Full Text Search

Note

Further updates to the search docs will come with future 4.x releases.

Enumerations

enum `couchbase.search.SearchScanConsistency`*(value)*

SearchScanConsistency

This can be:

NOT_BOUNDED

Which means we just return what is currently in the indexes.

Valid values are as follows:

`NOT_BOUNDED`= *<SearchScanConsistency.NOT_BOUNDED: 'not_bounded'>*

`REQUEST_PLUS`= *<SearchScanConsistency.REQUEST_PLUS: 'request_plus'>*

`AT_PLUS`= *<SearchScanConsistency.AT_PLUS: 'at_plus'>*

Options

class **couchbase.options.SearchOptions**(*timeout=None, limit=None, skip=None, explain=None, fields=None, highlight_style=None, highlight_fields=None, scan_consistency=None, consistent_with=None, facets=None, raw=None, sort=None, disable_scoring=None, scope_name=None, collections=None, include_locations=None, client_context_id=None, serializer=None, show_request=None, log_request=None, log_response=None*)

Available options to for a search (FTS) query.

Warning

Importing options from `couchbase.search` is deprecated. All options should be imported from `couchbase.options`.

Parameters:

- **timeout** (*timedelta*, *optional*) – The timeout for this operation. Defaults to global search query operation timeout.
- **limit** (*int*, *optional*) – Specifies the limit to the number of results that should be returned. Defaults to None.
- **skip** (*int*, *optional*) – Specifies the number of results to skip from the index before returning results. Defaults to None.
- **explain** (*bool*, *optional*) – Configures whether the result should contain the execution plan for the search query. Defaults to False.
- **fields** (*List[str]*, *optional*) – Specifies the list of fields which should be searched. Defaults to None.
- **highlight_style** (`HighlightStyle`, optional) – Specifies the mode used for highlighting. Defaults to None.
- **highlight_fields** (*List[str]*, *optional*) – Specifies the list of fields that should be highlighted. Defaults to None.
- **scan_consistency** (`SearchScanConsistency`, optional) – Specifies the consistency requirements when executing the search query.
- **facets** (Dict[str, `Facet`], optional) – Specifies any facets that should be included in the search query. Defaults to None.
- **client_context_id** (*str*, *optional*) – The returned client context id for this query. Defaults to None.
- **disable_scoring** (*bool*, *optional*) – Specifies that scoring should be disabled. This improves performance but makes it impossible to sort based on how well a particular result scored. Defaults to False.
- **include_locations** (*bool optional*) – If set to True, will include the locations in the search result. Defaults to False.
- **sort** (Union[List[str],List[`Sort`]], optional) – Specifies a list of fields or search `Sort`'s to use when sorting the result sets. Defaults to None.

- **scope_name** (*string*, *optional*) – Specifies the scope which should be searched as part of the search query. Defaults to None.
- **collections** (*List[str]*, *optional*) – Specifies the collections which should be searched as part of the search query. Defaults to None.
- **consistent_with** (`MutationState`, optional) – Specifies a `MutationState` which the search query should be consistent with. Defaults to None.
- **serializer** (`Serializer`, optional) – Specifies an explicit serializer to use for this specific search query. Defaults to `DefaultJsonSerializer`.
- **raw** (*Dict[str, Any]*, *optional*) – Specifies any additional parameters which should be passed to the search query engine when executing the search query. Defaults to None.
- **show_request** (*bool*, *optional*) – Specifies if the search response should contain the request for the search query. Defaults to False.
- **log_request** (*bool*, *optional*) – **UNCOMMITTED** Specifies if search request body should appear the log. Defaults to False.
- **log_response** (*bool*, *optional*) – **UNCOMMITTED** Specifies if search response should appear in the log. Defaults to False.

SearchRequest

class **`couchbase.search.SearchRequest`**(*query*)

Represents a search query and/or vector search to execute via the Couchbase Full Text Search (FTS) service.

Parameters: **query** (Union[`SearchQuery`, `VectorSearch`]) – A `SearchQuery` or `VectorSearch` to initialize the search request.

Raises: InvalidArgumentException – If neither a `SearchQuery` or `VectorSearch` is provided.

Returns: The created search request.

Return type: `SearchRequest`

classmethod **`create`**(*query*)→ SearchRequest

Creates a `SearchRequest`.

Parameters: **query** (Union[`SearchQuery`, `VectorSearch`]) – A `SearchQuery` or `VectorSearch` to initialize the search request.

Raises: InvalidArgumentException – If neither a `SearchQuery` or `VectorSearch` is provided.

Returns: The created search request.

Return type: `SearchRequest`

property `search_query`: *SearchQuery | None*

Returns the search request's `SearchQuery`, if it exists.

Type: Optional[`SearchQuery`]

property `vector_search`: *VectorSearch | None*

Returns the search request's `VectorSearch`, if it exists.

Type: Optional[`VectorSearch`]

`with_search_query`(*query*)→ SearchRequest

Add a `SearchQuery` to the search request.

Parameters: **query** (`SearchQuery`) – The `SearchQuery` to add to the search request.

