Joining in Lucene

Martijn van Groningen
martijn.vangroningen@searchworkings.com
Lucene Committer & PMC Member

Introduction

- Data is often relational, but Lucene's document model is not.
- Support for parent child like search from Lucene 3.4
  - Not a SQL join.
- The parent and children are stored in separate documents.
- Two types:
  - Index time join
  - Query time join

Index time join

- Two block join queries:
  - ToParentBlockJoinQuery
  - ToChildBlockJoinQuery
- One Lucene collector:
  - ToParentBlockJoinCollector
- Index time join requires block indexing.

Block indexing

- Atomically adding documents.
  - A block of documents.
- Each document gets sequentially assigned Lucene document id.
- IndexWriter#addDocuments(docs);

Block indexing

- Index doesn't record blocks.
- App is responsible for identifying block documents.
  - Marking a document in a block.
- Segment merging doesn’t re-order documents in a segment.
- Adding a document to a block requires you to reindex the whole block.
- Removing a document from a block doesn’t requires reindexing a block.

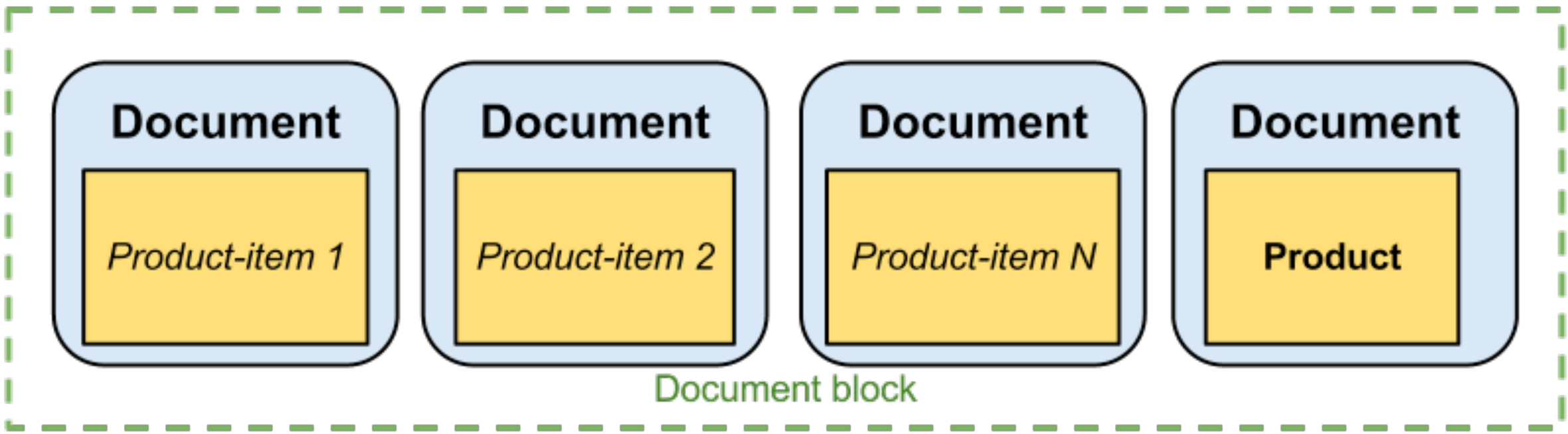
Domain example

- Product
  - Name
  - Description
- Product-item
  - Color
  - Size
  - Price

- Goal: Show the most applicable product based on product-item criteria.

Domain example

- Parent is the last document in a block.



Block indexing

Marking parent documents

```
private static Document createProduct(String name, String description) {
  Document document = new Document();
  document.add(new Field("name", name, TextField.TYPE_STORED));
  document.add(new Field("docType", "product", StringField.TYPE_UNSTORED));
  document.add(new Field("description", description, TextField.TYPE_STORED));
  return document;
}

private static Document createProductItem(String color, String size, int price) {
  Document document = new Document();
  document.add(new Field("color", color, TextField.TYPE_STORED));
  document.add(new Field("size", size, TextField.TYPE_STORED));
  document.add(new IntField("price", price));
  return document;
}
```

