RediSearch 2.4 release notes

Requirements
RediSearch v2.4.8 requires:

- Minimum Redis compatibility version (database): 6.0.0
- Minimum Redis Enterprise Software version (cluster): 6.0.0

v2.4.8 (May 2022)
This is a maintenance release for RediSearch 2.4.
Update urgency: MODERATE: Program an upgrade of the server, but it's not urgent. However, if you're using Vector Similarity (introduced in RediSearch 2.4), there are some critical bugs that may affect a subset of users. In this case, you should upgrade.

Details:

- Bug fixes:
  - #2739 Memory leak in coordinator related to Vector Similarity (MOD-3023)
  - #2736, #2782 Memory allocation restrictions for Vector Similarity indices (causing OOM) (MOD-3195)
  - #2755 Compare the entire vector field name instead of a prefix when creating a new vector index
  - #2780 Initialize all variables in EvalContext (which might have led to crashes in clustered databases)

- Improvements:
  - #2740 Performance optimization for hybrid vector queries

v2.4.6 (May 2022)
This is a maintenance release for RediSearch 2.4.
Update urgency: MODERATE: Program an upgrade of the server, but it's not urgent.

Details:

- Bug fixes:
  - #2716 Removed assert statement that could cause crashes with replica of (MOD-3008, MOD-3012)
  - #2734 ON_TIMEOUT RETURN policy fix: return results obtained until timeout rather than discarding them
  - #2714 Memory leak on non-TLS setup in coordinator

v2.4.5 (April 2022)
This is a maintenance release for RediSearch 2.4.
Update urgency: MODERATE: Program an upgrade of the server, but it's not urgent.

Details:

- Bug fixes:
  - #2702 INKEYS combined with Vector Similarity caused server unresponsiveness (MOD-2952)
  - #2705 Incorrect results when deleting a document that was skipped at index time
**Synonyms in Chinese**

**Improvements:**
- #2694 Performance: In a TEXT field, skip term iterator if term does not appear in requested field

### v2.4.3 (March 2022)

This is the General Availability release of RediSearch 2.4.

**Headlines**

RediSearch 2.4 introduces a new capability, Vector Similarity Search (VSS), which allows indexing and querying vector data stored (as BLOBs) in Redis hashes.

It also introduces a new query syntax to address query parser inconsistencies found in previous versions of RediSearch. Users can now choose between Dialect version 1 (to keep existing query parser behavior) or Dialect version 2 (to switch to the updated behavior).

All VSS queries or any query using the `PARAMS` option must use Dialect version 2.

**What's new in 2.4**

- **`FT.CREATE`** is extended to support the creation of 2 popular types of vector indexes:
  - **Flat Index**
    - This type of index is used when the recall is more important than the speed of query execution. A query vector will be compared to all vectors in a flat index. The search results will return the exact top k nearest neighbors to the query vector.
  - **Hierarchical Navigable Small World (HNSW) Index**
    - This index is a modified implementation of the library written by the author of this influential academic paper. An HNSW vector index is used when the speed of query execution is preferred over recall. The results returned are approximate nearest neighbors (ANNs).
    
    You can try out different HNSW index parameters (M, CONSTRUCTION, EFRUNTIME) to improve the "recall versus speed" balance.

- *Use `FT.SEARCH` to retrieve the top K hashes with the most similar vectors to a given query vector.*

- Hybrid queries in `FT.SEARCH`:
  - Use hybrid queries to retrieve Redis hashes that match a combination of vector and non-vector search criteria. Non-vector search criteria can include expressions combining NUMERIC, TEXT, TAG, and GEO fields.
  
  Hybrid queries are often used in modern ecommerce search applications featuring "visual" similarity plus metadata similarity. For example, you can use a single hybrid query to find products that are visually similar to a given image within a price range and/or geo-location.

- *Use `FT.CONFIG` to set `DEFAULT_DIALECT` at the module level. By default, `DEFAULT_DIALECT` is set to 1.*

- **Override DIALECT:**
  - It is possible to override the module-level dialect for a specific command at runtime. You can specify the dialect when executing any of the following commands:
    - `FT.SEARCH`
    - `FT.AGGREGATE`
    - `FT.EXPLAIN`
    - `FT.EXPLAINCLI`
• **FT SPELLCHECK**

If you do not specify dialect when running any of these commands, they will use the default module-level dialect value.

### Details

- **Features:**
  - [#2671](#) Add Dialect support
- **Performance enhancements (since 2.4-RC1):**
  - [#2647](#) Normalize vector once for ad-hoc flat search
  - [#2638](#) Optimized hybrid query when no scores are required
  - [#2653](#) Updating specific field load optimization rule
  - [#2670](#) Use REDISMODULE EVENT SHUTDOWN to clear resources
- **Security and privacy (since 2.4-RC1):**
  - [#2584](#) Fix MOD-2086, added support for TLS passphrase
- **Bug fixes (since 2.4-RC1):**
  - [#2651](#) Fix client freeze on docs expire during query
  - [#2641](#) Memory leak in the coordinator
  - [#2645](#) Ignore NULL value on ingest
  - [#2654](#) VecSim hybrid query - return empty iterator for invalid intersection child iterator

### Introducing DIALECT

RediSearch 2.4.3 introduces a new query syntax to address query parser inconsistencies found in previous versions of RediSearch. Users can now choose between:

- Dialect version 1 (to keep the query dialect as in RediSearch 2.2)
- Dialect version 2 (to use the updated dialect)

Existing RediSearch 2.2 users will not have to modify their queries since the default dialect is 1. However, all RediSearch users should gradually update their queries to use dialect version 2.

### Background

Under certain conditions, some query parsing rules did not behave as originally intended. Queries containing the following operators could return unexpected results:

- **AND**
- quotes, ~, -, and % (exact, optional, negation, fuzzy)
- **OR**

To minimize the impact on existing, unaffected RediSearch users, a DIALECT setting was introduced to allow:

- Existing queries to run without any modification (DIALECT 1)
- New queries to benefit from the updated query-parsing behavior (DIALECT 2)

### Examples of impacted queries

Your existing queries may behave differently under different DIALECT versions, if they fall into any of the following categories:

- Your query has a field modifier followed by multiple words.
Consider this simple query @name:James Brown.

The field modifier @name is followed by 2 words: James and Brown.

With DIALECT 1, the parser interprets this query as “find James Brown in the @name field.”

With DIALECT 2, the parser interprets it as “find James in the @name field AND Brown in ANY text field.” In other words, the query parser interprets it as (@name:James) Brown.

With DIALECT 2, you could achieve the default behavior from DIALECT 1 by updating your query to @name:(James Brown).

- Your query uses quotes, -, % (exact, optional, negation, fuzzy).

Consider a simple query with negation -hello world.

With DIALECT 1, the parser interprets this query as “find values in any field that does not contain hello AND does not contain world.” This is the equivalent of -(hello world) or -hello -world.

With DIALECT 2, the parser interprets it as -hello AND world, so only hello is negated.

With DIALECT 2, you could achieve the default behavior from dialect 1 by updating your query to -(hello world).

Another example that illustrates the differences in parser behavior is hello world | "goodbye" moon:

- With DIALECT 1, the parser interprets this query as searching for (hello world | "goodbye") moon
- With DIALECT 2, the parser interprets it as searching for either hello world OR "goodbye" moon.

Note: This is the first GA version of 2.4. The version inside Redis will be 2.4.3 in semantic versioning. Since the version of a module in Redis is numeric, we could not add a GA flag.

Updated: May 26, 2022