Architecture and Administration Basics

Workshop Day 1 - FTS

4 | Full Text Search Capabilities

Full Text Search - Capabilities

Query

- **Basic**: Match, Match Phrase, Fuzzy, Prefix, Regexp, Wildcard, Boolean Field
- **Compound**: QueryString, Boolean, Conjunction, Disjunction
- **Range**: DateRange, NumericRange
- **Special Purpose**: DocID, MatchAll, MatchNone, Phrase, Term
- **Scoring** (TF/IDF), boosting, field scoping
- **New/DP:** TermRange, Geospatial
- See http://www.blevesearch.com/docs/Query/

Indexing

- Real time indexing (inverted index, auto-updated upon mutation)
- Default map and map by document type
- Dynamic mapping
- Stored fields, Term vectors
- Analyzers: Tokenization, Token Filtering (stop word removal, stemming – language specific)

LAB: import Datasets

Product Dataset - Import Instructions

Below steps are required to import data when you use the RightScale environment.

1. Create a bucket named **"products"** on the Couchbase cluster.
2. ssh to any one of the Couchbase nodes.
3. Download the product data set from the USB Stick.
4. cd to Couchbase bin directory.
5. ./cbimport json -c couchbase://127.0.0.1 -u Administrator -p password -b products -d file:///Path/to/downloaded/products_data.json -f lines -g %asin% -t 4

LAB: Postman Setup

Postman – API testing platform

All queries and indexes for labs can be found in the below collection

- https://www.getpostman.com/collections/064fd2bcdce5f523483d
- http://bit.ly/2zzLPYO

Tip: For RightScale users can enable Global Proxy on Postman and provide IP address of RightScale host in the proxy server with port 8094 to route traffic to RightScale servers instead of manually changing host name for every request.

Full Text Search – Basic Queries

Match Query - A match query analyzes the input text and uses that analyzed text to query the index. An attempt is made to use the same analyzer that was used when the field was indexed.

Match Phrase Query - The input text is analyzed and a phrase query is built with the terms resulting from the analysis. This type of query searches for terms occurring in the specified positions and offsets. This depends on term vectors, which are consulted to determine phrase distance.

Fuzzy Query - A fuzzy query is a term query that matches terms within a specified edit distance (Levenshtein distance). Also, you can optionally specify that the term must have a matching prefix of the specified length.

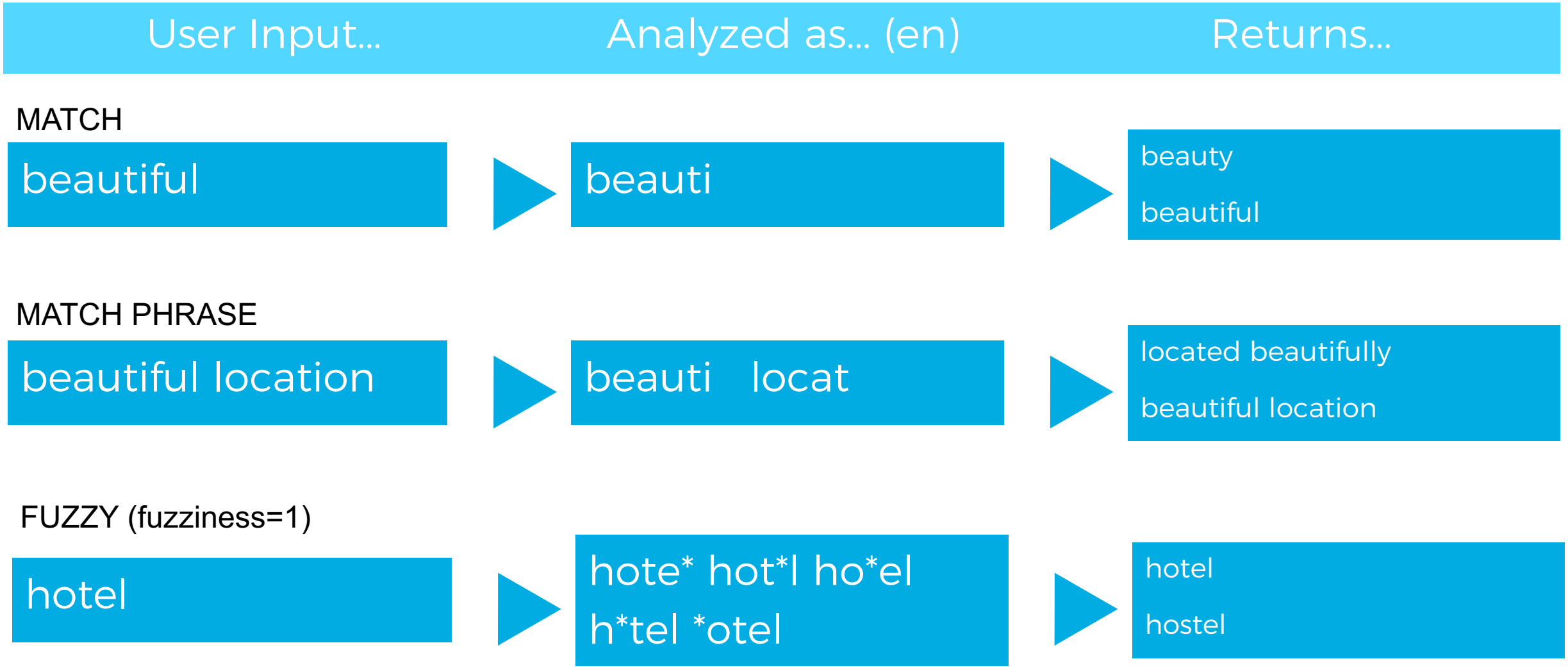
Prefix Query - The prefix query finds documents containing terms that start with the provided prefix.

