LanceDB

Work with data › **Metadata Filtering**

Work with data

Metadata Filtering

We support rich filtering features of query results based on metadata fields. While joint vector and metadata search at scale presents a significant challenge, LanceDB achieves sub-100ms latency at thousands of QPS, enabling efficient vector search with filtering capabilities even on datasets containing billions of records.

By default, *pre-filtering* is applied before executing the vector search to narrow the search space within large datasets, thereby reducing query latency. Alternatively, *post-filtering* can be employed to refine results after the vector search completes.

Best Practices for Filtering

Scalar Indices: We strongly recommend creating scalar indices on columns used for filtering, whether combined with a search operation or applied independently (e.g., for updates or deletions).

SQL Filters: Built on DataFusion, LanceDB supports SQL filters. For example: for a column of type LIST(T) can use `LABEL_LIST` to create a **scalar index**. Then leverage DataFusion's **array functions** like `array_has_any` or `array_has_all` for optimized filtering.

Python Typescript

```python
import lancedb

# connect to LanceDB
db = lancedb.connect(
  uri="db://your-project-slug",
  api_key="your-api-key",
  region="us-east-1"
)
# Create sample data
data = [
    {"vector": [3.1, 4.1], "item": "foo", "price": 10.0},
    {"vector": [5.9, 26.5], "item": "bar", "price": 20.0},
    {"vector": [10.2, 100.8], "item": "baz", "price": 30.0},
    {"vector": [1.4, 9.5], "item": "fred", "price": 40.0},
]
table = db.create_table("metadata_filter_example", data=data, mode="overwrite")

#  Filters with vector search
filtered_result = (
    table.search([100, 102])
    .where("(item IN ('foo', 'bar')) AND (price > 9.0)")
    .limit(3)
    .to_arrow()
)

# With post-filtering
post_filtered_result = (
    table.search([100, 102])
    .where("(item IN ('foo', 'bar')) AND (price > 9.0)", prefilter=False)
    .limit(3)
    .to_arrow()
)

# Filters without vector search
filtered_no_search_result = (
    table.search()
    .where("(item IN ('foo', 'bar', 'baz')) AND (price > 9.0)")
    .limit(3)
    .to_arrow()
)
```

⚠ When querying large tables, omitting a `limit` clause may:

Overwhelm Resources: return excessive data.

Increase Costs: query pricing scales with data scanned and returned (**LanceDB Cloud pricing**).

SQL filters

Because it's built on top of DataFusion, LanceDB embraces the utilization of standard SQL expressions as predicates for filtering operations. SQL can be used during vector search, update, and deletion operations.

LanceDB supports a growing list of SQL expressions:

- `>, >=, <, <=, =`
- AND, OR, NOT
- IS NULL, IS NOT NULL
- IS TRUE, IS NOT TRUE, IS FALSE, IS NOT FALSE
- IN
- LIKE, NOT LIKE
- CAST
- regexp_match(column, pattern)
- **DataFusion Functions**

If your column name contains special characters, upper-case characters, or is a SQL Keyword, you can use backtick (`) to escape it. For nested fields, each segment of the path must be wrapped in backticks.

SQL

```
`CUBE` = 10 AND `UpperCaseName` = '3' AND `column name with space` IS NOT NU
AND `nested with space`.`inner with space` < 2
```

⚠ Field names containing periods (.) are NOT supported.

Hybrid Search
LanceDB
Schema Evolution