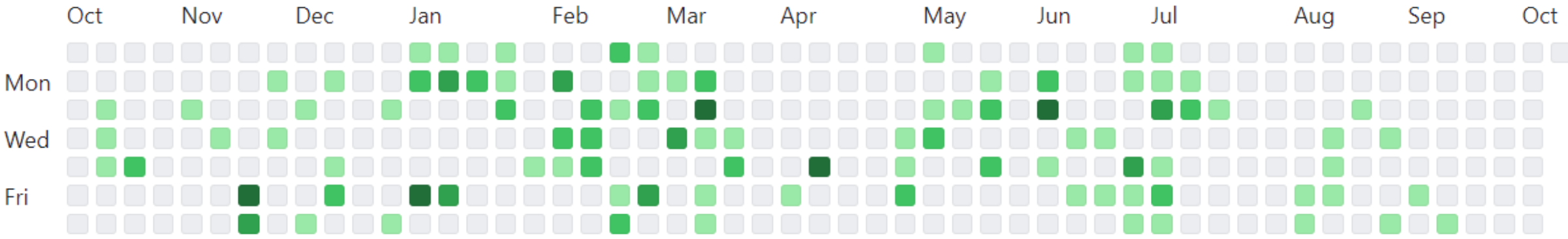
https://t.me/MUST_SEARCH

10.2023

- Apache Lucene,Solr Committer since 2015
- RNTGroup
- https://t.me/MUST_SEARCH

153 contributions in the last year

Contribution settings



Learn how we count contributions

Less More

План

Продуктовый каталог

Почему не b-tree

Обратный Индекс

Распределённый поиск

Join локальный

- компромиссы

Распределённый Join

Аспекты функциональности

Длинные строковые значения индексного поля

Многозначные поля, особенно текст

Произвольные комбинации фильтров

Сортировки по произвольным полям

Аспекты функциональности	B-tree БД
Длинные строковые значения индексного поля	
Многозначные поля, особенно текст	
Произвольные комбинации фильтров	
Сортировки по произвольным полям	

osgreSQL есть array!

индексируется GIN Generalized Inverted Index

PosgreSQL комбинирует дексы

материализуя битовые карты и комбинируя их

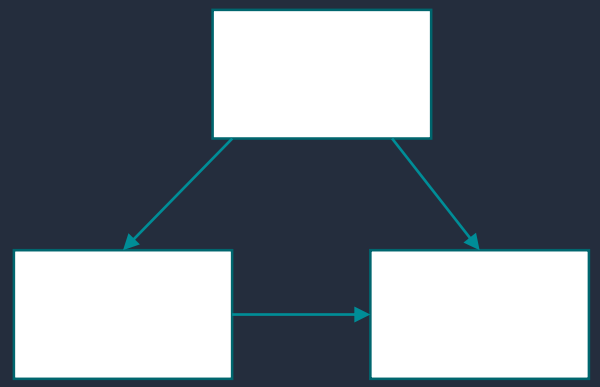
Аспекты функциональности	B-tree БД	Обратный индекс
Длинные строковые значения индексного поля	уве	сжатие индекса
Многозначные поля, особенно текст	не соответствуют нормальной форме	документная модель без нормализации
Произвольные комбинации фильтров	композитные индексы	потоковый алгоритм комбинации индексов
Сортировки по произвольным полям	включающие индексы	колонки данных

Медленно!

```
WHERE BRAND='Nike'
ORDER BY price ASC LIMIT 10
```

RNT GROUP

...WHERE BRAND='Nike' ORDER BY PRICE LIMIT 10



B-tree

ID	NAME	BRAND	PRICE
23	Classic T-Shirt	Nike	12
454	Bluetooth Earphones	Sony	32
232	Stainless Steel Water Bottle	Hydro Flask	6
5232	Portable Bluetooth Speaker	JBL	27
15	Ceramic Coffee Mug	Starbucks	16

In an index scan, the index access method is responsible for regurgitating the TIDs of all the tuples it has been told about that match the *scan keys*.

RNT GROUP

5. Колонки данных

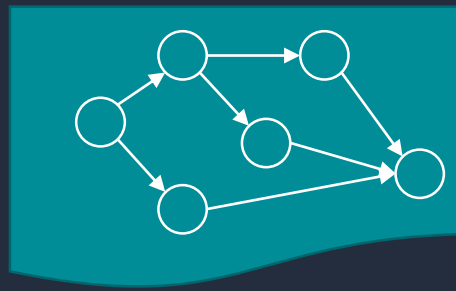
[docNum]

offset
125
267
456
672
702
756

ID	NAME	BRAND	PRICE
23	Classic T-Shirt	Nike	12
454	Bluetooth Earphones	Sony	32
232	Stainless Steel Water Bottle	Hydro Flask	6
5232	Portable Bluetooth Speaker	JBL	27
15	Ceramic Coffee Mug	Starbucks	16

[docNum]

PRICE
12
32
6
27
16



-er, -ed, -le

-ly,-ed,-ny

docNums[]

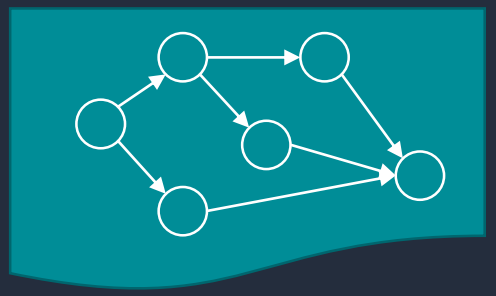
5,42,76,82,103,4005

...

Обновления обратного индекса

aka log-structured

4. Обновление обратного индекса



-er, -ed, -le

-ly,-ed,-ny

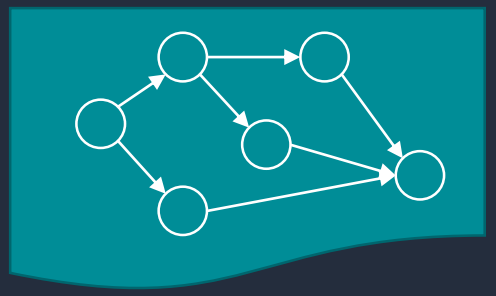
5,42,76…

ID	NAME

N
N
N

N
Y
N
Y
N
N
N
N

N
Y
N
Y
N
Y
Y
N
N
N
N
N
N
N

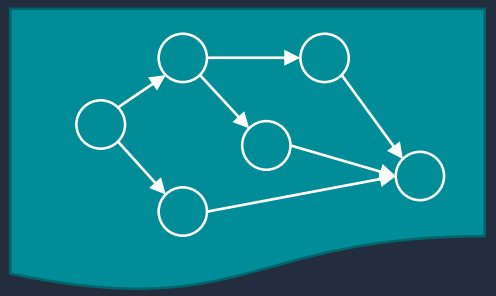


-er, -ed, -le

-ly,-ed,-ny

5,42,76…

ID	NAME



-er, -ed, -le

-ly,-ed,-ny

5,42,76…

ID	NAME

Преимущества сегментации

- Файлы записываются только один раз добавлением без обновления
- Простая синхронизация
- Сильное сжатие

Недостатки

- Нет транзакций на уровне записи
- Дублирование словаря термов в каждом сегменте
- Необходимость слияния
- Обновления тяжёлые

Аспекты Функциональности		Обратный Индекс
Длинные строковые значения индексного поля		сжатие индекса
Многозначные поля, особенно текст		документная модель без нормализации
Произвольные комбинации фильтров		потоковый алгоритм комбинации индексов
Сортировки по произвольным полям		колонки данных