Regexp Query - Regexp query finds documents containing terms that match the specified regular expression.

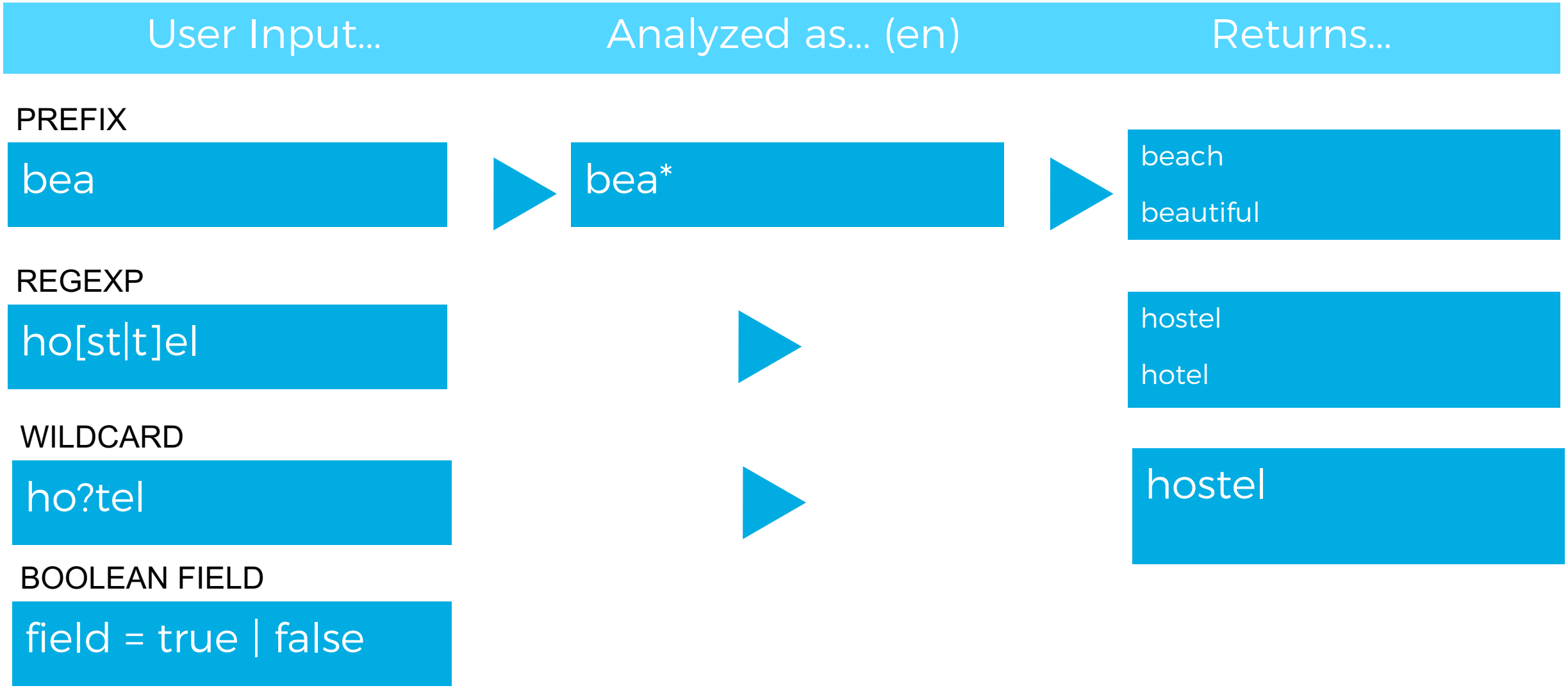
Wildcard Query - The wildcard query uses a wildcard expression to search within individual terms for matches. Wildcard expressions can be any single character (?) or zero to many characters (*).

Boolean Field Query - The Boolean field query searches a field that contains Boolean true or false values. A Boolean field query searches the actual content of the field

Basic Queries: MATCH, MATCH PHRASE and FUZZY



Basic Queries: PREFIX, REGEXP, WILDCARD, BOOLEAN FIELD



LAB: Basic Queries - Match

Match Query

Find all products that have “sunglass” in the title field.

Index

Mapping => type:product, check Only Index Specified field, child-field: title

Analyzer: en

Advanced -> Default Analyzer : en

Query

curl -XPOST -H "Content-Type: application/json"
*http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query -d '{"query": {"**match**": "sunglass", "field": "title"}}' | jq*

LAB: Basic Queries – Match Phrase

Match Phrase query

Find all products that have the phrase “polarized sunglass” in the title field.

Index

Mapping => type:product, check Only Index Specified field, child-field: title

Analyzer: en

Advanced -> Default Analyzer : en

Query

*curl -XPOST -H "Content-Type: application/json" http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query -d '{"query": {"**match_phrase**": "polarized sunglass", "field": "title"}}' | jq*

LAB: Basic Queries - Fuzzy

Fuzzy Query

(Levenshtein distance - Replace, Add, Drop)

Find all the products whose brand is a L-distance of 2 away from Pepper.

Index

Mapping => type:product, check Only Index Specified field, child-field:brand

Analyzer: standard

Query

*curl -XPOST -H "Content-Type: application/json" http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query -d '{"query": {"**term**": "Pepper", "**fuzziness**": 2, "field": "brand"}}' | jq*

LAB: Basic Queries - Prefix

Prefix Query

Find all the products whose ids begin with 1617160.

