How does Hibernate Search (Lucene indexing) work?

Question: How does Hibernate Search (Lucene indexing) work?

I am using Hibernate Search built on top of Lucene indexing. If indexes are created against database table the performance will be good in returning the results.

My question is, once indexes are created, if we query for the results does Hibernate Search fetch results from the original database table using the created indexes? or does it not need to hit the database to fetch the results?

Thanks!

Answer:

Unless you use Projections the indexes are used only to identify the set of primary keys matching the query, these are then used to load the entities from the Database.

There are many good reasons for this:

- As you pointed out, we don't store all data in the index: a larger index is a slower index
- Adding all needed metadata to the index would make indexing a very expensive operation
- Value extraction from the index is not efficient at all: it's good at queries, no more
- Relational databases are very good at loading data by primary key
- If you DB isn't good enough, second level cache is excellent to load by primary key
- By loading from the DB we guarantee consistency especially with async indexing
- By loading from the DB you have entities participate in Transactions and isolation

That said, if you don't need fully managed entities you can use Projections to load the fields you annotated as **Stored.YES**. A common pattern is to provide preview of matches using projections, and then when the user clicks for details to load the full entity matching that result.