Query rewriting outside of the search engine

Future perspectives on Querqy

The conflict of rewriting / restructuring as a plugin

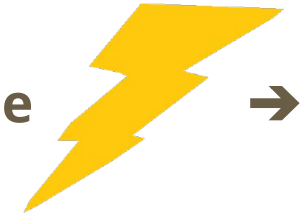
- Query rewriting is central
- Majority of business logic affects the rewriting
- Integration of ML algorithms

➔ **Be dynamic and flexible**

The conflict of rewriting as a plugin

- Query rewriting is central
- Majority of business logic affects the rewriting
- Integration of ML algorithms

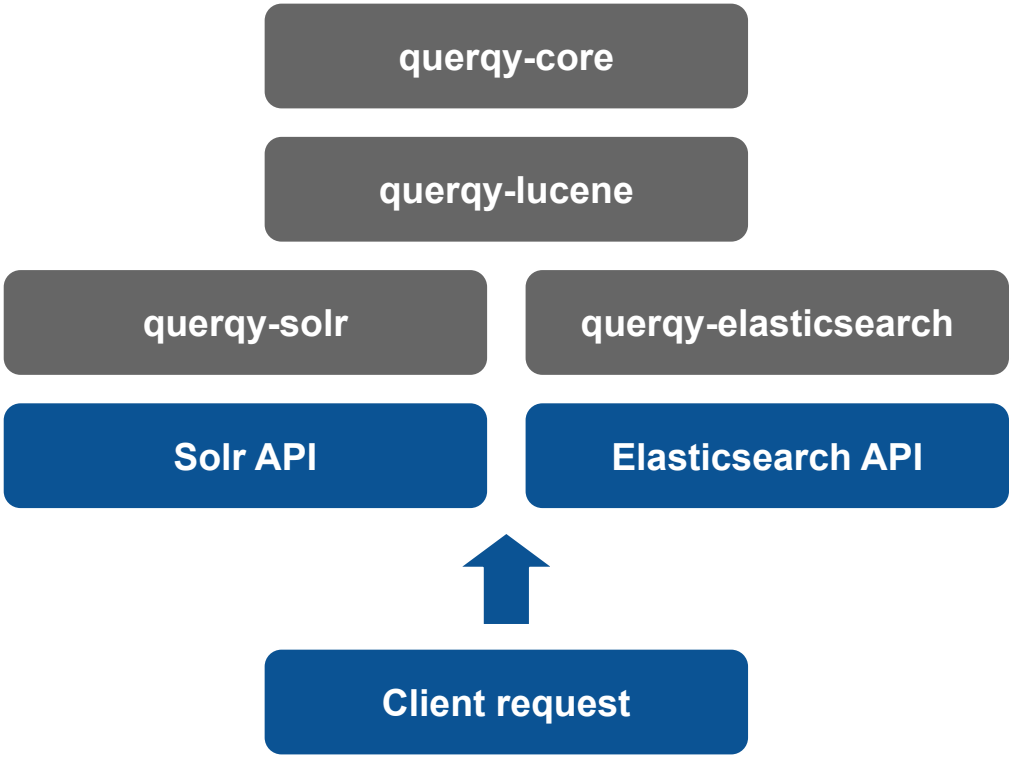
➔ **Be dynamic and flexible**



- Search engine is the most fragile component
- Hosting consumes a lot of time and requires expertise
- Deployments are painful