Index

Mapping => type:product, check Only Index Specified field, child-field: asin

Analyzer: standard

Query

curl -XPOST -H "Content-Type: application/json"

*http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query -d '{"query": {"**prefix**": "1617160", "field": "asin"}}' | jq*

LAB: Basic Queries - Regexp

Regexp Query

Find all products that have reviews by a Jon or a Joe.

Index

Mapping => type:product, child-mapping: reviews → child-mapping: review → child-field: reviewerName, check Only Index Specified field

Analyzer: standard

Query

*curl -XPOST -H "Content-Type: application/json" http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query -d '{"query": {"**regexp**": "jo[e|n]", "field": "reviews.review.reviewerName"}}' | jq*

LAB: Basic Queries - Wildcard

Wildcard Query

*Wildcards → * (0 or more characters), ? (0 or 1 character)*

Find all products that have reviews from a person whose name is “alina something”.

Index

Mapping => type:product, child-mapping: reviews → child-mapping: review → child-field: reviewerName, check Only Index Specified field

Analyzer: standard

Query

curl -XPOST -H "Content-Type: application/json"

http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query

*--d '{"query": {"**wildcard**": "alina*", "field": "reviews.review.reviewerName"}}' | jq*

Full Text Search – Compound Queries

Conjunction Query -The conjunction query is a compound query. The result documents must satisfy all of the child queries.

Disjunction Query - The disjunction query is a compound query. The result documents must satisfy a configurable min number of child queries. By default this min is set to 1.

Boolean Query - The boolean query is a useful combination of conjunction and disjunction queries. A boolean query takes three lists of queries

- must - result documents must satisfy all of these queries.
- should - result documents should satisfy these queries.
- must not - result documents must not satisfy any of these queries.

Doc ID Query - The doc ID query returns the indexed document or documents among the specified set. This is typically used in conjunction queries to restrict the scope of other queries' output.

Query String Query - Also known as the String Query, the query string query allows humans to describe complex queries using a simple syntax.

Compound Queries: CONJUNCT/DISJUNCT

User Input...	Returns...

CONJUNCTION

```
conjuncts:
        match: "beautiful",
        free_breakfast : true
```

"beautiful" AND

"free_breakfast":true"

DISJUNCTION

```
disjuncts:
        match: "location",
        free_breakfast: true
```

"location" OR

"free_breakfast":true"

Compound Queries: Doc IDs, BOOLEAN

User Input...	Returns...
DOC ID docids: 9789814232, 6789814333	docids: 9789814232, 6789814333
BOOLEAN Must: conjuncts - "match": "fashion" Must_Not: disjuncts - "match": "trend"	"fashion industry" "fashion studio" ~~"fashion trend"~~

LAB: Compound Queries - Conjunction

Conjunction Query

Find all products that have the words “casual” and “shirt” in their description.

Index

Mapping => type:product, check Only Index Specified field,child-field:description

Analyzer: en

Advanced -> Default Analyzer : en

Query

curl -XPOST -H "Content-Type: application/json"
*http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query -d '{"query": {"**conjuncts**": [{"**match**": "casual", "field": "description"}, {"**match**": "shirt", "field": "description"}]}}' | jq*

LAB: Compound Queries - Disjunction

Disjunction Query

Find all products that has the words “fashionable” or “latest” in their description.

Index

Mapping => type:product, check Only Index Specified Field, child-field:description,

Analyzer: en

Advanced -> Default Analyzer : en

Query

curl -XPOST -H "Content-Type: application/json"
*http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query - -d ' {"query": {"**disjuncts**": [{"match": "fashionable", "field": "description"}, {"match": "latest", "field": "description"}]}} ' | jq*

LAB: Compound Queries - Boolean

Boolean Query

Find all products that have the word fashion but not the word trend in their description.

Index

Mapping => type:product, check Only Index Specified field, child-field:description

Analyzer: en

Advanced -> Default Analyzer : en

Query

curl -XPOST -H "Content-Type: application/json"

http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query -d '{"query": {"must": {"conjuncts": [{"match": "fashion", "field": "description"}]}, "must_not": {"disjuncts": [{"match": "trend", "field": "description"}]}}}' | jq

LAB: Compound Queries - Doc ID

Doc ID Query

Fetch specific Docs if available.

Index

Use any of the previous Index

Query

curl -XPOST -H "Content-Type: application/json"

*http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query - -d '{"query": {"**ids**": ["9789814232", "9822490682"]}}' | jq*

Full Text Search – Query String Query

Match - A term without any other syntax is interpreted as a match query for the term in the default field. The default field is _all unless overridden in the index mapping. Example: water

Match Phrases - Placing the search terms in quotes performs a match phrase query. Example: "light beer"

Field Scoping - You can qualify the field for these searches by prefixing them with the name of the field separated by a colon. Example: description:water

Required, Optional, and Exclusion - When your query string includes multiple items, by default these are placed into the SHOULD clause of a *Boolean Query*. You can change this by prefixing your items with a '+'(MUST) or '-'(MUST NOT). Example: +description:water -light beer.

Boosting : You can influence the relative importance of the clauses by suffixing clauses with the ^ operator followed by a number. Example: description:water name:water^5.

Numeric Ranges :You can perform numeric ranges by using the >, >=, <, and <= operators, followed by a numeric value.