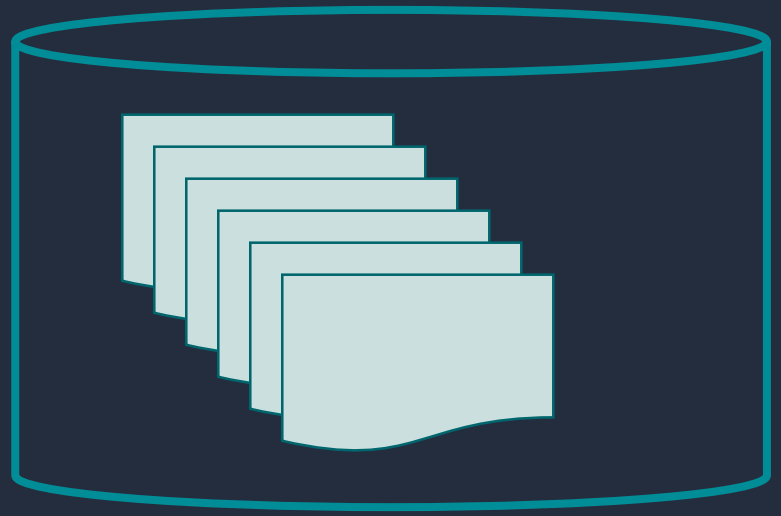
Распределённый Поиск

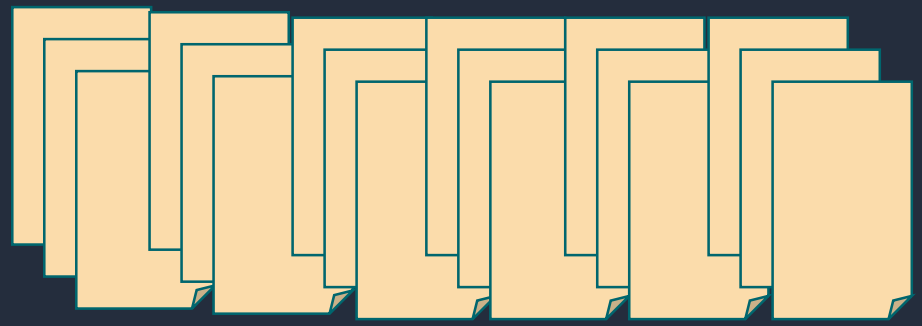
4. Обновление обратного индекса

ID	NAME

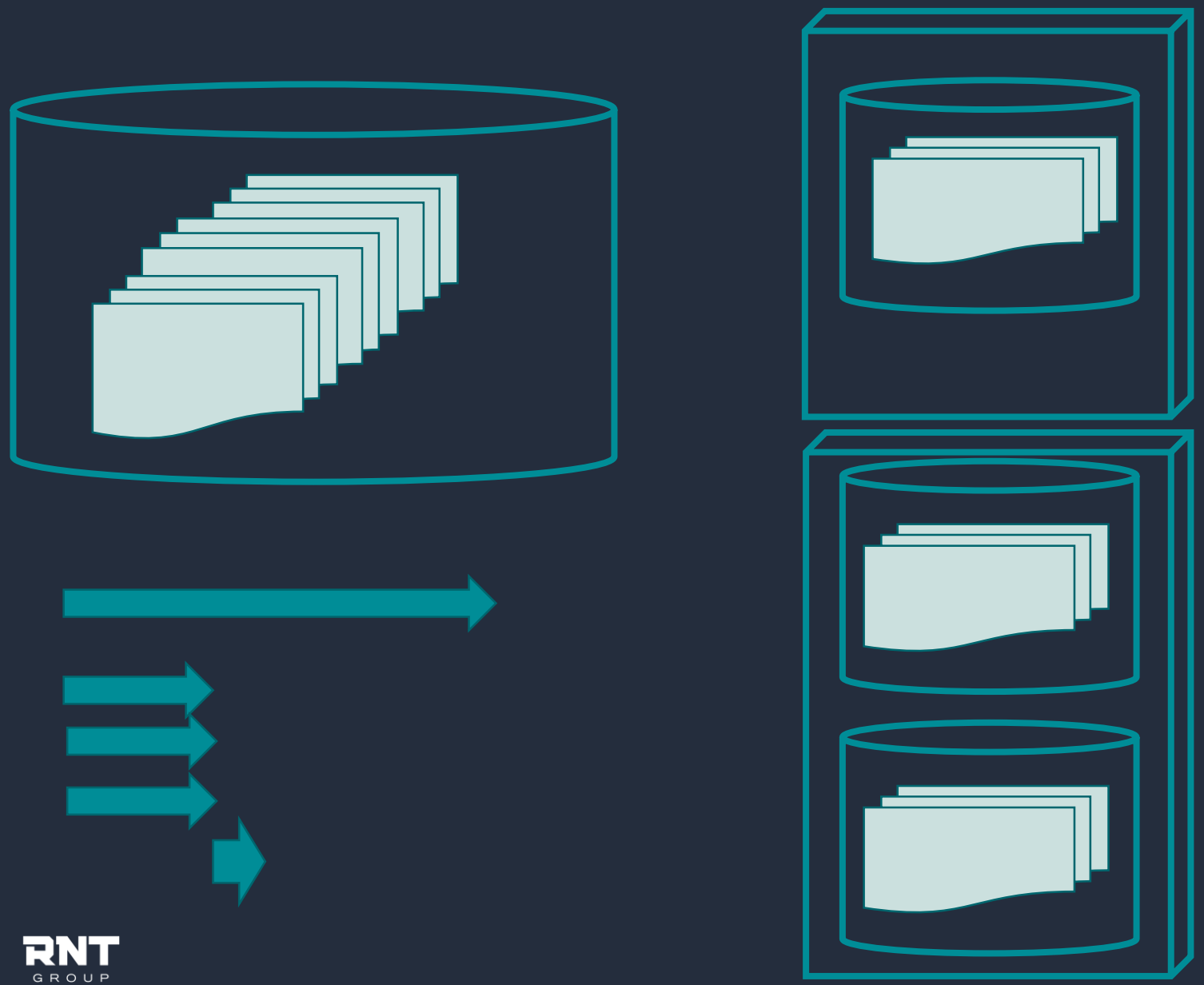
ID	NAME

ID	NAME

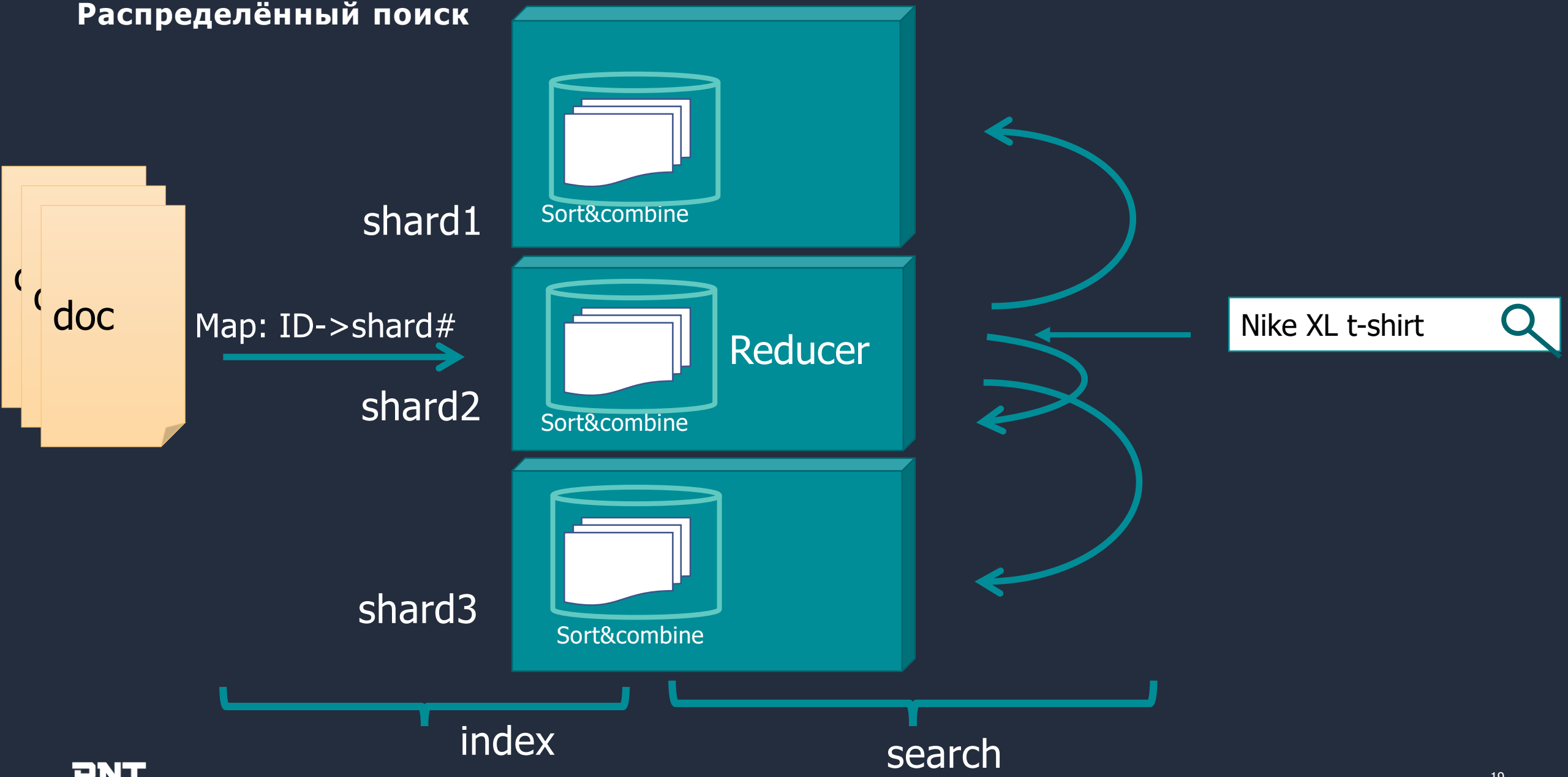




Поиск по цепочке сегментов



Распределённый поиск



REST
Кластер (распределённый поиск)
Обновление отдельных документов

Документные Базы Данных

```
{
  "contact": {
    "firstname": "Bob",
    "lastname": "Smith",
    "phone": [
      {
        "type": "Cell",
        "value": "(123) 555-0178"
      },
      {
        "type": "Work",
        "value": "(890) 555-0133"
      }
    ],
    "address": {
      "type": "Home",
      "street1": "123 Back St.",
      "city": "Boys",
      "state": "AR",
      "zip": "32225",
      "country": "US"
    }
  }
}
```

Lucene

- Многозначные поля – ☑
- Дочерние сущности - ?

- Единственный индекс
- Результат - docSet
- Нет кортежей*

- Документы разного вида в одном индексе

Join для продуктового каталога

Name: Premium Running Shoes
Brand: SpeedRunner
Description: High-performance running shoes designed for optimal comfort and speed.
Category: Footwear
Price: $99.99
SKUs:

SKU: SR001
- Size: US 8 / EU 41
- Color: Black/White
- Stock: 20 units

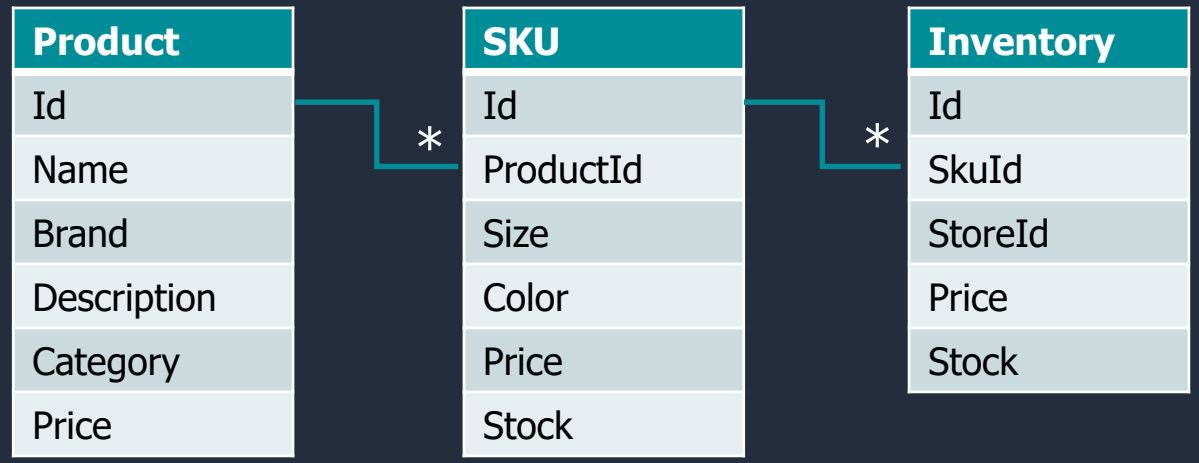
SKU: SR002
- Size: US 9 / EU 42
- Color: Blue/Orange
- Stock: 15 units

SKU: SR003
- Size: US 10 / EU 43
- Color: Gray/Yellow
- Stock: 12 units

Store 67
- Stock: 13 units
- Price: $89.9

Store 2
- Stock: 4 units
- Price: $109.3

Store 532
- Stock: 60 units
- Price: $95.3



Тип

- [ ] Футболка
- [ ] Комплект футболок

Международный размер

- [ ] XXS
- [ ] XS
- [ ] S
- [ ] M
- [ ] L

Посмотреть все

Цвет

Найти

- [ ] Бежевый
- [ ] Белый
- [ ] Бирюзовый
- [ ] Бордовый
- [ ] Голубой
- [ ] Желтый
- [ ] Зеленый
- [ ] Коричневый
- [ ] Красный
- [ ] Оранжевый
- [ ] Розовый
- [ ] Светло-бежевый
- [ ] Светло-зеленый
- [ ] Светло-коричневый
- [ ] Светло-розовый

Свернуть

Бренды

- [x] TBOE
- [ ] BEST TRICOTAGE
- [ ] HUGO

918 ₽ 1299 ₽ −29%
Стало дешевле
Футболка TBOE

743 ₽ 999 ₽ −26%
Стало дешевле
Футболка TBOE

918 ₽ 1299 ₽ −29%
Стало дешевле
Футболка TBOE

743 ₽ 999 ₽ −26%
Стало дешевле
Футболка TBOE

668 ₽ 899 ₽ −26%
Стало дешевле
Футболка TBOE
4.7 42

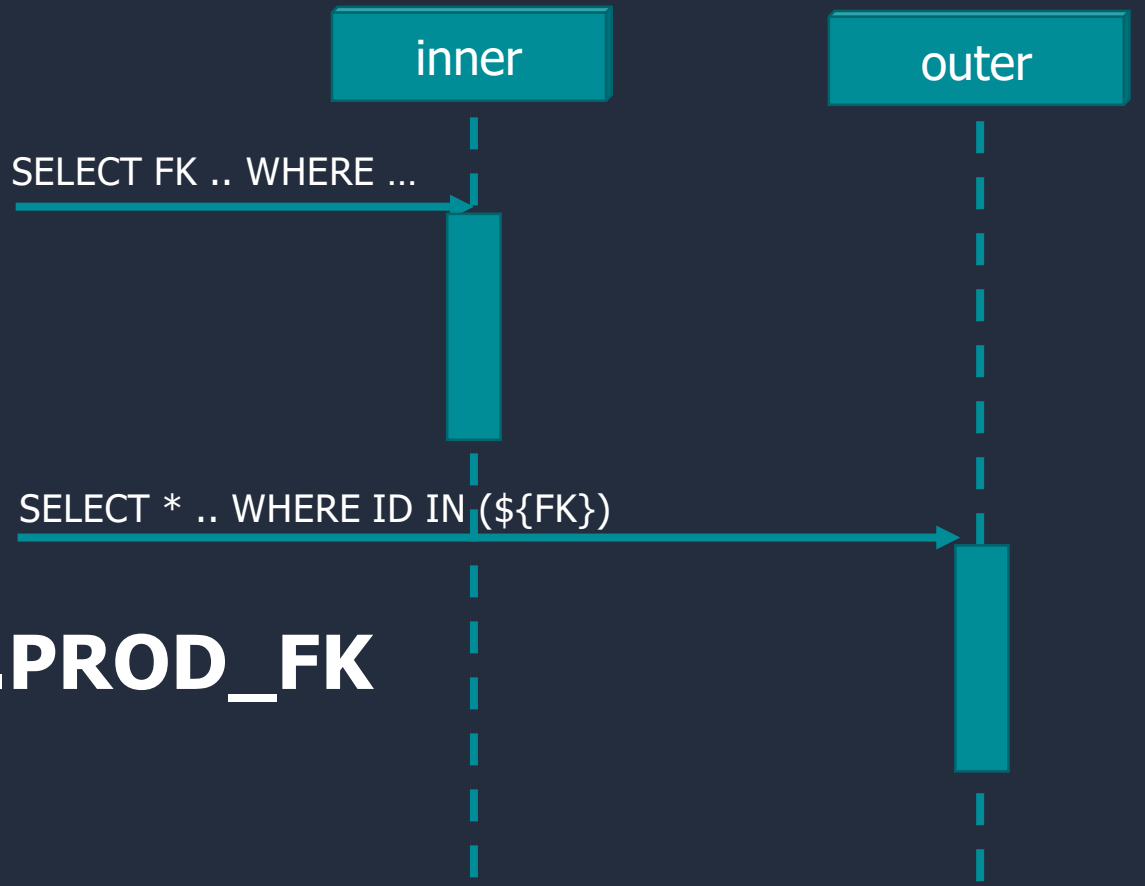
743 ₽ 999 ₽ −26%
Стало дешевле
Футболка TBOE
4.8 200

918 ₽ 1299 ₽ −29%
Стало дешевле
Футболка TBOE

918 ₽ 1299 ₽ −29%
Стало дешевле
Футболка TBOE

RNT GROUP

Equi-, Semi- Join




```
SELECT *   FROM products
           JOIN sku
           ON products.ID=sku.PROD_FK
           WHERE sku.size="XL"
```

```
SELECT * FROM products WHERE ID IN (
      SELECT PROD_FK FROM sku WHERE size="XL")
```

Компромиссы

Product

SKU1
Color:
Red
Size: XL

SKU2
Color:
Blue
Size: M

flattened into
multi-value fields

Product
[SKU1, SKU2]

Color: [Red, Blue]
Size: [XL, M]

Elasticsearch object
field type

Денормализация

SKU1+Product

SKU2+Product

Elasticsearch Result Collapsing

RNT GROUP

Solr Join

```
{"join": {      "query":"SIZE:XL",
                "from":"product_fk",
                "to":"product_id}}
```

```
SELECT * FROM products_skus WHERE product_id
        IN (SELECT product_fk  FROM products_skus WHERE SIZE="XL")
```

Product_fk	docNums
3	…
6	0,1,14,53,678…
9	…

SIZE="XL"

Product_id	docNums
4	…
6	45
8	…

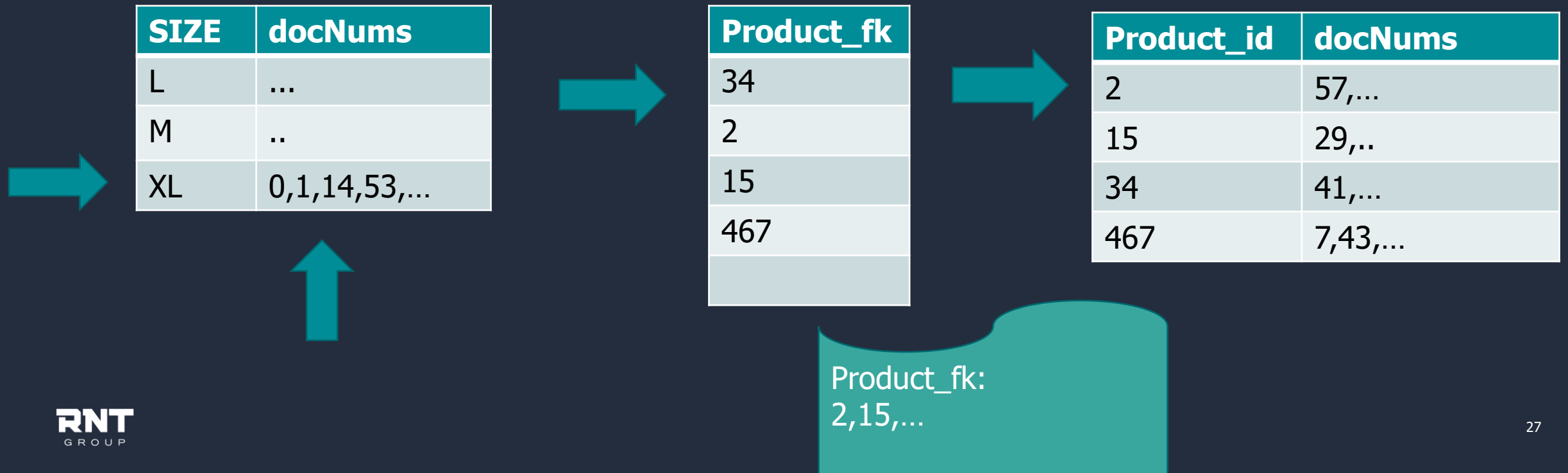
RNT GROUP

Lucene Join (Elasticsearch&Solr)