Block indexing

```
IndexWriter writer = new IndexWriter(directory, config);
List<Document> documents = new ArrayList<~>();
documents.add(createProductItem("red", "s", 999));
documents.add(createProductItem("red", "m", 1099));
documents.add(createProductItem("red", "l", 1199));
documents.add(createProduct("...Polo Shirt", "Made of 100% cotton..."));
writer.addDocuments(documents);
documents.clear();

documents.add(createProductItem("light blue", "s", 1999));
documents.add(createProductItem("blue", "s", 1999));
documents.add(createProductItem("dark blue", "s", 1999));
documents.add(createProductItem("light blue", "m", 2099));
documents.add(createProductItem("blue", "m", 2099));
documents.add(createProductItem("dark blue", "m", 2099));
documents.add(createProduct("...White Colored...", "...stripe pattern..."));
writer.addDocuments(documents);

IndexReader indexReader = DirectoryReader.open(writer, false);
IndexSearcher indexSearcher =  new IndexSearcher(indexReader);
```

Add block

Add block

ToParentBlockJoinQuery

- Parent filter marks the parent documents.

```
Query parentQuery = new TermQuery(new Term("docType", "product"));
Filter parentsFilter = new CachingWrapperFilter(
    new QueryWrapperFilter(parentQuery)
);
```

- Child query is executed in the parent space.

```
Query childQuery = new TermQuery(new Term("size", "m"));
ScoreMode scoreMode = ScoreMode.Max;

BooleanQuery mainQuery = new BooleanQuery();
mainQuery.add(userQuery, BooleanClause.Occur.MUST);

ToParentBlockJoinQuery productItemQuery = new ToParentBlockJoinQuery(...);
mainQuery.add(productItemQuery, BooleanClause.Occur.MUST);
TopDocs result = indexSearcher.search(mainQuery, 10);
```

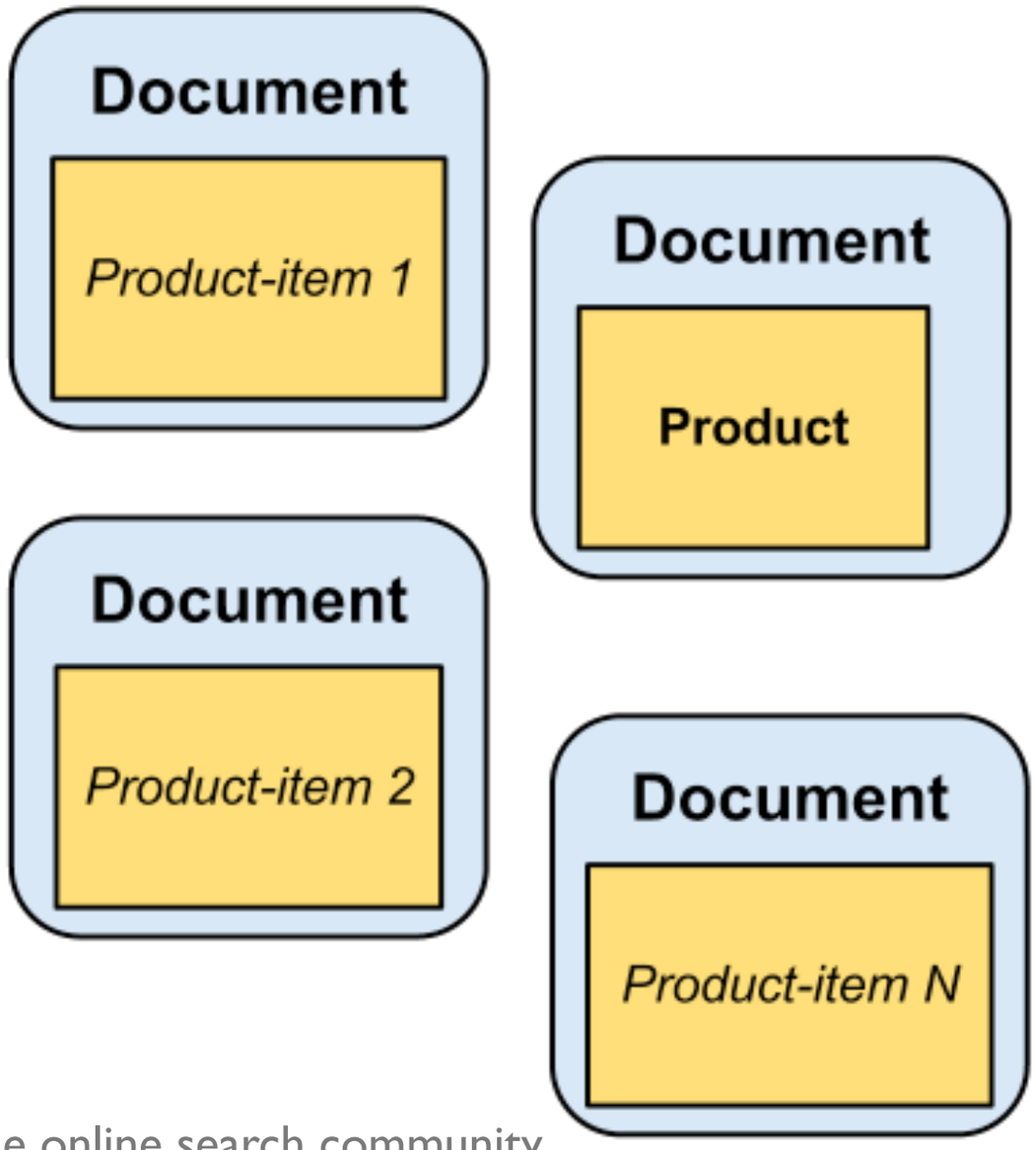
- ToChildBlockJoinQuery works in the opposite direction.