QUERY STRING QUERY - OPTIONS

User Input...	Returns...
FIELD SCOPING description:water	"water" *in description field*
REQUIRED, OPTIONAL, EXCLUSION, BOOLEAN QUERY +description:water –"light beer" Ie + MUST –MUST_NOT	description:water strong beer description:water random beer ~~description:water light beer~~
BOOSTING description:water name:water^5	description:water [*score: 10]* name: water [*score: 50 *i.e. boosted]*

LAB: Query String Query

Required, optional, exclusion (Prefix with +, -) → Boolean query

Find all products that have the word fashion but not the word trend in their description.

Index

Mapping => type:product, check Only Index Specified field, child-field:description

Analyzer: en

Advanced -> Default Analyzer : en

Query

curl -XPOST -H "Content-Type: application/json"

http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query

- -d '{"query": {"query": "+description:fashion -description:trend"}}' | jq

LAB: Query String Query - Boosting

Boosting (suffix with ^) → To bias scoring

Find all products that have the words fashion or trend in description. Boost the score if description has trend.

Index

Mapping => type:product, check Only Index Specified field, child-field: description

Analyzer: en

Advanced -> Default Analyzer : en

Query

curl -XPOST -H "Content-Type: application/json"

http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query

- -d '{"query": {"query": "description:fashion ***description:trend^5****"}}' | jq*

LAB: Query String Query – Numeric Ranges

Numeric ranges (>, >=, =, =<, <) → Numeric range queries

Find all products more expensive than $100

Index

Mapping => type: product, check Only Index Specified field, child-field: price

Query

curl -XPOST -H "Content-Type: application/json"

*http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query - -d '{"query": {"query": "**price: > 100**"}}' | jq*

Full Text Search – Range & Special Queries

Date Range Query - The date range query finds documents containing a date value in the specified field within the specified range. Define the endpoints using the fields start and end. The inclusiveStart and inclusiveEnd properties in the query JSON control whether or not the end points are included or excluded

Numeric Range Query - The numeric range query finds documents containing a numeric value in the specified field within the specified range. Define the endpoints using the fields min and max. The inclusiveMin and inclusiveMax properties control whether or not the end points are included or excluded.

Match All Query - Matches all documents in the index. Be aware that this query will match all documents that were indexed even if they have no terms in the index.

Match None Query- Matches no documents in the index.

Term Query - A term query is the simplest possible query. It performs an exact match in the index for the provided term. Most of the time users should use a Match Query instead. Does not perform any analysis.

Phrase Query - A phrase query searches for terms occurring in the specified position and offsets. The phrase query performs an exact term match for all the phrase constituents without using an analyzer

Range Queries: DATE and NUMERIC RANGE

User Input...	Returns...

DATE RANGE

```
"start": "2001-10-09T10:20:30-08:00",
"end": "2016-10-31",
"inclusive_start": false,
"inclusive_end": false,
"field": "review_date"
```

"2001-**10-10**T10:20:30-08:00"

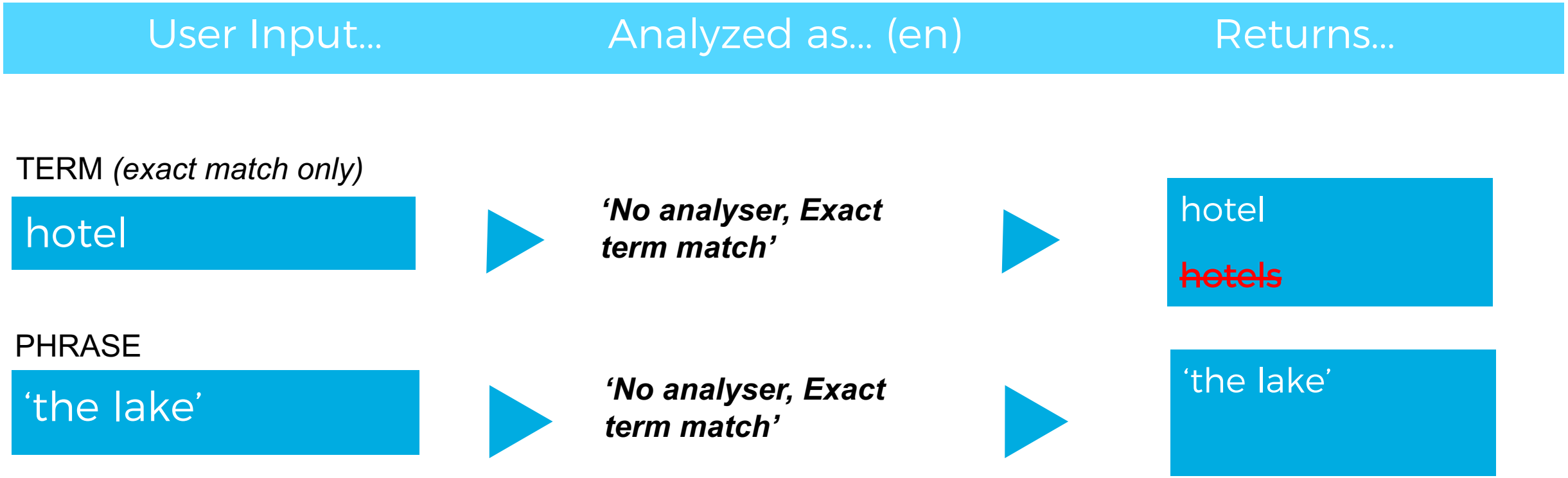
"2016-10-30"

NUMERIC RANGE

price <= 100 and price > 50

price = 55 or 65 or 89 or 100 etc.