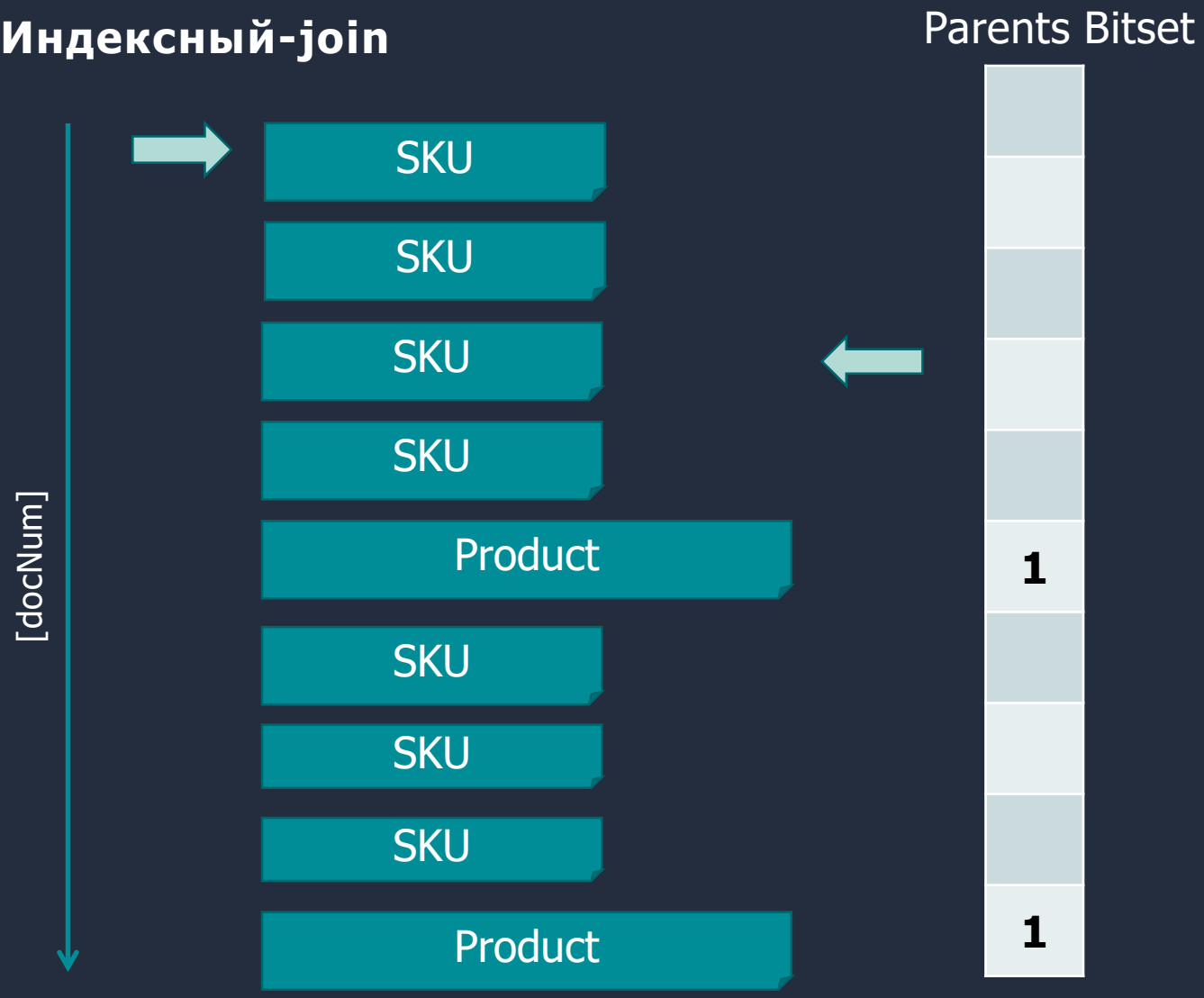
```
SELECT * FROM products_skus WHERE product_id
        IN (SELECT product_fk  FROM products_skus WHERE SIZE="XL")
```

```
JoinUtil.createJoinQuery(fromField, true, toField, new TermQuery(new Term("SIZE", "XL")),
                                                                fromSearcher, scoreMode)
```

```
{ "has_child": { "type": "child", "query": {"match": {"SIZE":"XL"} },
```



Индексный-join



Elasticsearch Nested Type/Query

Подходы к Join

Соединение во-время запроса

- Быстрые обновления

Индексные соединения

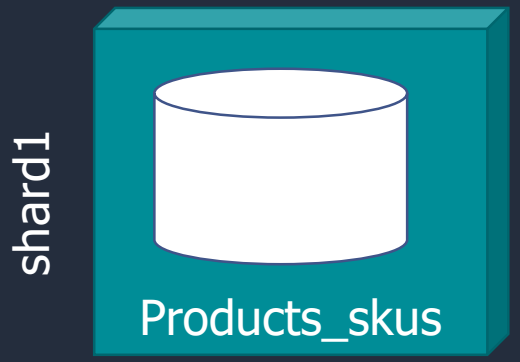
- Усложняют обновление

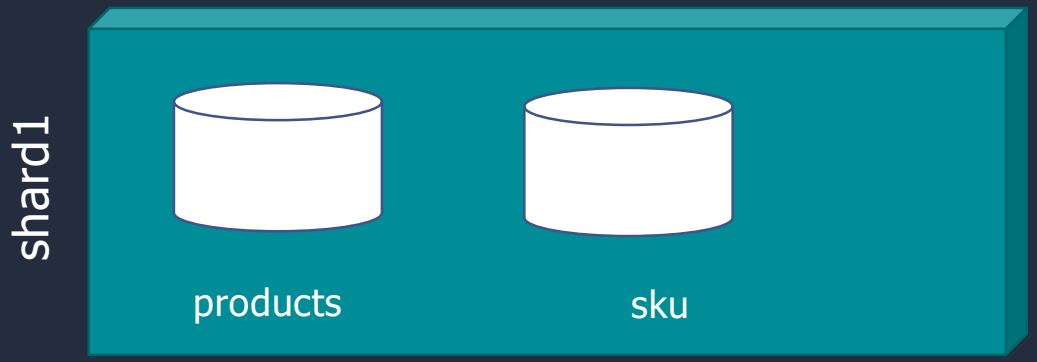
Денормализация, конкатенация

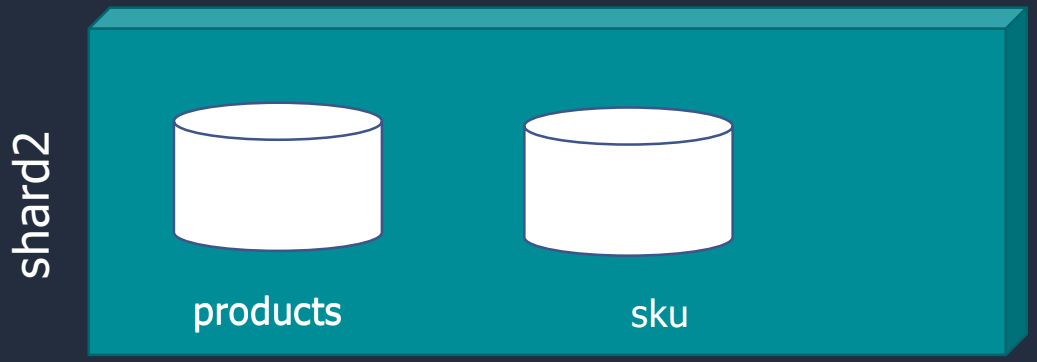
- Усложнение индексации и поиска

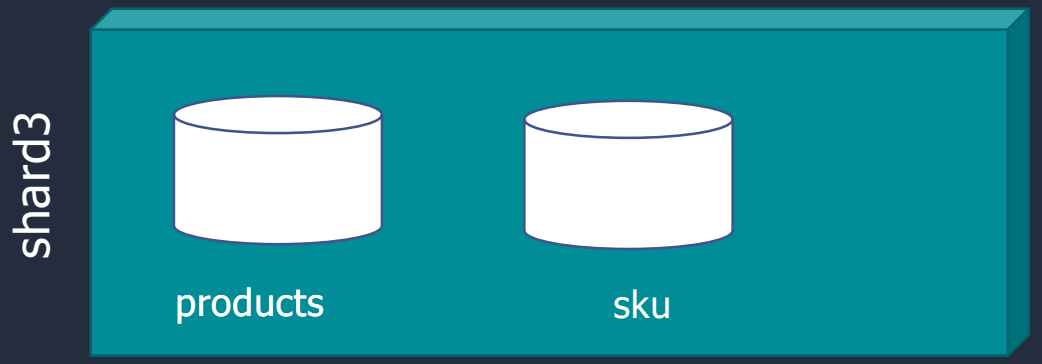
Для сложных иерархий нужна комбинация нескольких методов