Block joining & ElasticSearch

- ElasticSearch has support for nested objects since version 0.17.0
  - Nested type in the mapping definition.
- NestedQuery & NestedFilter
  - Uses ToParentBlockJoinQuery
- Allows to query for nested objects as if they were separate documents and then return the root object

Query time joining

- Query time joining is executed in two phases and is field based:
  - fromField
  - toField
- Doesn’t require block indexing.



Query time joining

- First phase collects all the terms in the `fromField` for the documents that match with the original query.

- The second phase returns the documents that match with the collected terms from the previous phase in the `toField`.

- One public method:
  - JoinUtil#createJoinQuery(...)

Query time joining - Indexing

```
private static Document createProduct(String id, String name, String description) {
  Document document = new Document();
  document.add(new Field("id", id, TextField.TYPE_STORED));
  document.add(new Field("name", name, TextField.TYPE_STORED));
  document.add(new Field("description", description, TextField.TYPE_STORED));
  return document;
}

private static Document createProductItem(String color, String size, int price,
                                          String productId) {
  Document document = new Document();
  document.add(new Field("color", color, TextField.TYPE_STORED));
  document.add(new Field("size", size, TextField.TYPE_STORED));
  document.add(new IntField("price", price));
  document.add(new Field("productId", productId, TextField.TYPE_STORED));
  return document;
}
```

Referrer the product id.

Query time joining - Indexing

```
IndexWriter writer = new IndexWriter(directory, config);
writer.addDocument(
    createProduct("1", "...Polo Shirt", "Made of 100% cotton,...")
);
writer.addDocument(createProductItem("red", "s", 999, "1"));
writer.addDocument(createProductItem("red", "m", 1099, "1"));
writer.addDocument(createProductItem("red", "l", 1199, "1"));

writer.addDocument(
    createProduct("2", "...White Colored...", "...stripe pattern...")
);
writer.addDocument(createProductItem("light blue", "s", 1999, "2"));
writer.addDocument(createProductItem("blue", "s", 1999, "2"));
writer.addDocument(createProductItem("dark blue", "s", 1999, "2"));
writer.addDocument(createProductItem("light blue", "m", 2099, "2"));
writer.addDocument(createProductItem("blue", "m", 2099, "2"));
writer.addDocument(createProductItem("dark blue", "m", 2099, "2"));
```

Query time joining

```
String fromField = "productId";
Query fromQuery = NumericRangeQuery.newIntRange("price", 0, 1000...);
boolean multipleValuesPerDoc = false;
ScoreMode scoreMode = ScoreMode.None;
String toField = "id";

Query toQuery = JoinUtil.createJoinQuery(...);
mainQuery.add(toQuery, BooleanClause.Occur.MUST);
TopDocs result = indexSearcher.search(mainQuery, 10);
```

- Result will contain one product.

- Possible to join over two indices.

Final thoughts

- Joining module has good solutions to model parent child relations.
- Joining has impact on the query time.

- Index time joining is much faster than query time joining
- Query time joining is more flexible than index time joining

- Mostly a Lucene feature only.
- All code is annotated as experimental.

Any questions?

ToParentBlockJoinCollector

```
Sort sort = Sort.RELEVANCE;
int numParents = 10;
ToParentBlockJoinCollector collector = new ToParentBlockJoinCollector(...);
indexSearcher.search(query, collector);

Sort sortWithinChildDocs = new Sort(
    new SortField("price", SortField.Type.INT)
);
int offset = 0;
int offsetInChildDocs = 0;
int maxChildDocs = 5;
TopGroups groups = collector.getTopGroups(...);
```

- TopGroups contains a group per top N parent document.
- Each group contains a parent and child documents.