Special Queries: TERM and PHRASE



Special Queries: MATCH ALL & MATCH NONE

User Input...	Returns...
MATCH ALL	
" "	All indexed docs
MATCH NONE	
" "	No indexed documents

These special queries are used in special cases with compound queries, facets, and other testing queries.

LAB: Range Queries - Numeric range

Numeric Range Query

Find all products that whose price is between 9 and 10.

Index

Mapping => type: product, check Only Index Specified field, child-field: price

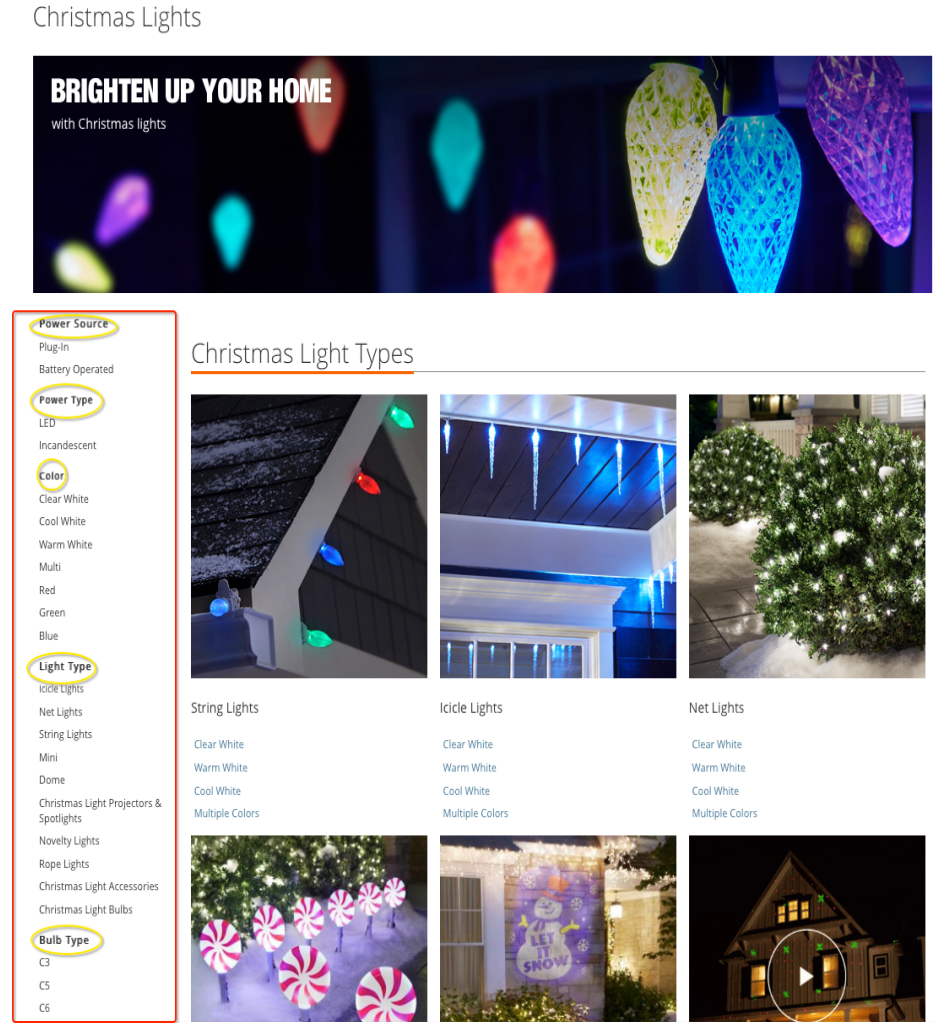
Query

curl -XPOST -H "Content-Type: application/json"

http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query -d '{"query": {"min": 9, "max": 10, "field": "price"}}' | jq

Faceted Search

- Faceted search is the dynamic clustering of search results into categories that lets users drill into search results by any value in any field (facet).
- Each facet displayed also shows the number of hits within the search that match that category.
- Users can then “drill down” by applying specific constraints to the search results.
- Faceted search is also called faceted browsing, faceted navigation, guided navigation and sometimes parametric search.



LAB: Term Facets

Term Facet – Example 1 – Simple Facet

Fetch all the words and the number of their occurrences in the categories field of all products that have the word nylon in the description field or cozy in either description or title fields.

Index

Mapping => type: product, check Only Index Specified field, child-field: title, child-field:description, child-field:categories

Analyzer: standard

Query

*curl -XPOST -H "Content-Type: application/json" http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query -d '{"query": {"disjuncts": [{"match": "nylon", "field": "description"}, {"match": "cozy"}]}, "**facets**": {"category_tokens": {"field": "categories", "size": 15}}}' | jq*

LAB: Term Facets

Term Facet – Example 2 – Nested field

Fetch top 10 reviewer ids (based on number of reviews) for all products that have the term gift in the description field.

Index

Mapping => type: product, check Only Index Specified field, child-field: title, child-field:description, child-mapping:reviews{} → child-mapping:review{} → child-field:reviewerID

Analyzer: standard

Query

*curl -XPOST -H "Content-Type: application/json" http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query -d '{"query": {"query": "+gift", "field": "description"}, "facets": {"reviewers": {"field": "**reviews.review.reviewerID**", "size": 10}}}' | jq*

LAB: Term Facets

Term Facet – Example 3 – Multiple Facets

Fetch the top 5 brands and the top 3 categories on all the of products that could be gifts.