Распределённый Join



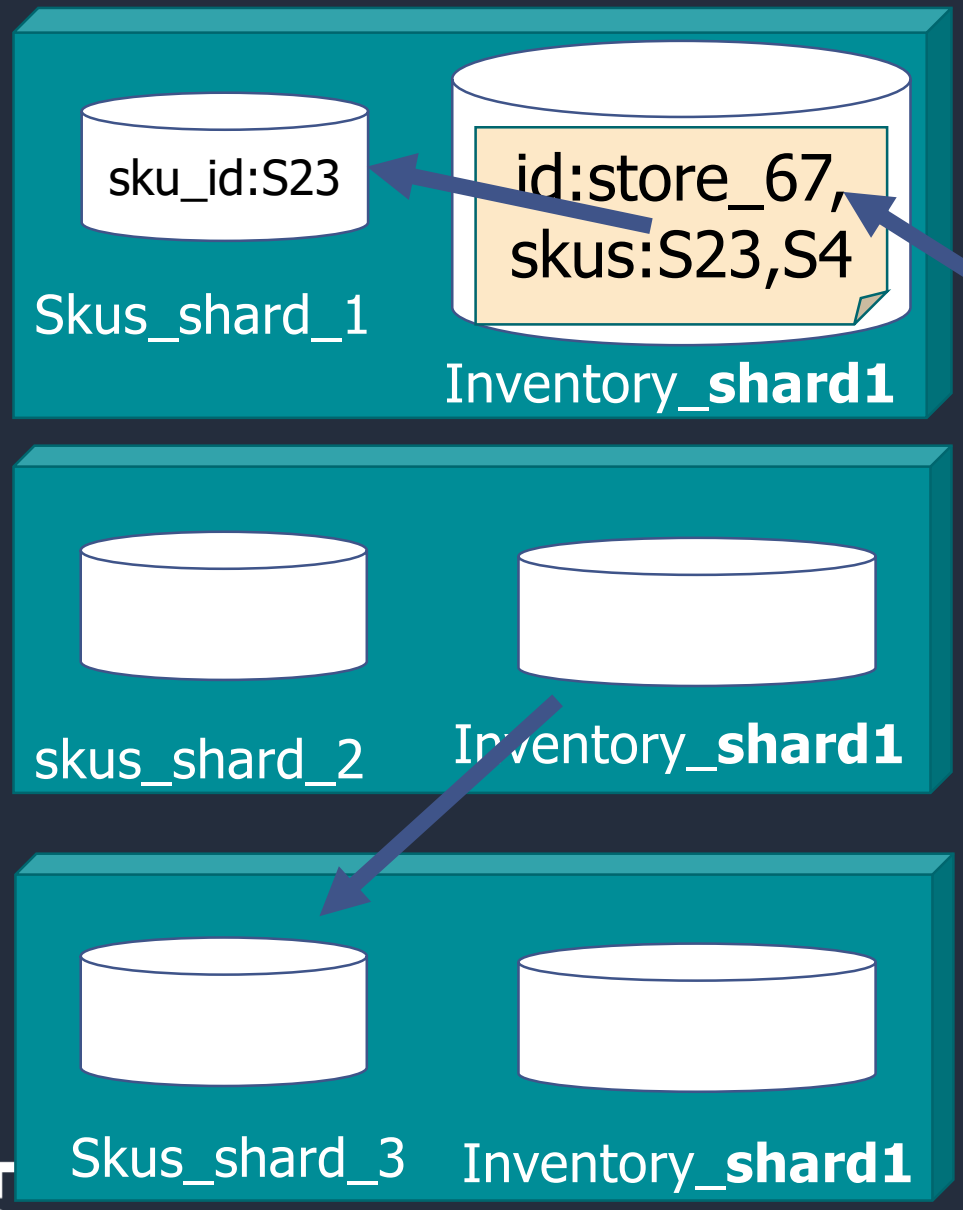






RNT GROUP

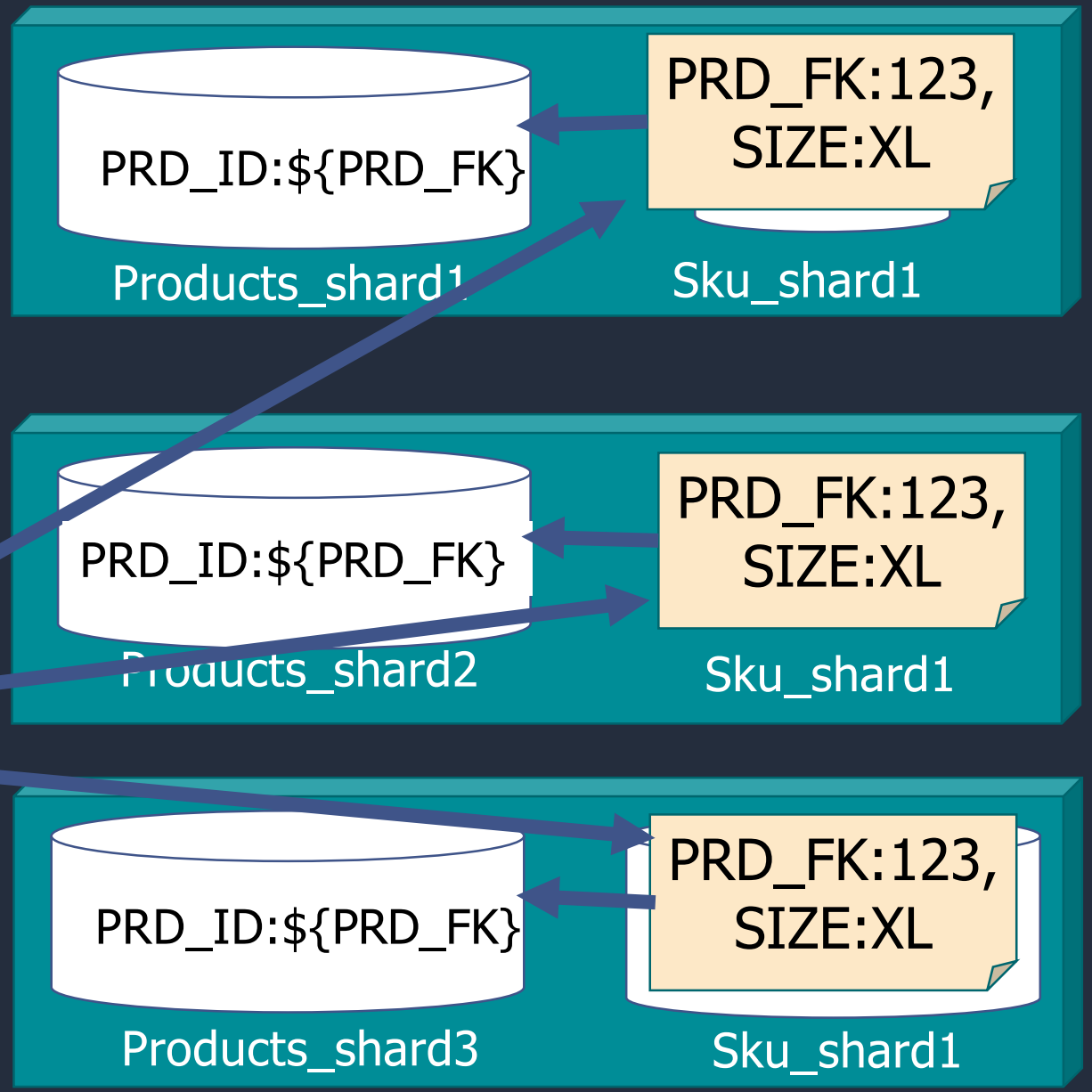
Elasticsearch/OpenSearch termsLookup



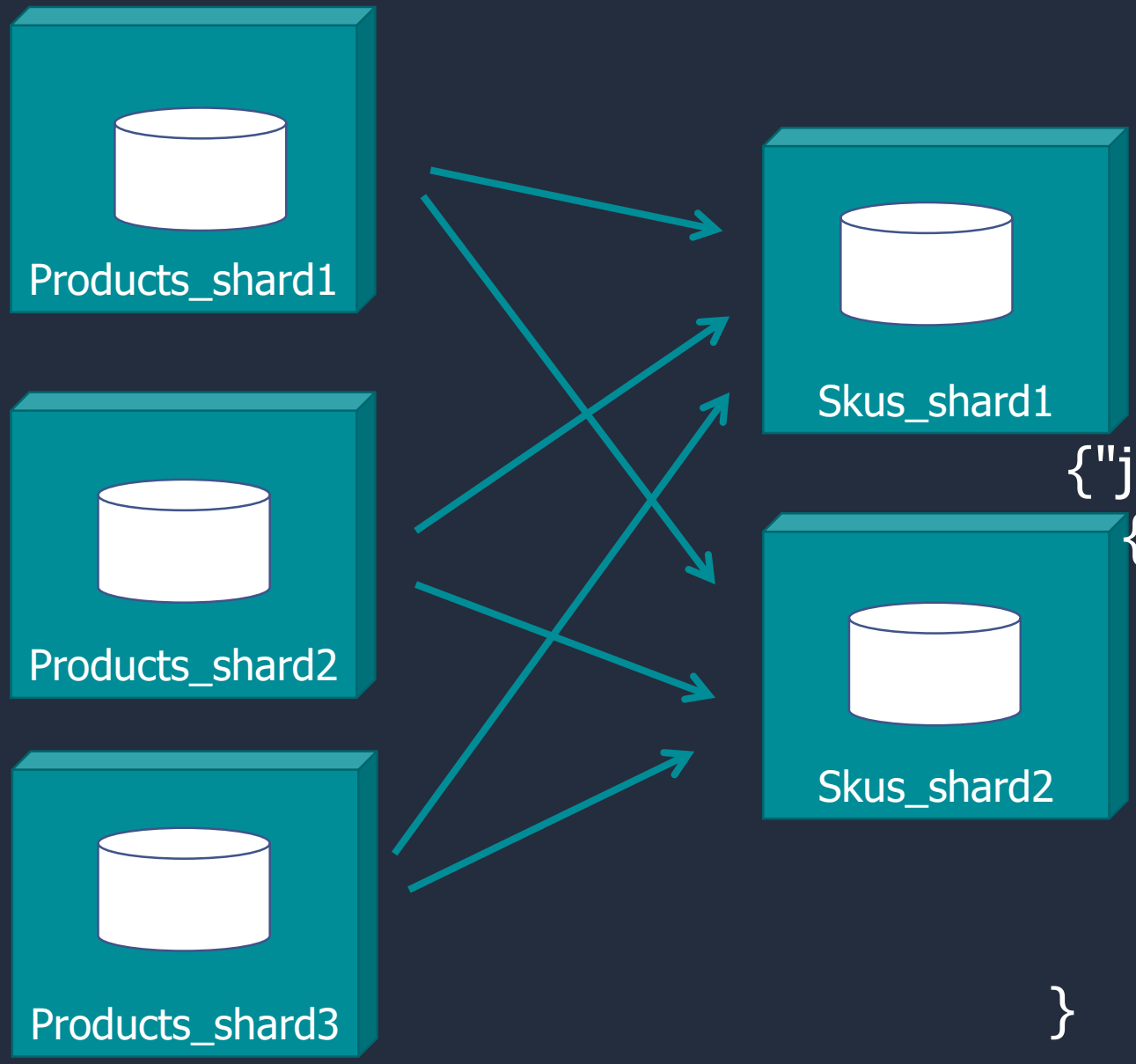

```
GET skus/_search
{
  "query": {
    "terms": {
        "sku_id" : {
            "index" : "inventory",
            "id" : "store_67",
            "path" : "skus"
        }
    }
  }
}
```

Apache Solr pre 9.3

```
GET /solr/products/select
{
  "query":
      {"join":
          {
                  "fromIndex": "sku",
                  "from":"PRD_FK",
                  "to":"PRD_ID",
                  "query":"SIZE:XL"}}
}
```



Apache Solr – crossCollection join




```
{"join":
    {
        "fromIndex": "sku",
        "from":"product_fk",
        "to":"product_id",
        "query":"SIZE:XL"
        "method":"crossCollection"
    }
}
```

RNT GROUP

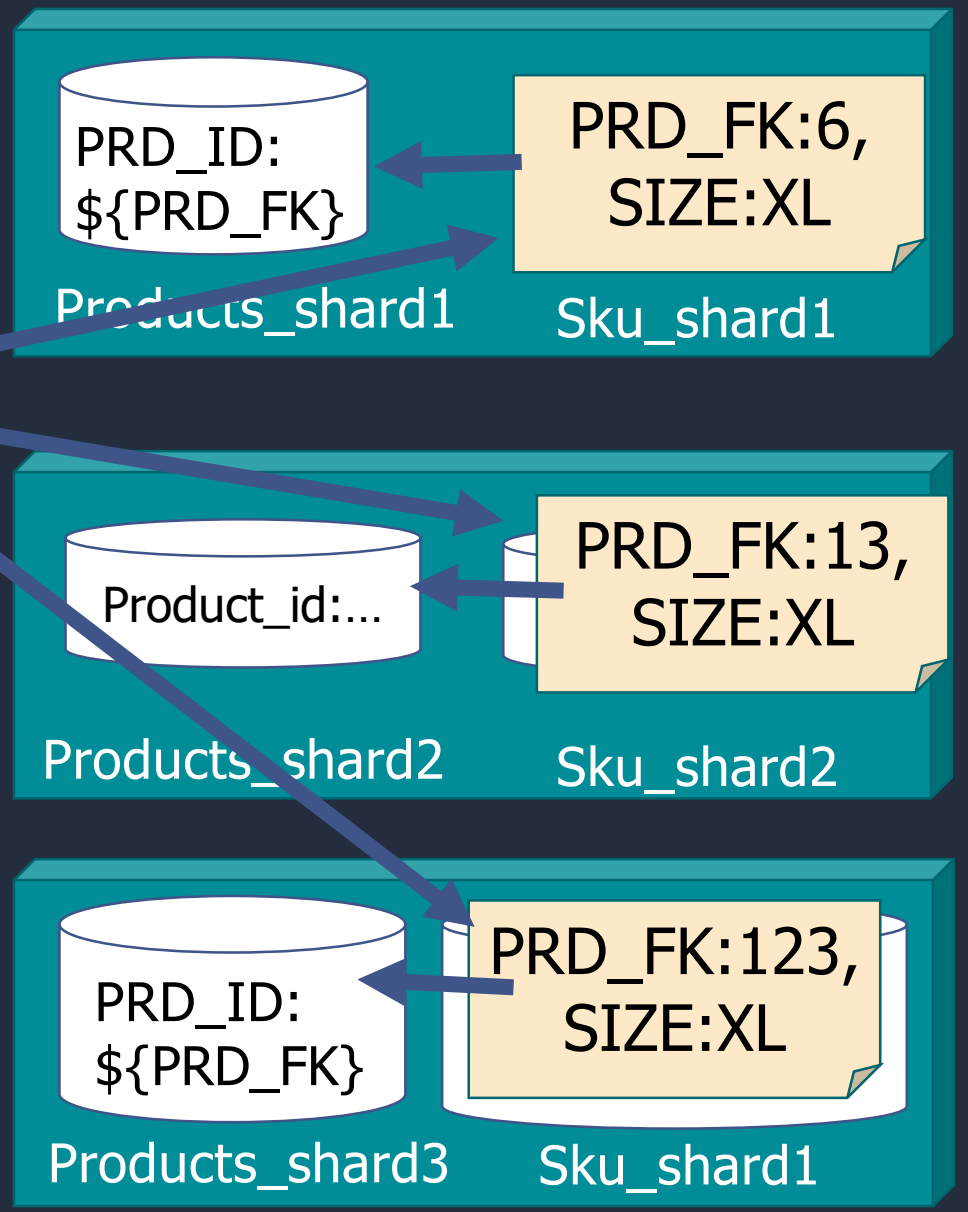
Babu, S. (2012). Massively Parallel Databases and MapReduce Systems

2.3.1 Parallel Query Execution

Consider a query that joins two tables R and S based on the equi-join condition $R.a = S.a$. In §2.2.2, we introduced a *collocated join* operator that can perform the join if tables R and S are both partitioned and the partitions are assigned such that any pair of joining partitions is stored on the same node. A collocated join operator is often the most efficient way to perform the join because it performs the join in parallel while avoiding the need to transfer data between nodes. However, a collocated join is only possible if the partitioning and assignment of the joining tables is planned in advance.

Apache Solr 9.3 и выше

```
GET /solr/products/select
{ "query": {"join":   {
        "fromIndex": "sku", from":"product_fk",
        "to":"product_id",   "query":"SIZE:XL"}}
}
...
{
    "add":{
        "name": ".placement-plugin",
        "class": "...AffinityPlacementFactory",
        "config": {
          "withCollectionShards": {
            "products": "sku"
          }
        }
    }}'
```



Алгоритм	Ограничение на внутренний индекс	Передача FK между узлами
ES/OS Terms Lookup *	нет	при необходимости
Solr pre 9.3	нешардирован полно-реплицирован	нет
Solr crossCollection, ES SIREN	нет	да
Solr since 9.3	совместимо шардирован и коллоцирован	нет

Рассмотрели

- Задачу продуктового каталога
- Обратный индекс
- Распределённый поиск
- Алгоритмы соединения
    - во время индексации
    - во время запроса
- Коллоцированный распределённый join

А давайте!

- напишем коллоцированый join плагин для OpenSearch

Литература

- B-Tree индекс и его производные в PostgreSQL. Азат Якупов. https://habr.com/ru/companies/quadcode/articles/696498/
- Designing Data-Intensive Applications. Martin Kleppmann
- An Efficient Implementation of Trie Structures JUN-ICHI AOE AND KATSUSHI MORIMOTO 1992
- Performance of compressed inverted list caching in search engines. J.Zhang, X.Long, T.Suel. WWW'08:
- Massively Parallel Databases and MapReduce Systems. Foundations and Trends® in Databases. Babu, S. (2012).
- The Design and Implementation of Modern Column-Oriented Database Systems, Foundations and Trends® ..D. Abadi,.. (2013)
- Information Retrieval: The Early Years. Foundations and Trends® in Information Retrieval. Donna Harman (2019)

Ссылки

https://jokerconf.com/talks/4ae982202c3d435494538e3017866a8c/ **Есть что новое в поиске? А если найду? Jokerconf'22**

https://www.elastic.co/blog/frame-of-reference-and-roaring-bitmaps

https://www.elastic.co/blog/this-week-in-elasticsearch-and-apache-lucene-2019-10-28

https://blog.mikemccandless.com/2014/05/choosing-fast-unique-identifier-uuid.html

https://blog.mikemccandless.com/2019/10/concurrent-query-execution-in-apache.html

https://www.elastic.co/guide/en/elasticsearch/reference/current/consistent-scoring.html

https://www.elastic.co/guide/en/elasticsearch/reference/8.9/search-aggregations-bucket-terms-aggregation.html

https://www.youtube.com/watch?v=EkkzSLstSAE Berlin Buzzwords 2019: M. Sokolov & M. McCandless– E-Commerce search at scale

https://www.youtube.com/watch?v=vyMkwmAoplI&t=1177s Cross Collection Join Query

https://docs.siren.io/siren-federate-user-guide/32/siren-federate/introduction.html

https://github.com/apache/lucene/pull/12434 **Add ParentJoin KNN support**

https://solr.apache.org/guide/solr/latest/query-guide/join-query-parser.html#joining-multiple-shard-collections Joining Multiple Shard Collections

