

Search index design guide (PDF)

The Azure AI Search and pgvector index shapes SPSync builds from SharePoint lists, what text gets embedded, and how to bound who can query them.

Support staff want a search box over their Knowledge Articles list that finds "VPN drops on hotel wifi" when the article is titled "Remote access disconnects". SharePoint search handles that poorly, and an agent grounded on the list needs the same thing. Before anyone builds the index, decide which columns are searchable, which are filters, what text becomes a vector and who may query it.

SPSync builds two kinds of index from a list. On Azure AI Search it creates and maintains the indexes. On Postgres with pgvector it adds embedding columns to the replicated tables. Both re-embed only rows whose text changed.

Azure AI Search index layout

Each replicated table becomes its own index. The service does not allow _ in index names, so each one becomes -. A kb_articles list gives:

Index | Holds | Document key

kb-articles | One document per list item | sp_key, the item id as a string

kb-articles--audiences | One document per value of a multi-choice Audiences column | sp_key, built from the item id and ordinal

sp-users | One document per person or group | sp_key, the user id

The synthetic sp_key exists because the service needs a string key and the item id is a number. Field attributes follow the column type, and you do not configure them:

Column type | Searchable | Filterable | Facetable

Title, single-line text, choice, lookup display value | Yes | Yes | Yes

Multi-line text | Yes | No | No

Hyperlink URL and description | Yes | Yes | No

Whole numbers, dates, Yes/No | No | Yes | Yes

Decimal numbers and currency | No | Yes | No

Ids | No | Yes | No

Multi-line text is searchable only because a long value can exceed the service's size limit for a filterable term. When an item is deleted in SharePoint, its document is removed from the index on the next run, along with its documents in the child indexes. If a column changes type, the service cannot alter the field in place, so SPSync starts a new kb-articles-v2 index and writes there from then on.

SPSync sets up one HNSW vector profile with the service defaults. A semantic ranker or scoring profile you add in the portal is left alone, because SPSync only updates the field list. SPSync authenticates with an admin API key; it does not sign in to the search service with Entra ID.

pgvector tables and embedding columns

On pgvector there is no separate index to design. The list is replicated into Postgres tables as usual, and SPSync adds a vector column to each table that carries a column you chose to embed:

Column or table | Holds

sp_embedding | The vector, on the list's own table by default, or in a side table if you prefer

sp_embedding_hash | A hash of the source text and model name, used to skip unchanged rows

sp_embedding_meta | One row per embedded column, recording the model and dimensions used

An HNSW index over the vector column is kept current. Because the vector sits beside the relational columns, one SQL statement can filter on status or owner and order by distance. A deleted item keeps its row and its vector with sp_deleted_at set, so add WHERE sp_deleted_at IS NULL to every similarity query. This destination needs PostgreSQL 15 or later.

What text gets embedded

On both destinations you name the columns, and the text for a row is their values joined in the order you list them. Providers are OpenAI, Azure OpenAI, or a local model that runs on the sync host with no API key.

Setting | Effect

Columns to embed | Their values, concatenated in order, are the input text

Model | Changing it re-embeds every row, in place on both destinations

Dimensions | Changing them writes to sp_embedding_v2 on pgvector or a new index on Azure AI Search

Put the title first, because it carries the most meaning per word, and leave out codes and ids. They add noise to the vector and work better as filters.

Long text and chunking

SPSync builds one vector per row. That works for short text such as an article summary. A multi-line column holding pages of meeting minutes becomes one vector for the whole passage, which retrieves poorly and can exceed a provider's input limit. Splitting long text into passages is not shipped, though it is on our list of possible future work. Until then, embed the short columns and let your own pipeline split long ones by reading the replica.

Who can query the index

SPSync does not copy SharePoint item permissions. Every Azure AI Search item index carries a filterable permitted_groups field so the standard trimming filter has somewhere to go, but SPSync

leaves it empty today. Anyone holding a query key can search every document.

Bound access outside the index. On Azure AI Search, query keys belong to the whole service, so give them only to applications whose users may see every list in it, and put sensitive lists on a separate search service. On pgvector, use a database role per application and row-level security on the replicated tables. Neither replaces item permissions, so choose the lists with that in mind.

Stale embeddings: keeping a SharePoint vector index current covers the change detection in more depth. Search over SharePoint lists is about the search page built on top.

Full, up to date version: <https://spsync.io/resources/search-index-design/>