Index

Mapping => type:product, check Only Index Specified field, child-field: description, child-field: brand, child-field: categories

Analyzer: standard

Query

curl -XPOST -H "Content-Type: application/json"

*http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query -d '{"query": {"query": "+gift ","field": "description"},"**facets**": {"**brands**": {"field": "brand","size": 5},"**categories**": {"field": "categories","size": 3}}}' | jq*

LAB: Numeric range Facets

Numeric Range Facet – Example 1 – Simple Numeric range facet

Fetch the count of the products that whose price is less than 15 and those whose price is greater than 15, that have the words fashion or pirate in the title field or the description field.

Index

Mapping => type: product, check Only Index Specified field, child-field:description, child-field:categories, child-field:price

Analyzer: Standard

Query

curl -XPOST -H "Content-Type: application/json"

*http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query -d '{"query": {"disjuncts": [{"match": "fashion"}, {"match": "pirate"}]}, "facets": {"type": {"field": "price", "size": 10, "**numeric_ranges**": [{"name": "<15", "**max**": 15}, {"name": ">15", "**min**": 15}]}}}' | jq*

LAB: Numeric range Facets

Numeric Range Facet – Example 2 – Nested field

Fetch number of outdoor shoe products whose sales rank is greater than 100000.

Index

Mapping => type: product, check Only Index Specified field, child-field: description, child-mapping: salesRank → child-field: Sports & Outdoors (searchable as Outdoors)

Analyzer: en

Advanced -> Default Analyzer : en

Query

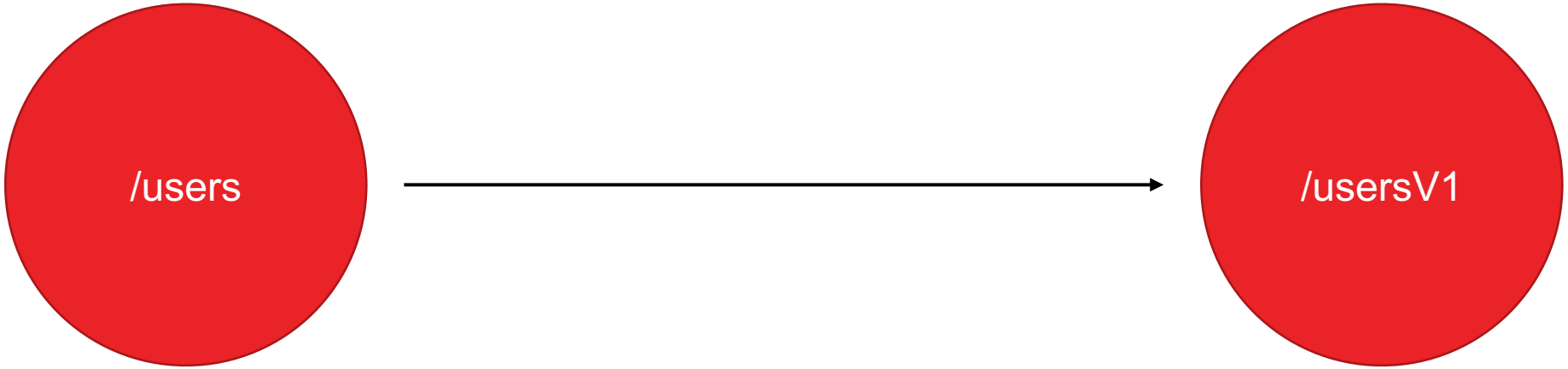
curl -XPOST -H "Content-Type: application/json"

http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query -d '{"query": {"disjuncts": [{"match": "shoe", "field": "description"}, {"match": "boot", "field": "description"}]}, "facets": {"type": {"field": "salesRank.Outdoors", "size": 5, "numeric_ranges": [{"name": "greater_than_100000", "min": 100000}]}}}' | jq