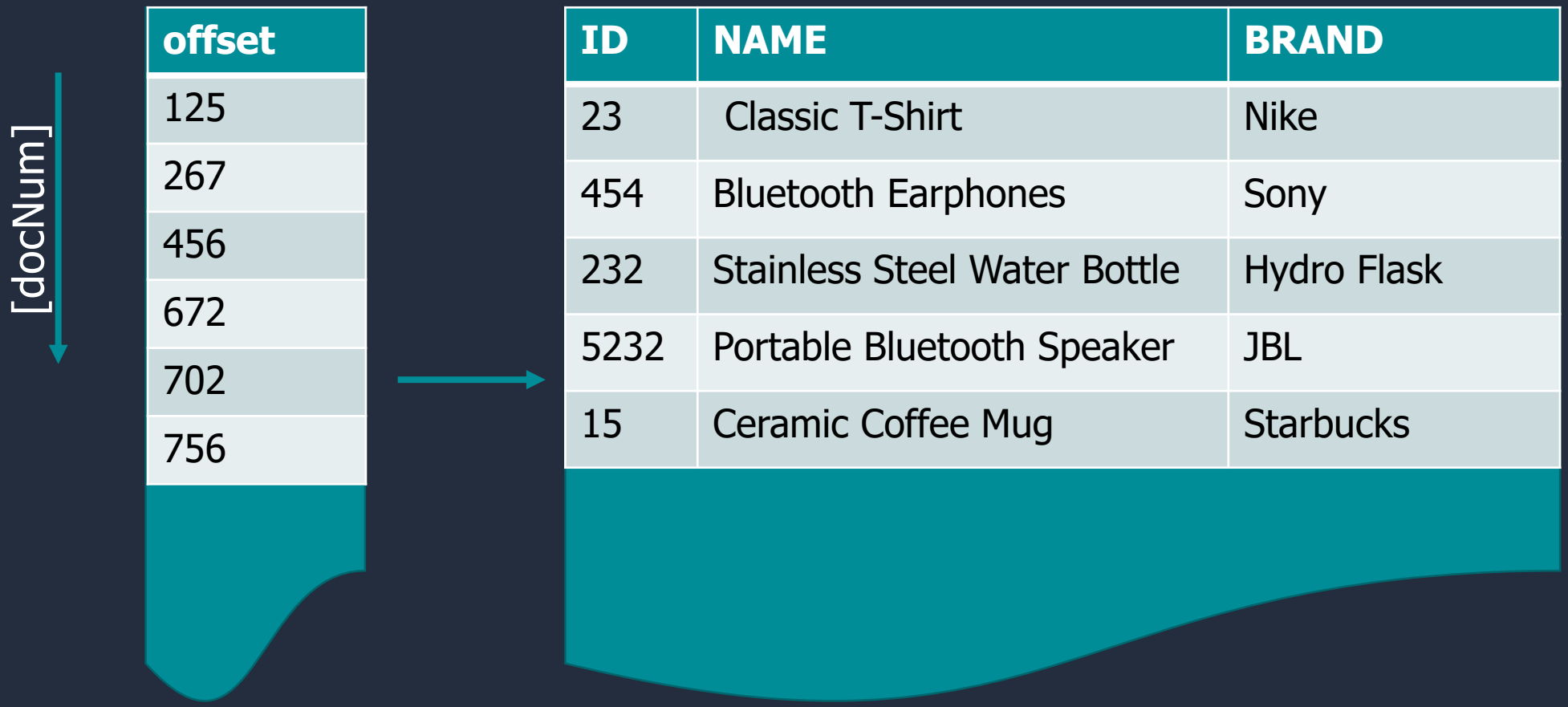
ПРИЛОЖЕНИЯ

- Комбинация фильтров в обратном индексе
- Декомпрессия с помощью векторных инструкций (SIMD) в Java
- Что не так с распределённым поиском

Обратный Индекс

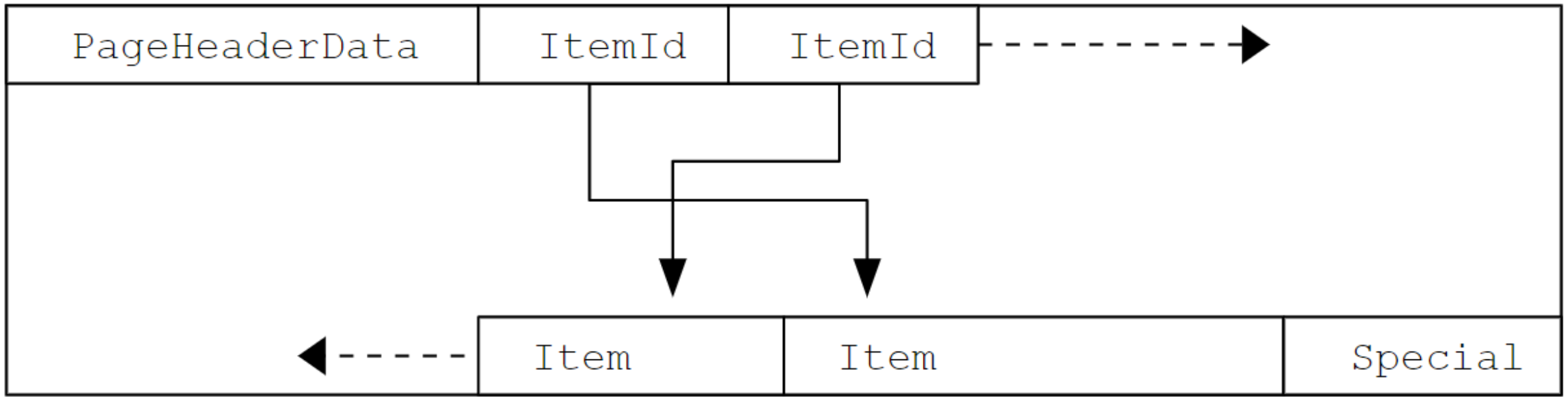
Детали по прямому индексу, словарю термов, комбинации фильтров

1. Прямой индекс



Тем временем в PostgresSQL

Figure 73.1. Page Layout

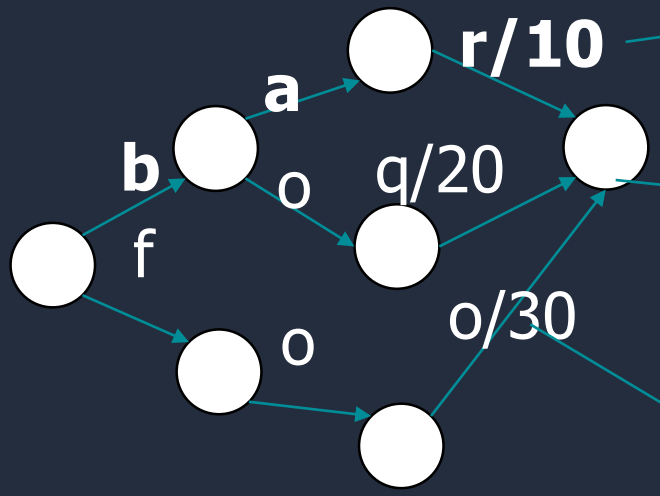


2. Словарь термов

barber
barely
barney
barrel
barter

boquerones
boquet
boquetin

food
fool
foolish
foosball
football
footprint



Block prefix FST

-ber,-ely,-ney,-rel,-ter

boq- term block

foo- term block

Имеется и более подходящая структура данных, которая позволяет объединить скорость доступа к массиву с экономным расходом памяти списками с состояниями по умолчанию. Эту структуру можно рассматривать как четыре массива, как показано на рис. 3.66.[8] Массив *base* используется для определения базового расположения записей для состояния s, хранящихся в массивах *next* и *check*. Массив *default* применяется для определения альтернативной базовой позиции, если массив *check* говорит нам, что данное значение *base* $[s]$ некорректно.

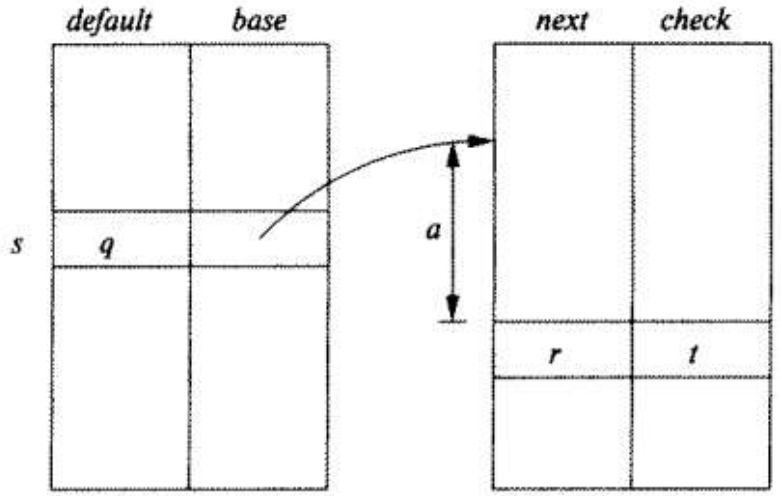


Рис. 3.66. Структура данных для представления таблиц переходов

Для вычисления перехода из состояния s для входного символа a *nextState* (s, a) мы проверяем значения записей *next* и *check* в позиции $l = base\,[s] + a$, где символ a рассматривается как целое число, вероятно, из диапазона от 0 до 127. Если $check\,[l] = s$, то эта запись корректна, и следующим для состояния s и входного символа a является состояние $next\,[l]$. Если же $check\,[l] \neq s$, то мы находим другое состояние $t = default\,[s]$ и повторяем процесс так, как если бы t было текущим состоянием. Более формально функция *nextState* определяется следующим

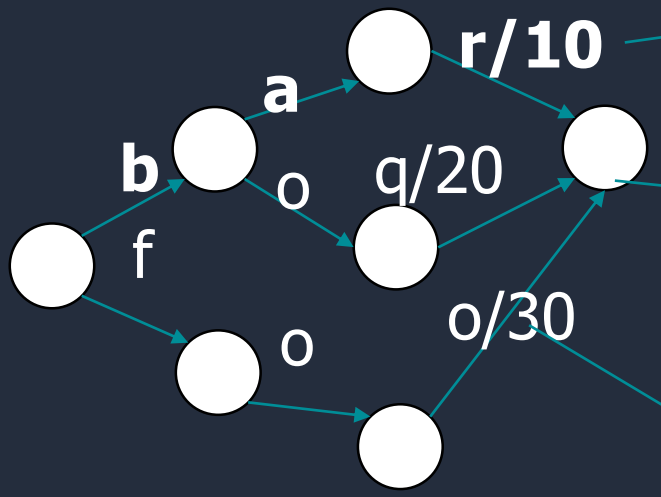
In the double-array stru
with a number more than
sition $g(t, a) = t'$ from one

$$t' = base[t] + a,$$
$$check[t'] = t.$$

1 and its state transition

Поиск barrel

barrel



Block prefix FST

-ber,-ely,-ney,-rel,-ter..
100,110,120,130,140

boq- term block

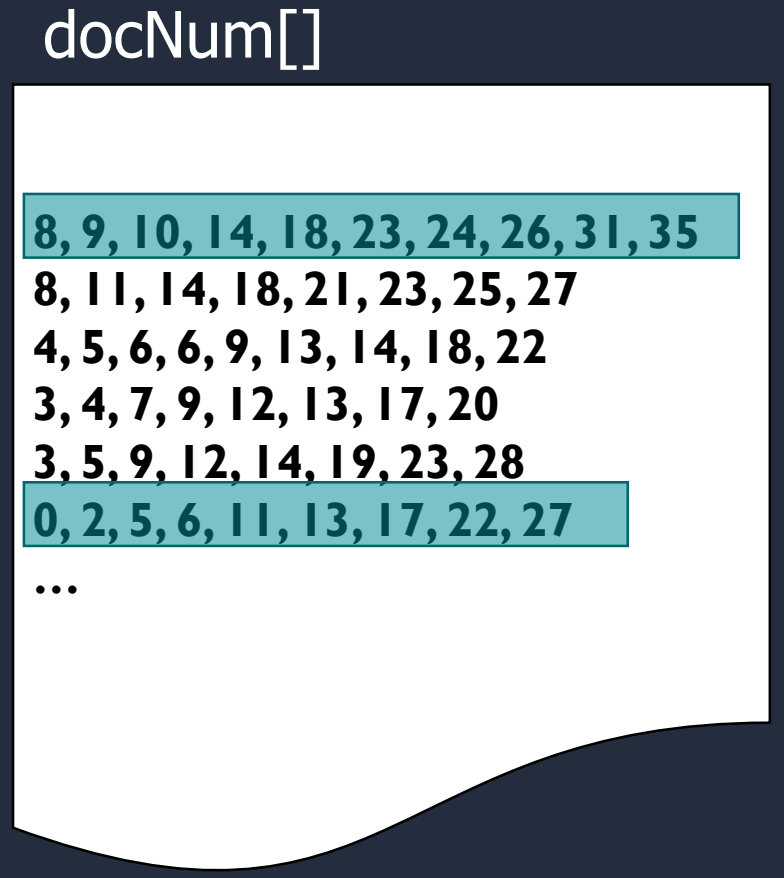
foo- term block

docNum[]

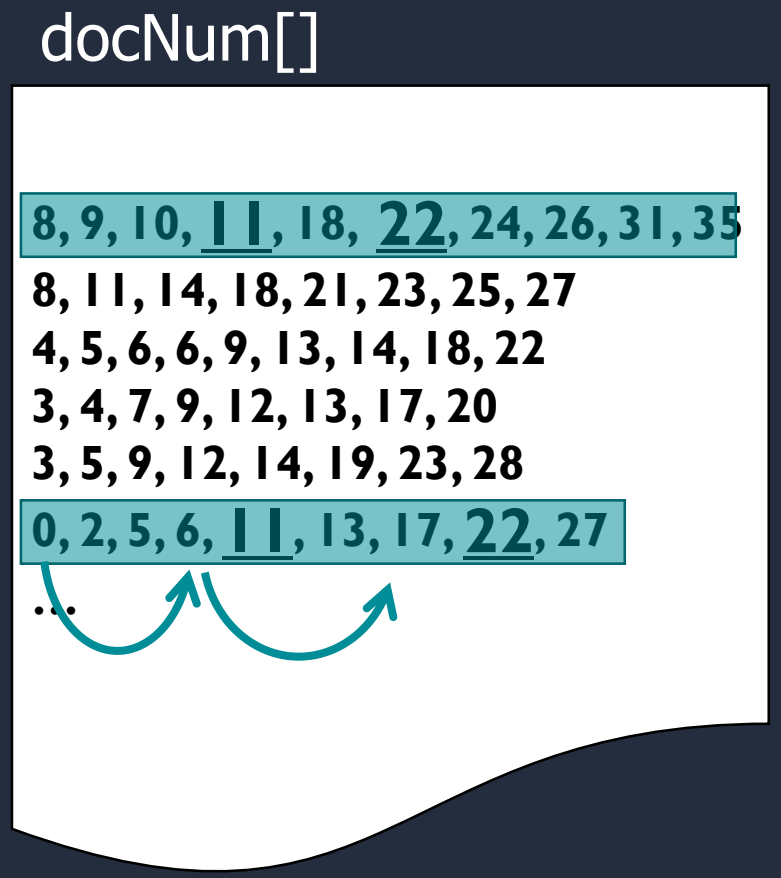
8, 9, 10, 14, 18, 23, 24, 26, 31, 35
8, 11, 14, 18, 21, 23, 25, 27
4, 5, 6, 6, 9, 13, 14, 18, 22
3, 4, 7, 9, 12, 13, 17, 20
3, 5, 9, 12, 14, 19, 23, 28
0, 2, 5, 6, 11, 13, 17, 22, 27
…

RNT GROUP

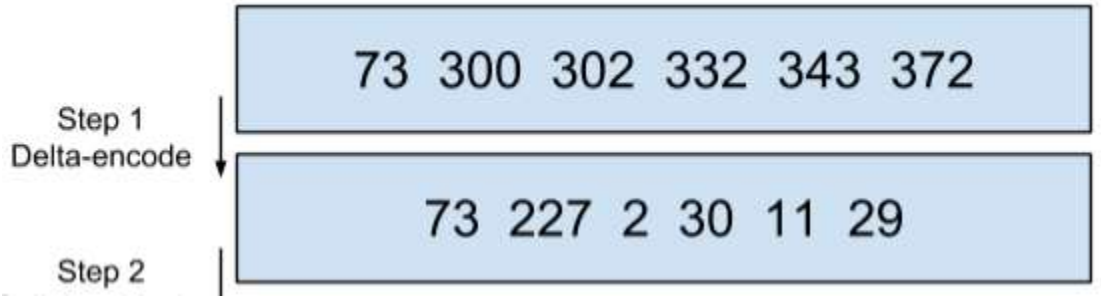
barrel OR old



barrel AND old



Delta encoding



V-Byte Encoding

k	Number of bytes
$k < 2^7$	1
$2^7 \leq k < 2^{14}$	2
$2^{14} \leq k < 2^{21}$	3
$2^{21} \leq k < 2^{28}$	4