➔ **Avoid change**

Current querqy architecture

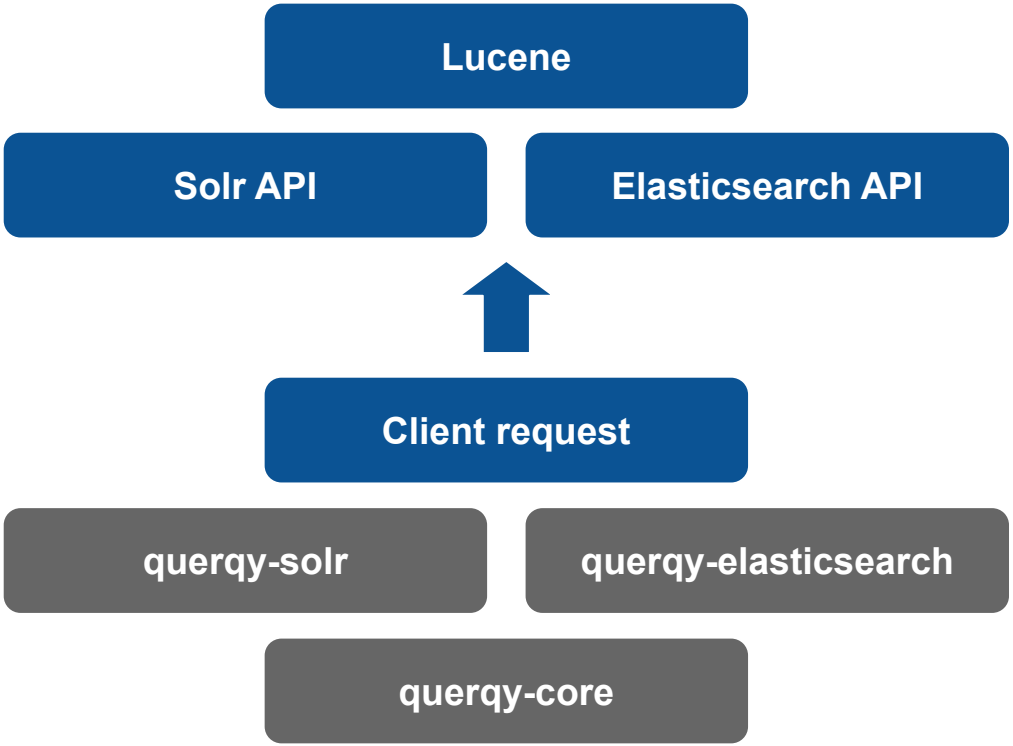


Rewriting logic

Matching / Scoring

Adapter

Rewriting outside architecture



Matching / Scoring

Adapter

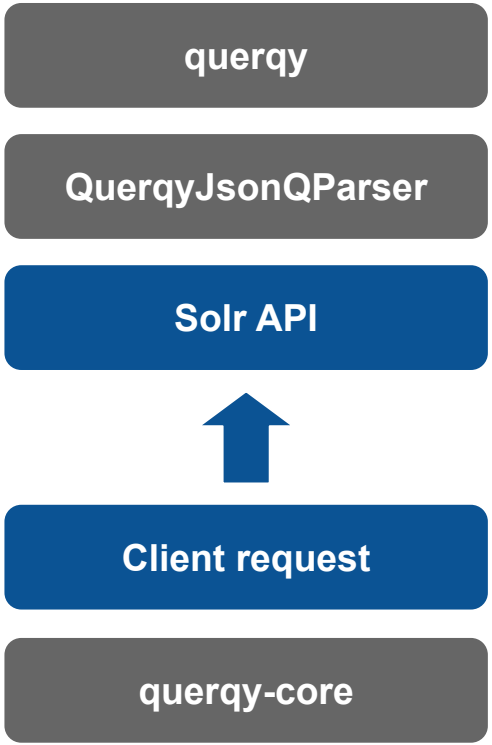
Rewriting logic

API capabilities

Feature	Solr	Elasticsearch / Open Search
JSON Query DSL	✅	✅
Compound queries	✅	✅
Combined customization matching & scoring	❌	✅

```
{
  should: [
    must: [
      term: dishwasher
      score: [max|mean]
    ]
    must: [
      term: dish
      term: washer
      score: [max|mean]
    ]
  ]
}
```

Current state (experimental)




```
final QueryRewritingHandler rewritingHandler = QueryRewritingHandler.builder()
        .addReplaceRewriter("notbook => notebook")
        .addCommonRulesRewriter(
                "notebook => \n " +
                        "SYNONYM: laptop \n " +
                        "DECORATE(title): Awesome notebooks"
        )
        .build();

final RewrittenQuery rewrittenQuery = rewritingHandler.rewriteQuery( queryString: "notbook");
final JsonQueryRequest request = new JsonQueryRequest();

request.setQuery(createQuerqyQuery(rewrittenQuery))
        .withParam( name: "qf",  value: "product_type");

final QueryResponse response = request.process(solrClient);
```

https://github.com/JohannesDaniel/explore-querqy-api