Managing Semantic View Verified Queries

A Programmatic Approach using Stored Procedures

Executive Summary

Verified queries improve Cortex Analyst accuracy by providing pre-validated question/SQL pairs. While Snowflake's UI supports adding verified queries via Snowsight, there is no native **ALTER SEMANTIC VIEW ADD VERIFIED QUERY** SQL command. This guide provides stored procedures that enable programmatic management of verified queries through the SQL API.

The Challenge

Cortex Analyst uses verified queries to improve response accuracy. These queries are stored in the semantic view's YAML specification under the `verified_queries` section. However, the current SQL interface only supports:

- ALTER SEMANTIC VIEW ... RENAME TO
- ALTER SEMANTIC VIEW ... SET COMMENT
- ALTER SEMANTIC VIEW ... UNSET COMMENT

There is no `ADD VERIFIED QUERY` option, making automation difficult.

The Solution

We leverage two system functions to create a workaround:

- **SYSTEM$READ_YAML_FROM_SEMANTIC_VIEW** - Reads the current YAML specification
- **SYSTEM$CREATE_SEMANTIC_VIEW_FROM_YAML** - Creates/replaces a semantic view from YAML (includes COPY GRANTS)

> **Important:** Agents reference semantic views by name, not by internal ID. When a semantic view is replaced, existing agent configurations remain valid.

Stored Procedures

1. ADD_VERIFIED_QUERY

Adds or updates a verified query in a semantic view.

```
CREATE OR REPLACE PROCEDURE ADD_VERIFIED_QUERY(
    SEMANTIC_VIEW_NAME VARCHAR,  -- Fully qualified: DB.SCHEMA.VIEW
    QUERY_NAME VARCHAR,          -- Unique identifier for the query
    QUESTION VARCHAR,            -- Natural language question
    SQL_QUERY VARCHAR            -- SQL using logical table names (__tablename)
)
RETURNS VARCHAR
LANGUAGE PYTHON
RUNTIME_VERSION = '3.11'
PACKAGES = ('snowflake-snowpark-python', 'pyyaml')
HANDLER = 'add_verified_query'
EXECUTE AS CALLER
```

Usage Example:

```
CALL ADD_VERIFIED_QUERY(
    'ANALYTICS.SEMANTIC_MODELS.REVENUE_VIEW',
    'monthly_revenue',
    'What is the total revenue by month?',
    'SELECT DATE_TRUNC(''month'', order_date) AS month,
            SUM(revenue) AS total_revenue
     FROM __orders GROUP BY 1 ORDER BY 1'
);
```

2. LIST_VERIFIED_QUERIES

Lists all verified queries in a semantic view.

```
CREATE OR REPLACE PROCEDURE LIST_VERIFIED_QUERIES(
    SEMANTIC_VIEW_NAME VARCHAR
)
RETURNS TABLE (NAME VARCHAR, QUESTION VARCHAR, SQL VARCHAR)
LANGUAGE PYTHON
RUNTIME_VERSION = '3.11'
PACKAGES = ('snowflake-snowpark-python', 'pyyaml')
HANDLER = 'list_verified_queries'
EXECUTE AS CALLER
```

Usage Example:

```
CALL LIST_VERIFIED_QUERIES('ANALYTICS.SEMANTIC_MODELS.REVENUE_VIEW');

-- Returns:
-- NAME              | QUESTION                          | SQL
-- monthly_revenue   | What is the total revenue by month?| SELECT DATE_TRUNC...
```

3. REMOVE_VERIFIED_QUERY

Removes a verified query from a semantic view by name.

```
CREATE OR REPLACE PROCEDURE REMOVE_VERIFIED_QUERY(
    SEMANTIC_VIEW_NAME VARCHAR,
    QUERY_NAME VARCHAR
)
RETURNS VARCHAR
```

```
LANGUAGE PYTHON
RUNTIME_VERSION = '3.11'
PACKAGES = ('snowflake-snowpark-python', 'pyyaml')
HANDLER = 'remove_verified_query'
EXECUTE AS CALLER
```

Full Implementation Code

The complete Python handler for ADD_VERIFIED_QUERY (other procedures follow similar patterns):

```
import yaml
from snowflake.snowpark import Session

def add_verified_query(session: Session, semantic_view_name: str,
                       query_name: str, question: str, sql_query: str) -> str:
    # Parse fully qualified name
    parts = semantic_view_name.replace('"', '').split('.')
    if len(parts) != 3:
        return f"Error: Must be fully qualified (DB.SCHEMA.VIEW)"

    db_name, schema_name, view_name = parts
    fully_qualified_schema = f"{db_name}.{schema_name}"

    # Read current YAML
    result = session.sql(
        f"SELECT SYSTEM$READ_YAML_FROM_SEMANTIC_VIEW('{semantic_view_name}')"
    ).collect()
    current_yaml = result[0][0]

    # Parse and modify YAML
    sv_spec = yaml.safe_load(current_yaml)

    new_vq = {'name': query_name, 'question': question, 'sql': sql_query}

    if 'verified_queries' not in sv_spec:
        sv_spec['verified_queries'] = []

    # Check for duplicate and update or append
    existing_names = [vq.get('name') for vq in sv_spec['verified_queries']]
    if query_name in existing_names:
        for i, vq in enumerate(sv_spec['verified_queries']):
            if vq.get('name') == query_name:
                sv_spec['verified_queries'][i] = new_vq
                break
        action = "updated"
    else:
        sv_spec['verified_queries'].append(new_vq)
        action = "added"

    # Recreate semantic view
    updated_yaml = yaml.dump(sv_spec, default_flow_style=False,
                             allow_unicode=True, sort_keys=False)
    escaped_yaml = updated_yaml.replace("'", "''")

    session.sql(f"""
        CALL SYSTEM$CREATE_SEMANTIC_VIEW_FROM_YAML(
            '{fully_qualified_schema}', '{escaped_yaml}'
        )
    """).collect()

    return f"Successfully {action} verified query '{query_name}'"
```

Best Practices

Use Logical Table Names:

In verified query SQL, reference logical tables with double-underscore prefix (e.g., `__orders` not `SALES.PUBLIC.ORDERS`).

Unique Query Names:

Use descriptive, unique names for each verified query to enable easy management.

Test Queries First:

Validate SQL syntax and results before adding as a verified query.

Grant Management:

SYSTEM$CREATE_SEMANTIC_VIEW_FROM_YAML automatically preserves grants (COPY GRANTS behavior).

Backup Before Bulk Changes:

Use SYSTEM$READ_YAML_FROM_SEMANTIC_VIEW to backup the YAML before making bulk modifications.

SQL API Integration

These procedures can be called via the Snowflake SQL API for automation:

```
curl -X POST "https://<account>.snowflakecomputing.com/api/v2/statements" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -H "X-Snowflake-Authorization-Token-Type: PROGRAMMATIC_ACCESS_TOKEN" \
  -d '{
    "statement": "CALL ADD_VERIFIED_QUERY(...)",
    "warehouse": "COMPUTE_WH"
  }'
```