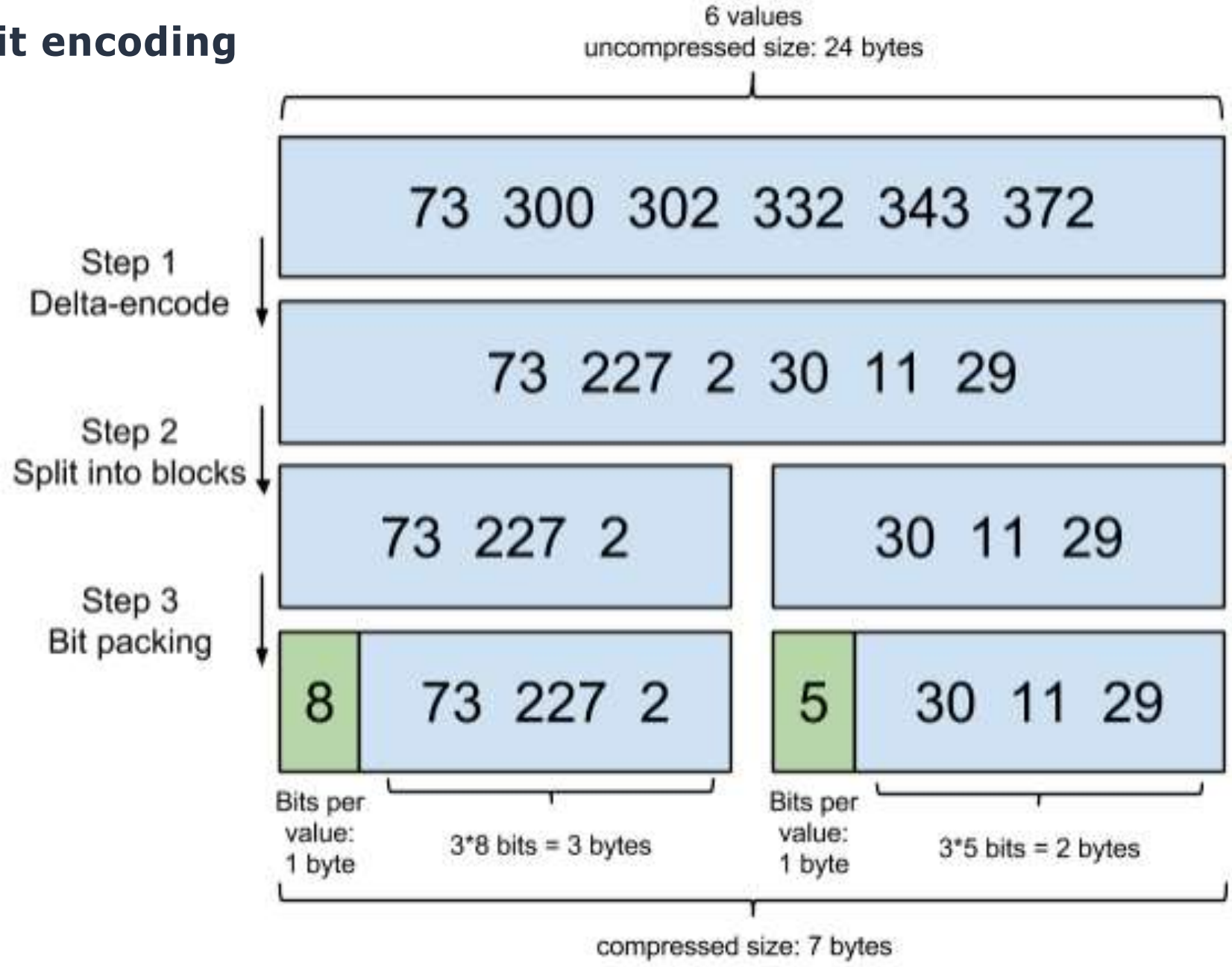
k	Binary Code	Hexadecimal
1	1 0000001	81
6	1 0000110	86
127	1 1111111	FF
128	0 0000001 1 0000000	01 80
130	0 0000001 1 0000010	01 82
20000	0 0000001 0 0011100 1 0100000	01 1C A0

aka Variable Byte,
VByte, **Varint**, **VInt**

Delta encoding

amet
dolores
ipsum
lorem
sit

V-Bit encoding



https://www.elastic.co/blog/frame-of-reference-and-roaring-bitmaps

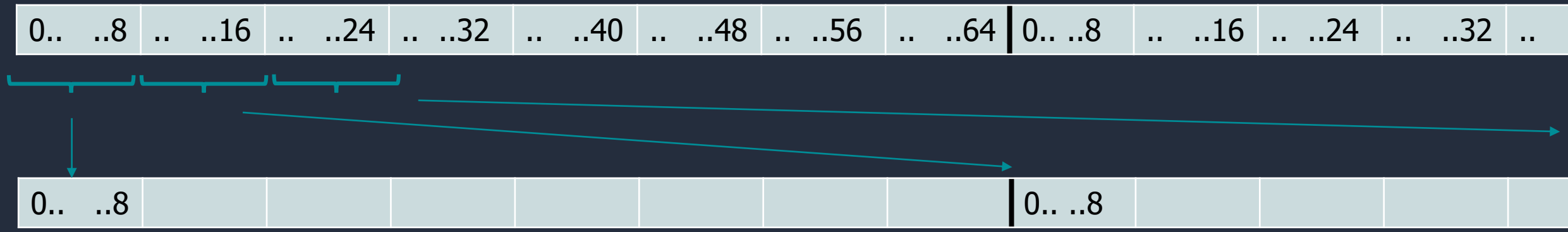
```
/**
 * Decode 128 integers into {@code ints}.
 */
void decode(DataInput in, long[] out) throws IOException {
  final int bitsPerValue = in.readByte();
  switch (bitsPerValue) {
    ...
```

```
switch (bitsPerValue) {
…
    case 2:
      decode2(in,                    out);
      break;
    case 3:
      decode3(in,                    out);
      break;
 ….
}
```

8bit block

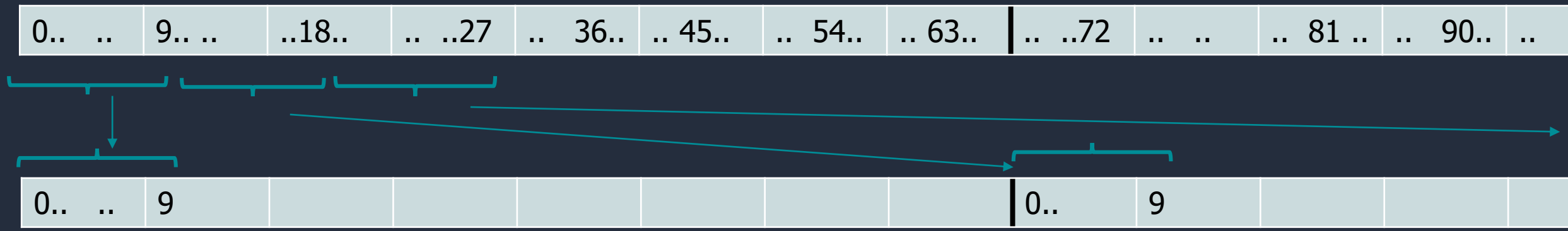
```
private static void decode8(DataInput in,                                long[] out) throws
IOException {

    in.readLongs(           out, 0, 16);
    expand8(out);
  }
```



9bit block

```
private static void decode9(DataInput in,
                    long[] tmp, long[] out) ..{
    in.readLongs(byteOrder, tmp, 0, 18);
    shiftLongs(tmp, 18, out, 0, 7, MASK16_9);
    for(int iter = 0, tmpIdx = 0, longsIdx = 18;
iter < 2; ++iter, tmpIdx += 9, longsIdx += 7) {
      long l0 = (tmp[tmpIdx+0] & MASK16_7) << 2;
      l0 |= (tmp[tmpIdx+1] >>> 5) & MASK16_2;
      out[longsIdx+0] = l0;
      long l1 = (tmp[tmpIdx+1] & MASK16_5) << 4;
      l1 |= (tmp[tmpIdx+2] >>> 3) & MASK16_4;
      out[longsIdx+1] = l1;
      out[longsIdx+3] = l3;
      long l4 = (tmp[tmpIdx+5] & MASK16_6) << 3;
      l4 |= (tmp[tmpIdx+6] >>> 4) & MASK16_3;
      out[longsIdx+4] = l4;
      long l5 = (tmp[tmpIdx+6] & MASK16_4) << 5;
      l5 |= (tmp[tmpIdx+7] >>> 2) & MASK16_5;
      out[longsIdx+5] = l5;
      long l6 = (tmp[tmpIdx+7] & MASK16_2) << 7;
      l6 |= (tmp[tmpIdx+8] & MASK16_7) << 0;
      out[longsIdx+6] = l6;
    }
    expand16(out);
```



SIMD

```
  /**
   * Decode 128 integers into {@code ints}.
   */
  void decode(DataInput in, long[] out) throws IOException {
    final int bitsPerValue = in.readByte();
    switch (bitsPerValue) {
      case 0:
        decode0(in, out);
        break;
    case 1:
      decode1(in, byteOrder, tmp, out);
      break;
    case 2:
      decode2(in, byteOrder, tmp, out);
      break;
    case 3:
      decode3(in, byteOrder, tmp, out);
      break;
 ….
}

private static void decode8(DataInput in, ByteOrder byteOrder,
long[] tmp, long[] out) throws IOException {
    in.readLongs(byteOrder, out, 0, 16);
    expand8(out);
  }

  private static void decode9(DataInput in, ByteOrder byteOrder,
long[] tmp, long[] out) throws IOException {
    in.readLongs(byteOrder, tmp, 0, 18);
    shiftLongs(tmp, 18, out, 0, 7, MASK16_9);
    for(int iter = 0, tmpIdx = 0, longsIdx = 18; iter < 2; ++iter,
tmpIdx += 9, longsIdx += 7) {
      long l0 = (tmp[tmpIdx+0] & MASK16_7) << 2;
      l0 |= (tmp[tmpIdx+1] >>> 5) & MASK16_2;
      out[longsIdx+0] = l0;
      long l1 = (tmp[tmpIdx+1] & MASK16_5) << 4;
      l1 |= (tmp[tmpIdx+2] >>> 3) & MASK16_4;
      out[longsIdx+1] = l1;
      long l2 = (tmp[tmpIdx+2] & MASK16_3) << 6;
      l2 |= (tmp[tmpIdx+3] >>> 1) & MASK16_6;
      out[longsIdx+2] = l2;
      long l3 = (tmp[tmpIdx+3] & MASK16_1) << 8;
      l3 |= (tmp[tmpIdx+4] & MASK16_7) << 1;
      l3 |= (tmp[tmpIdx+5] >>> 6) & MASK16_1;
      out[longsIdx+3] = l3;
      long l4 = (tmp[tmpIdx+5] & MASK16_6) << 3;
      l4 |= (tmp[tmpIdx+6] >>> 4) & MASK16_3;
      out[longsIdx+4] = l4;
      long l5 = (tmp[tmpIdx+6] & MASK16_4) << 5;
      l5 |= (tmp[tmpIdx+7] >>> 2) & MASK16_5;
      out[longsIdx+5] = l5;
      long l6 = (tmp[tmpIdx+7] & MASK16_2) << 7;
      l6 |= (tmp[tmpIdx+8] & MASK16_7) << 0;
      out[longsIdx+6] = l6;
    }
    expand16(out);
  }
```

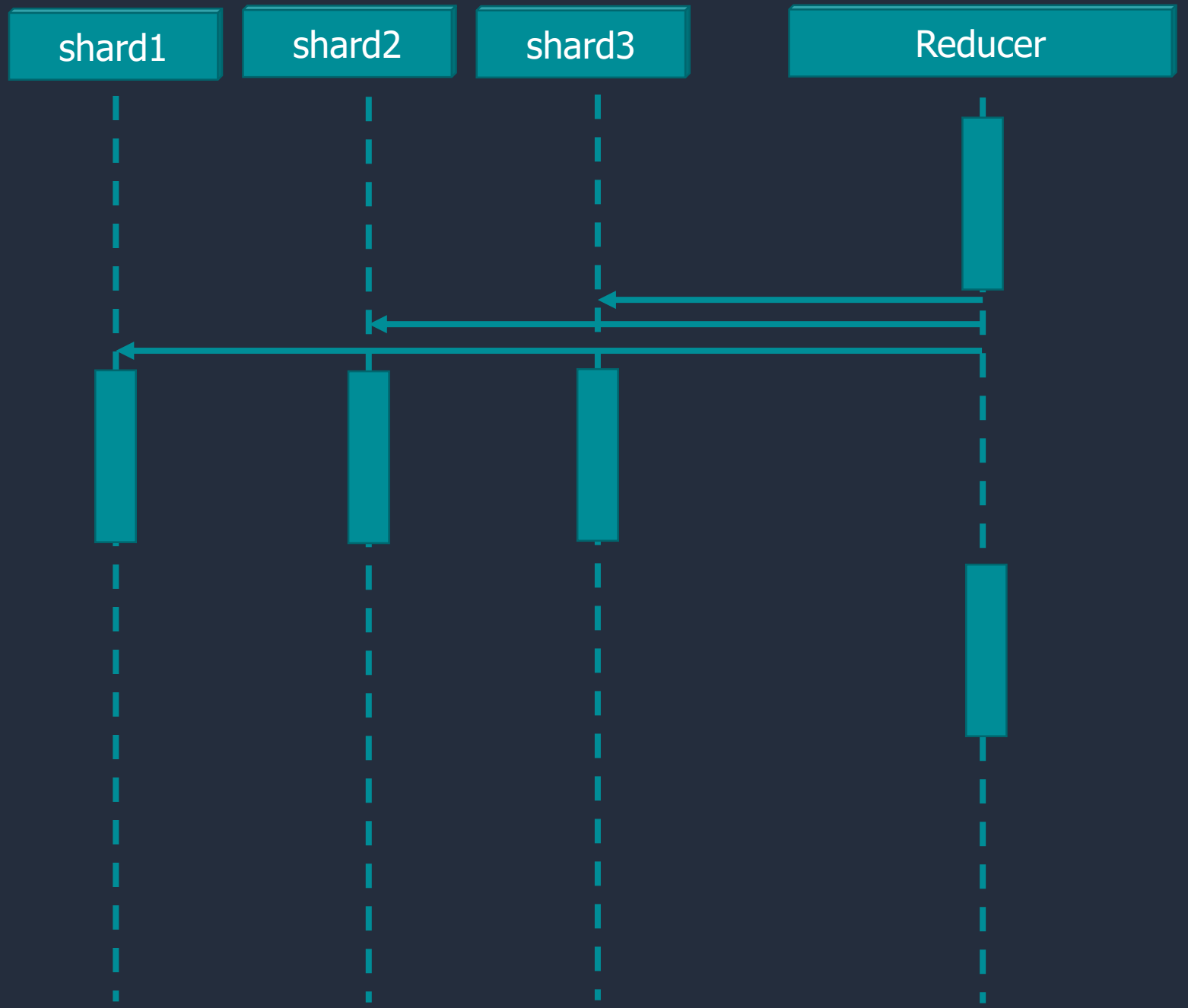
Выводы

- Сжатие данных – ключ к производительности (уменьшение io за счёт cpu)
- Алгоритмы выбираются с учётом аппаратных реализаций