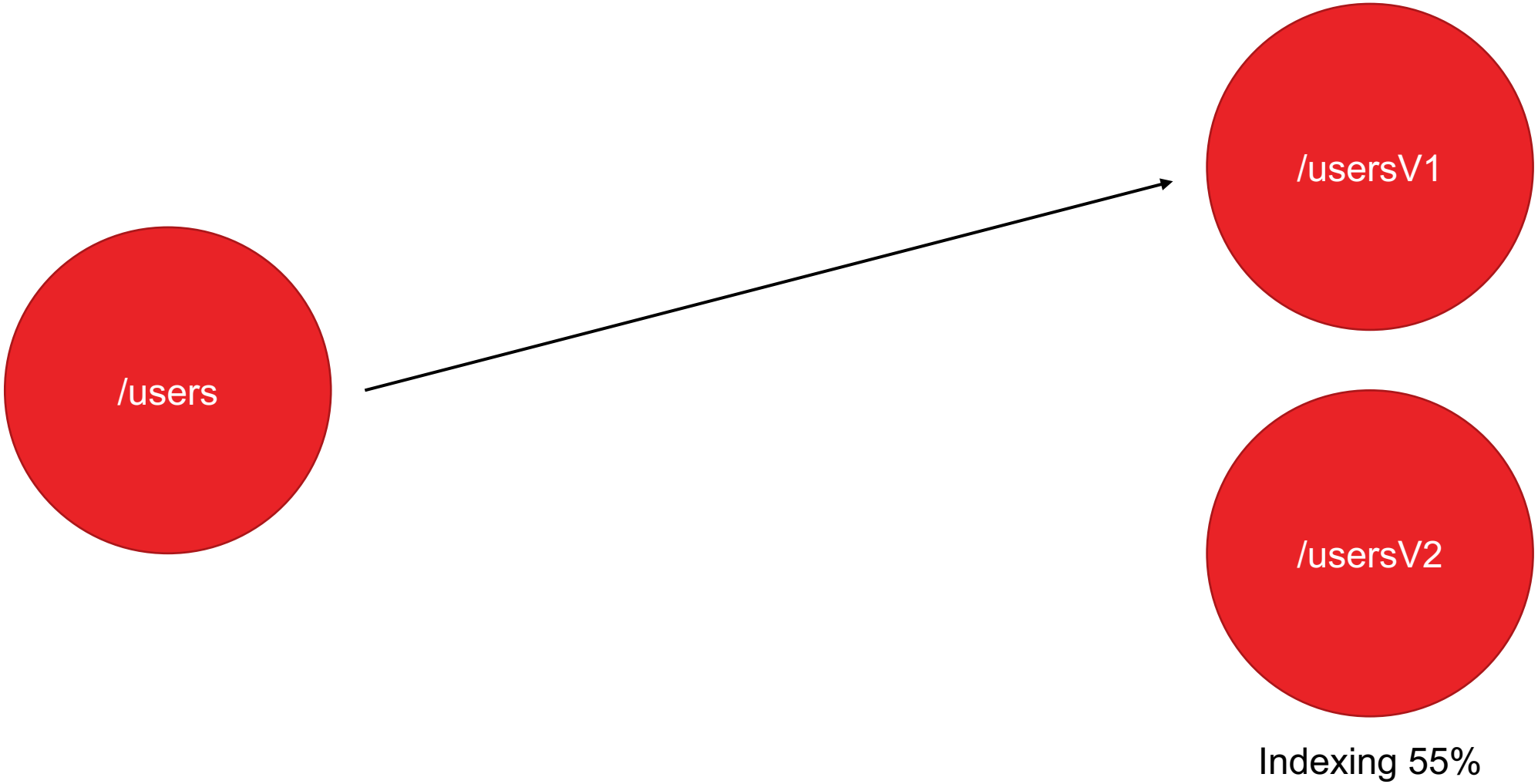
Full Text Search – Index Alias

- An index alias is a logical definition that points to actual full text indexes.
- Index alias allows a level of indirection between logical index name (used by your application) and the physical index name used in Couchbase so that applications can refer to a stable name(logical) while the actual index(Physical) is re-defined or moved.
- An alias switch will be instantaneous and the index will be ready to use.