Raises:

- InvalidArgumentException – If the search request already contains a `SearchQuery`.
- InvalidArgumentException – If the provided query is not an instance of a `SearchQuery`.

Returns: The search request in order to allow method chaining.

Return type: `SearchRequest`

`with_vector_search`(*vector_search*)→ SearchRequest

Add a `VectorSearch` to the search request.

Parameters: **vector_search** (`VectorSearch`) – The `VectorSearch` to add to the search request.

Raises:

- InvalidArgumentException – If the search request already contains a `VectorSearch`.
- InvalidArgumentException – If the provided query is not an instance of a `VectorSearch`.

Returns: The search request in order to allow method chaining.

Return type: `SearchRequest`

Results

SearchMetaData

class `couchbase.search.SearchMetaData`(*raw*)

Represents the meta-data returned along with a search query result.

SearchMetrics

class `couchbase.search.SearchMetrics`(*raw*)

SearchResult

class `couchbase.result.SearchResult`

`rows`()

The rows which have been returned by the search query.

Note

If using the *acouchbase* API be sure to use `async for` when looping over rows.

Returns:	Either an iterable or async iterable.
Return type:	Iterable

`metadata`()

The meta-data which has been returned by the search query.

Returns:	An instance of `SearchMetaData`.
Return type:	`SearchMetaData`

Vector Search

class `couchbase.vector_search.VectorQuery`(*field_name, vector, num_candidates=None, boost=None*)

Represents a vector query.

Parameters:

- **field_name** (*str*) – The name of the field in the search index that stores the vector.
- **vector** (*Union[List[float], str]*) – The vector to use in the query.
- **num_candidates** (*int, optional*) – Specifies the number of results returned. If provided, must be greater or equal to 1.
- **boost** (*float, optional*) – Add boost to query.

Raises:

- InvalidArgumentException – If the vector is not provided.
- InvalidArgumentException – If the vector is not a list or str.
- InvalidArgumentException – If vector is a list and all values of the provided vector are not instances of float.

Returns: The created vector query.

Return type: `VectorQuery`

property **`boost`**: *float | None*

Returns vector query's boost value, if it exists.

Type: Optional[float]

classmethod **`create`**(*field_name*, *vector*, *num_candidates=None*, *boost=None*)→ VectorQuery

Creates a `VectorQuery`.

Parameters:
- **field_name** (*str*) – The name of the field in the search index that stores the vector.
- **vector** (*Union[List[float], str]*) – The vector to use in the query.
- **num_candidates** (*int, optional*) – Specifies the number of results returned. If provided, must be greater or equal to 1.
- **boost** (*float, optional*) – Add boost to query.

Raises:
- InvalidArgumentException – If the vector is not provided.
- InvalidArgumentException – If the vector is not a list or str.
- InvalidArgumentException – If vector is a list and all values of the provided vector are not instances of float.

Returns: The created vector query.

Return type: `VectorQuery`

property **`field_name`**: *str*

Returns vector query's field name

Type: str

property **`num_candidates`**: *int | None*

Returns vector query's num candidates value, if it exists.

Type: Optional[int]

property **`vector`**: *List[float] | None*

Returns the vector query's vector.

Type: Optional[List[float]]

property **`vector_base64`**: *str | None*

Returns the vector query's base64 vector str.

Type: Optional[str]

class **`couchbase.vector_search.VectorSearch`**(*queries*, *options=None*)

Represents a vector search.

Parameters:

- **queries** (List[`VectorQuery`]) – The list of `VectorQuery`'s to use for the vector search.
- **options** (`VectorSearchOptions`, optional) – Options to set for the vector search.

Raises:

- InvalidArgumentException – If a list of `VectorQuery` is not provided.
- InvalidArgumentException – If all values of the provided queries are not instances of `VectorQuery`.

Returns: The created vector search.

Return type: `VectorSearch`

classmethod **`from_vector_query`**(*query*)→ VectorSearch

Creates a `VectorSearch` from a single `VectorQuery`.

Parameters: **query** (`VectorQuery`) – A `VectorQuery`'s to use for the vector search.

Raises: InvalidArgumentException – If the provided query is not an instance of `VectorQuery`.

Returns: The created vector search.

Return type: `VectorSearch`

property **`options`**: *VectorSearchOptions | None*

INTERNAL

property **`queries`**: *List[VectorQuery]*

INTERNAL

Enumerations

enum **`couchbase.vector_search.VectorQueryCombination`**(*value*)

Specifies how multiple vector searches are combined.

This can be one of:

AND: Indicates that multiple vector queries should be combined with logical AND.

OR: Indicates that multiple vector queries should be combined with logical OR.

Valid values are as follows:

AND= *<VectorQueryCombination.AND: 'and'>*

OR= *<VectorQueryCombination.OR: 'or'>*

Options

class `couchbase.options.VectorSearchOptions`*(vector_query_combination=None)*

Available options to for a FTS vector search.

Parameters: **vector_query_combination** (`VectorQueryCombination`, optional) – Specifies logical operation to use with multiple vector queries.