Недостатки

- Сжатие зависит от значений данных
- Остальные алгоритмы не так хорошо проработаны
  - Например, отсутствуют операции на сжатых данных

Что не так с распределённым поиском



RNT GROUP

Производитель

Поиск

- [ ] Все производители
- [ ] DEKO (1)
- [ ] Felo (6)
- [ ] KRAFTOOL (1)
- [ ] Matrix (8)
- [ ] Smartbuy (4)
- [ ] Stayer (1)
- [ ] Дело Техники (7)

Показать всё

Цена

от 40 до 23 199

- [ ] Менее 500 ₽ (26)
- [ ] 501 - 1 000 ₽ (13)
- [ ] 1 001 - 2 000 ₽ (11)
- [ ] 2 001 ₽ и более (5)

Набор отверток Felo Ergonic 41792191 [блистер, насадка/наконечник - PZ/S, PZ (крестовый усиленный), предметов - 2 шт]

2 199 ₽
от 214 ₽/ мес.

Уведомить

Сравнить ★★★★★ 9

Товара нет в наличии

Отвертка с насадками/битами Зубр ЭКСПЕРТ 25283-H46 [пластиковый кейс, насадка/наконечник - PH (крестовый), PZ (крестовый усиленный), SL (прямой), T (звездочка Torx), предметов - 46 шт]

1 899 ₽
от 185 ₽/ мес.

Уведомить

Сравнить ★★★★☆ 23

Товара нет в наличии

Показать ещё

« ‹ 1 2 3 4 › »

This is to handle the case when one term has many documents on one shard but is just below the `size` threshold on all other shards. If each shard only returned `size` terms, the aggregation would return an partial doc count for the term. So `terms` returns more terms in an attempt to catch the missing terms. This helps, but it's still quite possible to return a partial doc count for a term. It just takes a term with more disparate per-shard doc counts.

Scores are not reproducible

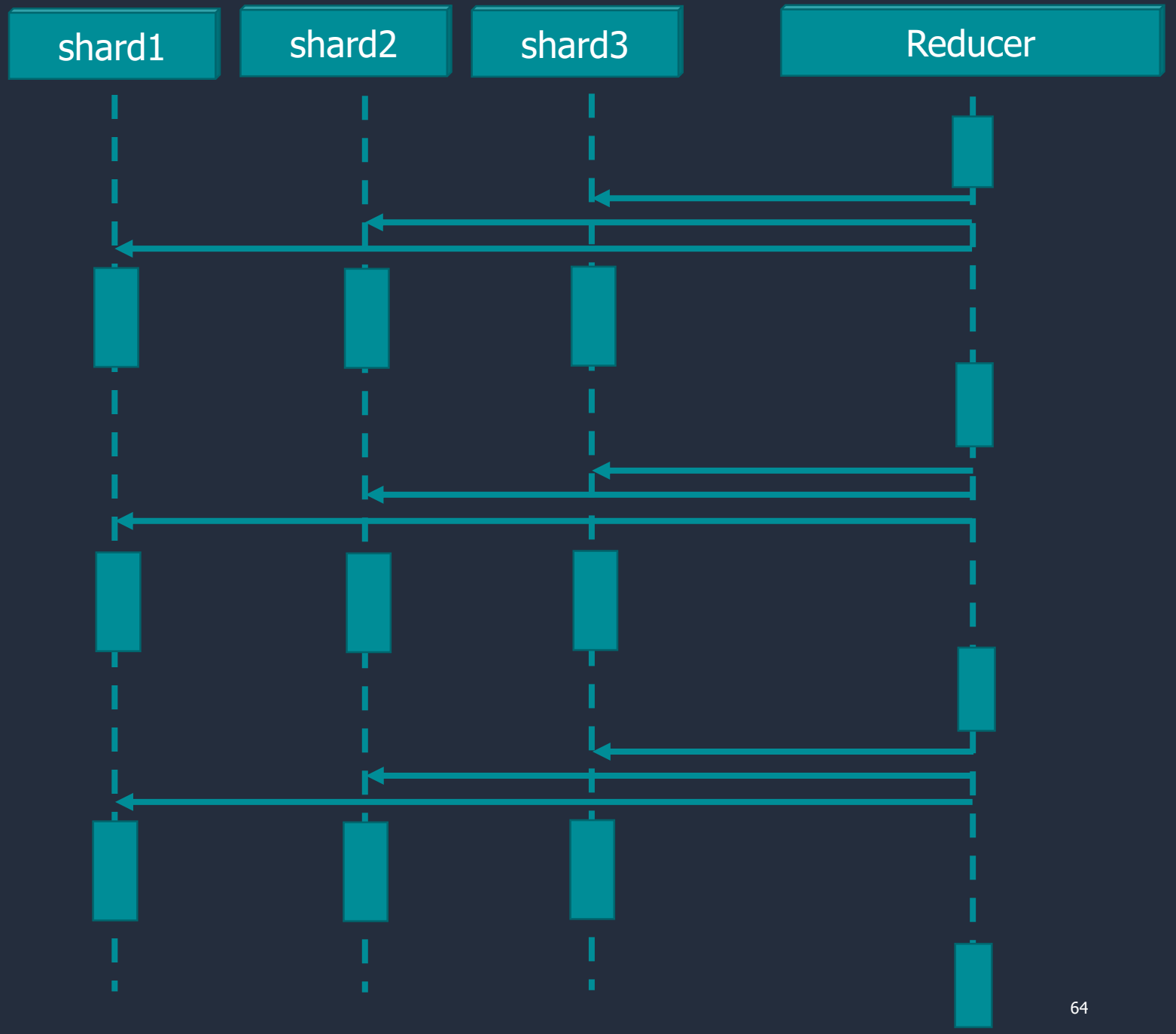
edit

Say the same user runs the same request twice in a row and documents do not come back in the same order both times, this is a pretty bad experience isn't it? Unfortunately this is something that can happen if you have replicas (`index.number_of_replicas` is greater than 0). The reason is that Elasticsearch selects the shards that the query should go to in a round-robin fashion, so it is quite likely if you run the same query twice in a row that it will go to different copies of the same shard.

Relevancy looks wrong

edit

If you notice that two documents with the same content get different scores or that an exact match is not ranked first, then the issue might be related to sharding. By default, Elasticsearch makes each shard responsible for producing its own scores. However since index statistics are an important contributor to the scores, this only works well if shards have similar index statistics. The assumption is that since documents are routed evenly to shards by default, then index statistics should be very similar and scoring would work as expected



RNT GROUP

Преимущества распределённого поиска

Shared-nothing:

- отказоустойчивость
- гибкость
- масштабирование
- производительность – отсутствие блокировок между узлами

И недостатки

- Ограничено выбором top-K и других агрегатов: Value->count()
  - И даже с ними проблемы – приходится делать много reduce запросов
- Масштабирование ограничено - чем больше шардов, тем больше строк будет отброшено при merge($topK_1$; $topK_2$;…)
- Сложный поток исполнения множества запросов ограничивает производительность
- На практике – индексирование замедляет поиск
  - Отчасти потому что удобно полностью перестроить каталог
  - Замедление поиска болезненно для посетителей.

Urs Hölzle
@uhoelzle

@JeffDean @GCPcloud R.I.P. MapReduce. After having served us well since 2003, today we removed the remaining internal codebase for good. Of course, external users of MR on GCP will continue to be supported with our fully upstream compatible managed Hadoop platform, Dataproc.

2:11 AM · Sep 27, 2019

286 Reposts **71** Quotes **593** Likes **33** Bookmarks

Что не так с распределённым поиском

СКЛАД

Обратный Индекс для Продуктового Каталога

- длинные строковые значения индексного поля

- многозначные поля, особенно текст
- произвольные комбинации фильтров
- сортировки по произвольным полям

- сжатие индекса
  - уменьшаем чтение, больше операций ЦП
- документная модель без нормализации
- потоковый алгоритм комбинации индексов
- индекс для сортировки

Требования к Продуктовому Каталогу

- длинные строковые значения индексного поля
- многозначные поля, особенно текст
- произвольные комбинации фильтров

- сортировки по произвольным полям

- большой размер b-tree индекса
- противоречат нормальной модели
- требуют большого количества композитных индексов
- требуют большого количества включающих индексов

.. но

- В PosgreSQL есть array
- PosgreSQL комбинирует индексы

- индексируется GIN Generalized Inverted Index
- материализуя битовые карты и комбинируя их

Неоспоримое преимущество b-tree - обновление отдельных записей.

Обратный Индекс для Продуктового Каталога

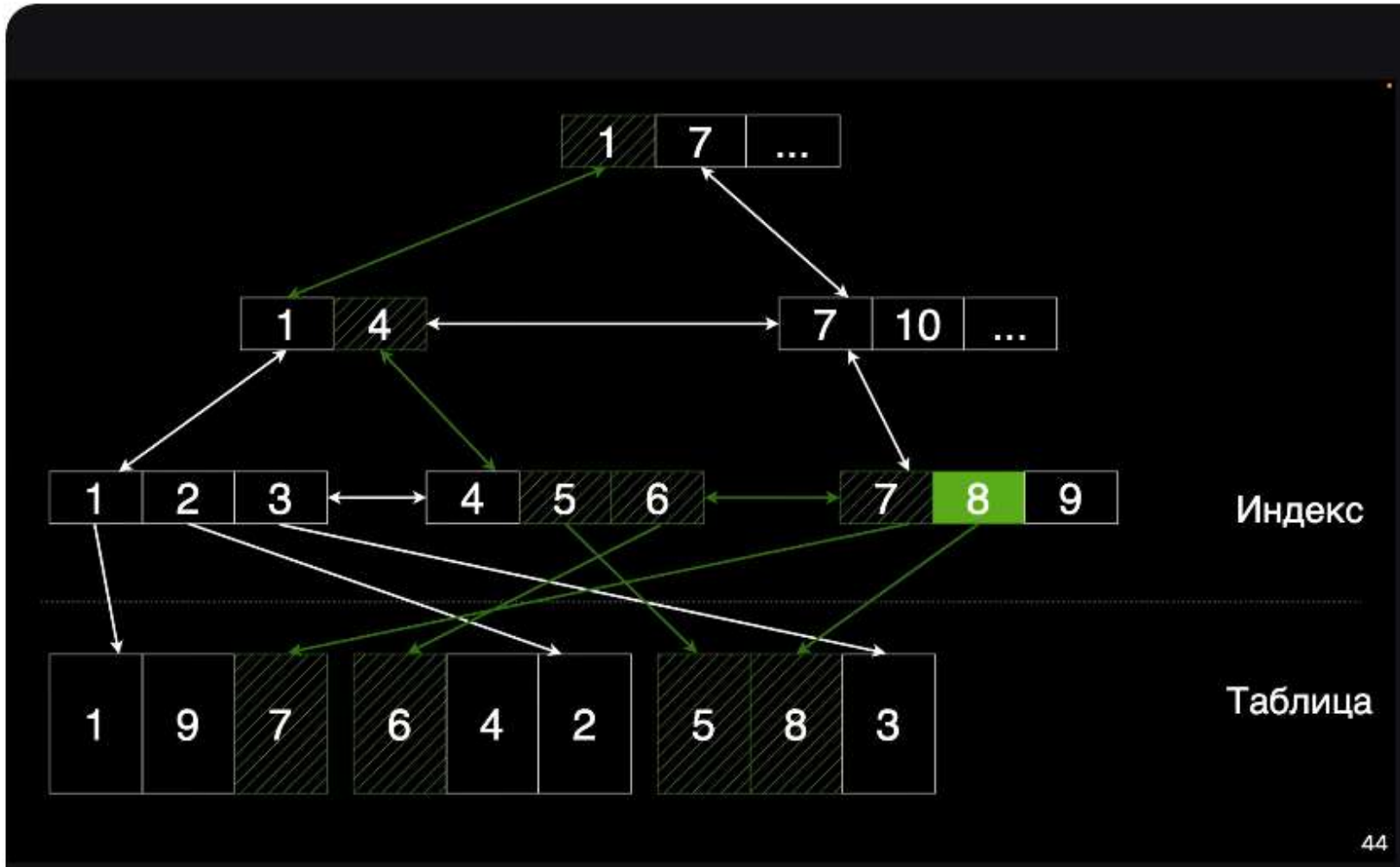
- длинные строковые значения индексного поля
- многозначные поля, особенно текст
- произвольные комбинации фильтров
- сортировки по произвольным полям
- сжатие индекса
  - уменьшаем чтение, больше нагрузка ЦП
- документная модель без нормализации
- потоковый алгоритм комбинации индексов
- индекс для сортировки