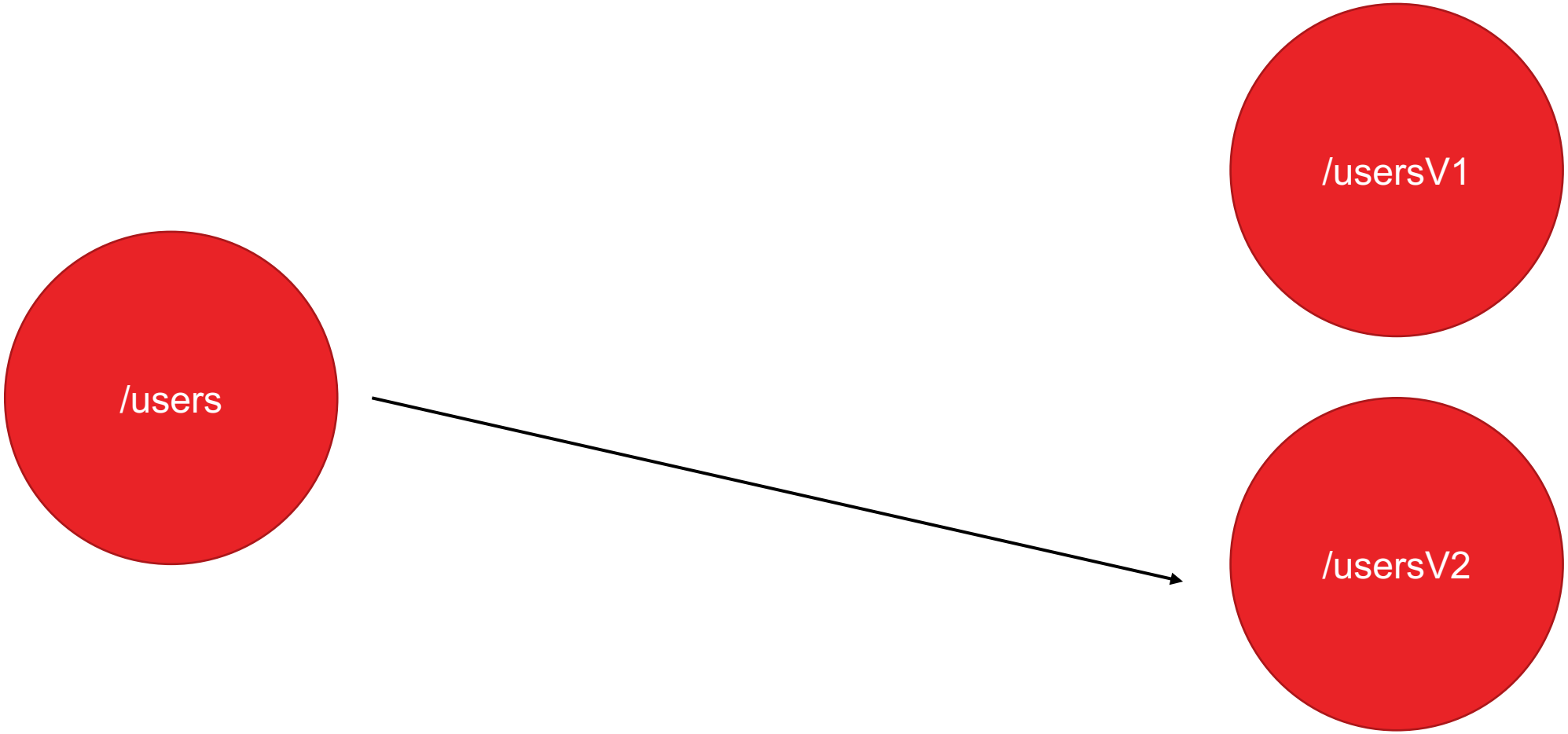
Index Aliases



Index Aliases

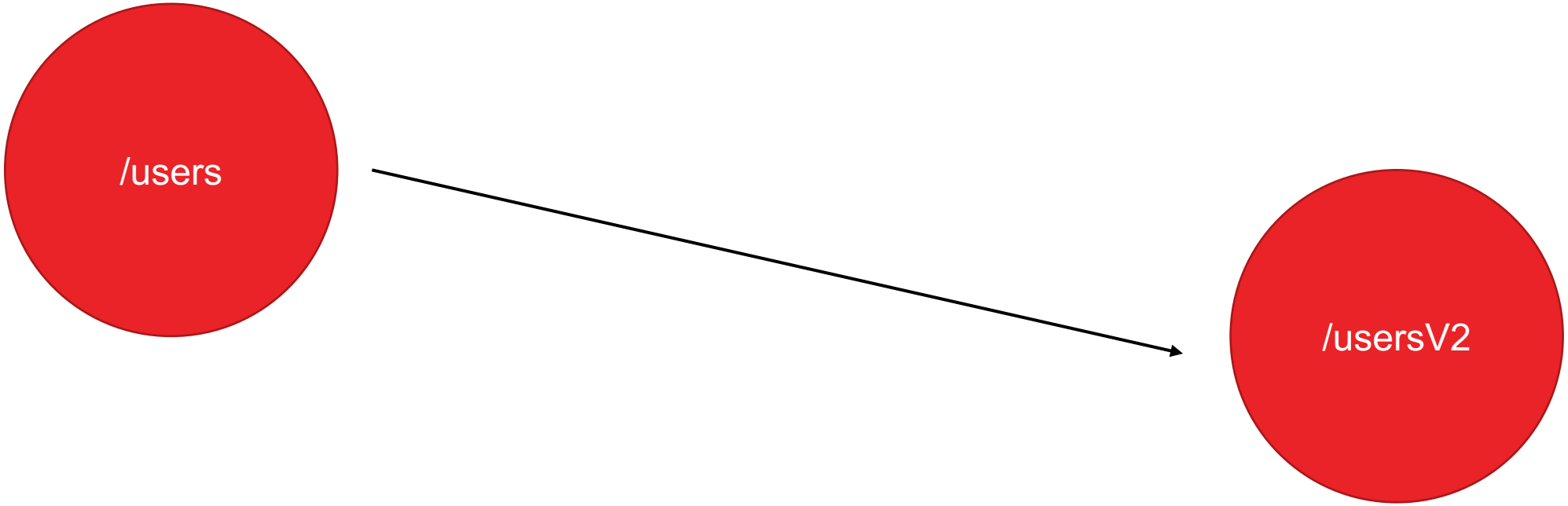


Index Aliases



Atomic Switch to /usersV2

Index Aliases



Atomic Switch to /usersV2

LAB:
Index Alias

- Add an Index Alias
- Create an index alias with alias name “users”
- Select Index created for match query as target index.

P.S Note : Multiple target index selection is allowed.

- Run the match query with alias name as index name
- *Edit the index Alias and select the index created for description as target index* and run its query with same index alias name
- Observe that the change in index is completely controlled at the database tier without any change to the application server.

LAB: Advanced Indexing

Stop Words – Custom Analyzer

Add “pocket” to the list of stop words in the title field.

Index

Custom Filters => +Add Word List => Name: remove_pocket, Words: pocket

Custom Filters => +Add Token Filter => Name: pocket-remover, Type: stop_tokens, Stop Words: remove_pocket

Analyzers => +Add Analyzer =>Name: custom, Token Filters: pocket-remover

Mapping => type: product, check Only Index Specified field,child-field: title, analyzer: custom

Query

curl -XPOST -H "Content-Type: application/json" http://Administrator:password@192.168.61.101:8094/api/index/<index-name>/query -d '{"query": {"query": "pocket"}}'

LAB: Queries – Challenge 1

Fetch products that are in the category “Keychains” and have “star wars” in either the title or description fields

LAB: Queries – Challenge 2

Find all products whose review contains phrase “great looking jacket”.

LAB: Queries – Challenge 3

Find the top 10 reviewers by number of products reviewed.

Resources

1. 5.0 Beta – Full Text Search Intro Webinar video - "FTS 5.0 Launch Overview"
2. Presentation archive: https://connect.couchbase.com/us/connect-new-york-2017
3. Query Syntax: "FTS Query Types" – including JSON samples
4. Blog posts: https://blog.couchbase.com/search_gcse/?q=fts
5. Documentation: http://bit.ly/2tHduXo & http://www.blevesearch.com/
6. Forums - https://forums.couchbase.com/c/couchbase-full-text-search