Azure Search - frequently asked questions (FAQ)

08/02/2017 • 5 minutes to read • Contributors

In this article

Find answers to commonly asked questions about concepts, code, and scenarios related to Azure Search.

Platform

How is Azure Search different from full text search in my DBMS?

Azure Search supports multiple data sources, linguistic analysis for many languages, custom analysis for interesting and unusual data inputs, search rank controls through scoring profiles, and user-experience features such as typeahead, hit highlighting, and faceted navigation. It also includes other features, such as synonyms and rich query syntax, but those are generally not differentiating features.

What is the difference between Azure Search and Elasticsearch?

When comparing search technologies, customers frequently ask for specifics on how Azure Search compares with Elasticsearch. Customers who choose Azure Search over Elasticsearch for their search application projects typically do so because we've made a key task easier or they need the built-in integration with other Microsoft technologies:

- Azure Search is a fully-managed cloud service with 99.9% service level agreements (SLA) when provisioned with sufficient redundancy (2 replicas for read access, 3 replicas for read-write).

- Microsoft's Natural language processors offer leading edge linguistic analysis.
- Azure Search indexers can crawl a variety of Azure data sources for initial and incremental indexing.
- If you need rapid response to fluctuations in query or indexing volumes, you can use slider controls in the Azure portal, or run a PowerShell script, bypassing shard management directly.
- Scoring and tuning features provide the means for influencing search rank scores beyond what the search engine alone can provide.

Can I pause Azure Search service and stop billing?

You cannot pause the service. Computational and storage resources are allocated for your exclusive use when the service is created. It's not possible to release and reclaim those resources on-demand.

Indexing Operations

Backup and restore (or download and move) indexes or index snapshots?

Although you can get an index definition at any time, there is no index extraction, snapshot, or backup-restore feature for downloading a *populated* index running in the cloud to a local system, or moving it to another Azure Search service.

Indexes are built and populated from code that you write, and run only on Azure Search in the cloud. Typically, customers who want to move an index to another service do so by editing their code to use a new endpoint, and then rerun indexing. If you want the ability to take a snapshot or backup an index, cast a vote on User Voice.

Can I restore my index or service once it is deleted?

No, you cannot restore indexes or services. If you delete an Azure Search index, the operation is final and the index cannot be recovered. When you delete an Azure Search service, all indexes in the service are deleted permanently. Also, if you delete an Azure resource group that contains one or more Azure Search services, all services are deleted permanently.

Restoring resources such as indexes, indexers, data sources, and skillsets requires that you recreate them from code. In the case of indexes, you must reindex data from external sources. For this reason, it is strongly recommended that you retain a master copy or backup of the original data in another data store, such as Azure SQL Database or Cosmos DB.

Can I index from SQL database replicas (Applies to Azure SQL Database indexers)

There are no restrictions on the use of primary or secondary replicas as a data source when building an index from scratch. However, refreshing an index with incremental updates (based on changed records) requires the primary replica. This requirement comes from SQL Database, which guarantees change tracking on primary replicas only. If you try using secondary replicas for an index refresh workload, there is no guarantee you get all of the data.

Search Operations

Can I search across multiple indexes?

No, this operation is not supported. Search is always scoped to a single index.

Can I restrict search index access by user identity?

You can implement security filters with `search.in()` filter. The filter composes well with identity management services like Azure Active Directory(AAD) to trim search results based on defined user group membership.

Why are there zero matches on terms I know to be valid?

The most common case is not knowing that each query type supports different search behaviors and levels of linguistic analyses. Full text search, which is the predominant workload, includes a language analysis phase that breaks terms down to root forms. This aspect of query parsing casts a broader net over possible matches, because the tokenized term matches a greater number of variants.

Wildcard, fuzzy and regex queries, however, aren't analyzed like regular term or phrase queries and can lead to poor recall if the query does not match the analyzed form of the word in the search index. For more information on query parsing and analysis, please see query architecture.

My wildcard searches are slow.

Most wildcard search queries, like prefix, fuzzy and regex, are rewritten internally with matching terms in the search index. This extra processing of scanning the search index adds to latency. Further, broad search queries, like `a*` for example, that are likely to be rewritten with many terms can be very slow. For performant wildcard searches, consider defining a custom analyzer.

Why is the search rank a constant or equal score of 1.0 for every hit?

By default, search results are scored based on the statistical properties of matching terms, and ordered high to low in the result set. However, some query types (wildcard, prefix, regex) always contribute a constant score to the overall document score. This behavior is by design. Azure Search imposes a constant score to allow matches found through query expansion to be included in the results, without affecting the ranking.

For example, suppose an input of "tour*" in a wildcard search produces matches on “tours”, “tourettes”, and “tourmaline”. Given the nature of these results, there is no way to reasonably infer which terms are more valuable than others. For this reason, we ignore term frequencies when scoring results in queries of types wildcard, prefix, and regex. Search results based on a partial input are given a constant score to avoid bias towards potentially unexpected matches.

Design patterns

What is the best approach for implementing localized search?

Most customers choose dedicated fields over a collection when it comes to supporting different locales (languages) in the same index. Locale-specific fields make it possible to assign an appropriate analyzer. For example, assigning the Microsoft French Analyzer to a field containing French strings. It also simplifies filtering. If you know a query is initiated on a fr-fr page, you could limit search results to this field. Or, create a scoring profile to give the field more relative weight. Azure Search supports over 50 language analyzers to choose from.

Next steps

Is your question about a missing feature or functionality? Request the feature on the User Voice web site.

See also

StackOverflow: Azure Search

How full text search works in Azure Search

What is Azure Search?