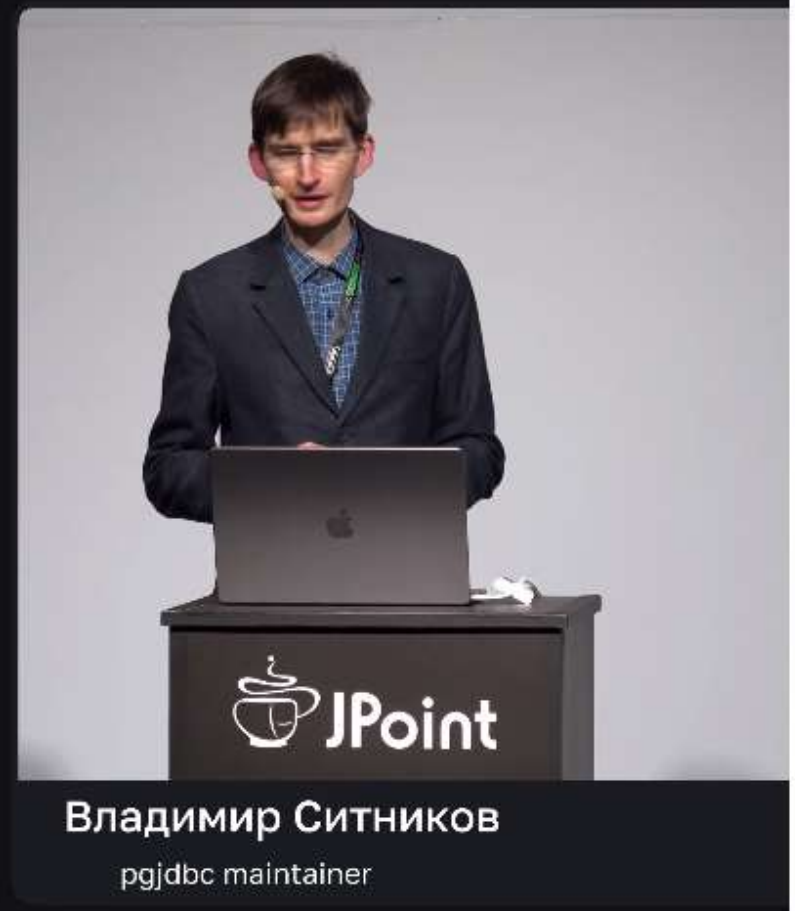
RNT GROUP

Обратный Индекс для Продуктового Каталога

- уменьшаем чтение, больше нагрузка ЦП

B-Tree индексы в базах данных на примере Spring Boot-приложений, PostgreSQL и JPA





Владимир Ситников

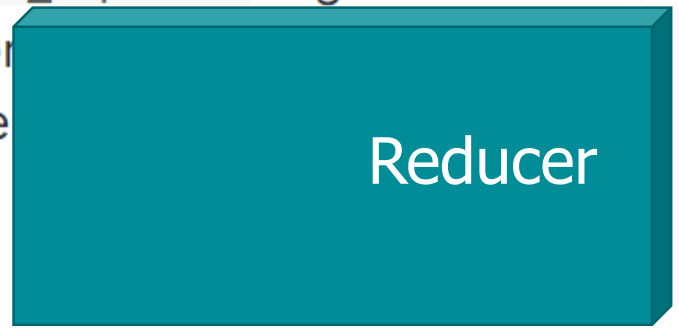
pgjdbc maintainer

This is to handle the case when one term has many documents on one shard but is just below the `size` threshold on all other shards. If each shard only returned `size` terms, the aggregation would return an partial doc count for the term. So `terms` returns more terms in an attempt to catch the missing terms. This helps, but it's still quite possible to return a partial doc count for a term. It just takes a term with more disparate per-shard doc counts.

Scores are not reproducible

edit

Say the same user runs the same request twice in a row and documents do not come back in the same order both times, this is a pretty bad experience isn't it? Unfortunately this is something that can happen if you have replicas (`index.number_of_replicas` is greater than 0). The reason is that Elasticsearch selects the shards that the quer robin fashion, so it is quite likely if you run the same query twice different copies of the same shard.

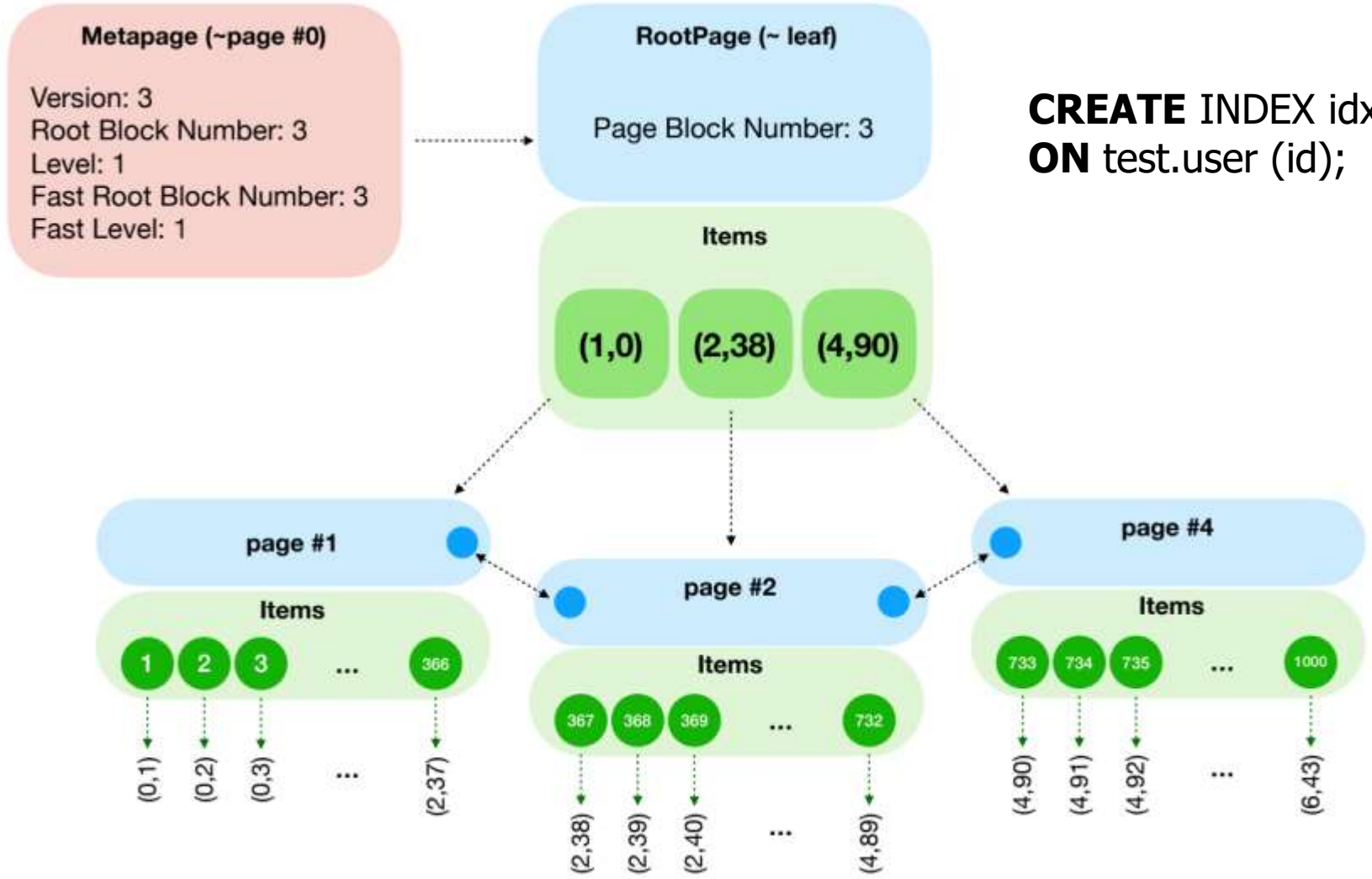


Relevancy looks wrong

edit

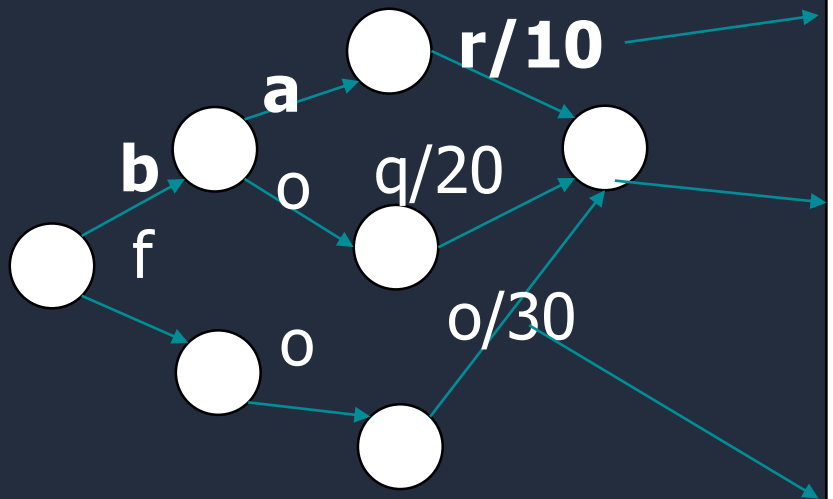
If you notice that two documents with the same content get different scores or that an exact match is not ranked first, then the issue might be related to sharding. By default, Elasticsearch makes each shard responsible for producing its own scores. However since index statistics are an important contributor to the scores, this only works well if shards have similar index statistics. The assumption is that since documents are routed evenly to shards by default, then index statistics should be very similar and scoring would work as expected

Азат Якупов. B-Tree индекс и его производные в PostgreSQL




```
CREATE INDEX idx_user
ON test.user (id);
```

3. Списки вхождений



Block prefix FST

-ber,-ely,-ney,-rel,-ter..
100,110,120,130,140

boq- term block

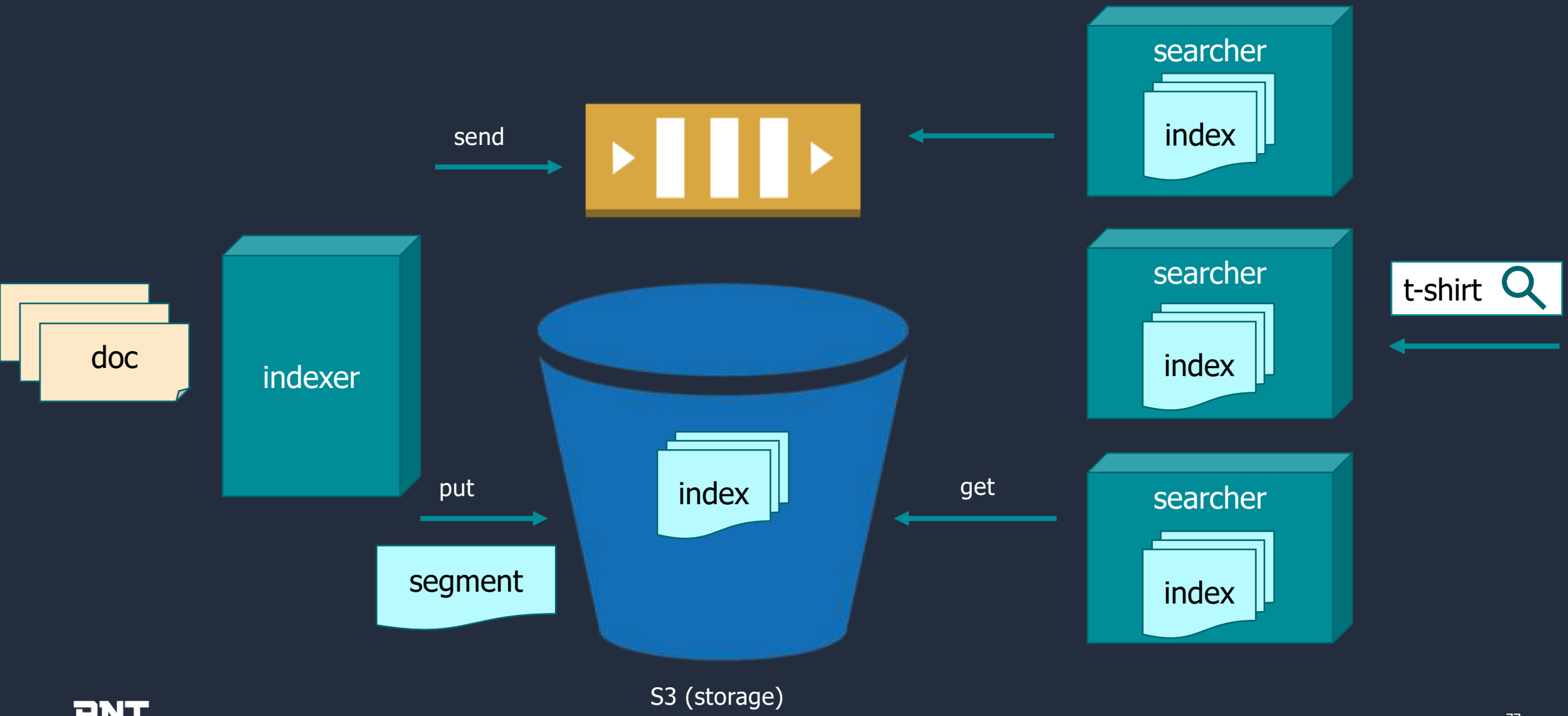
foo- term block

docNum[]

8, 9, 10, 14, 18, 23, 24, 26, 31, 35
8, 11, 14, 18, 18, 18, 21, 23, 25, 27
4, 5, 6, 6, 9, 13, 13, 14, 18, 22
3, 4, 7, 9, 9, 9, 12, 13, 17, 20
3, 3, 5, 9, 9, 12, 14, 19, 23, 28
0, 2, 5, 6, 11, 13, 17, 22, 27
...

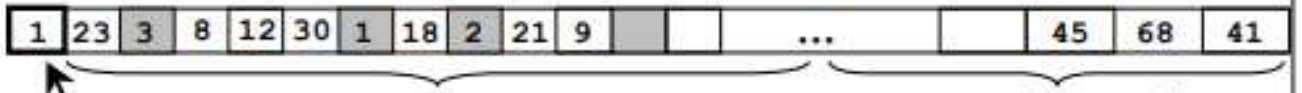
RNT GROUP

Shared-nothing vs Shared-storage



Data to be compressed:	…10, 10, 1, 10, 100110, 10, 1, 11, 10, 100000, 10, 110100, 10, 11, 11, 1…
Truncated data: ($16 \times 2 = 32$ bits)	…10, 10, 01, 10, 10, 10, 01, 11, 10, 00, 10, 00, 10, 11, 11, 01 …
Byte array: ($6 \times 8 = 48$ bits)	…2, 6, 3, 4, 9, 11 …
Exception data: (to be compressed)	…1001, 1000, 1101 …

- allocate 128 x 5 bits, plus space for exceptions
- exceptions stored at end as ints (using 4 bytes each)
- example: b=5 and sequence 23, 41, 8, 12, 30, 68, 18, 45, 21, 9, ..



- exceptions (grey) form linked list within the locations (e.g., 3 means "next except. 3 away")
- one extra slot at beginning points to location of first exception (or store in separate array)

Доступные реализации ClickHouse; Cassandra, Mongo.

Casaandra: materialized view для денормализации, GROUP BY ; или Spark – for remote semijoin

ClickHouse – похоже есть всё до LEFT OUTER, распределённые; Но он не на Java. (

- Стаценко: клик не очень джойнит
- Голов – кликхаус не очень умееет шардированный джоин, надо что бы всё на сервере помещалось.

Mongo -?