FAO Apache AGE Database Query Guide

Understanding the Graph Data Model

Key Concepts

1. **Graph Structure in AGE**:
   - Nodes and relationships are stored in AGE tables within the `fao` schema
   - Access via `cypher()` function: `cypher('fao', $$ QUERY $$)`
   - Results return as `agtype` - cast appropriately
2. **Source Dataset Tracking**:
   - Each reference table has `source_dataset` field
   - Important for distinguishing duplicate entries (e.g., area_codes from different sources)
   - Filter when needed: `WHERE c.source_dataset = 'production_crops_livestock'`
3. **Area Code Patterns**:
   - Individual countries: 1-3 digit codes (e.g., "231" = USA, "156" = China)
   - Regional aggregates: Start with "5" (e.g., "5000" = World, "5100" = Africa)
   - AQUASTAT duplicates: Filter with `WHERE area_code NOT LIKE 'aquastat-%'`
4. **Foreign Key References**:
   - Relationships store `sql_id` references from PostgreSQL
   - Join back to foreign tables for details when needed
   - Element lookups require matching on sql_id

Basic Query Syntax

AGE Query Structure

```sql
-- Basic pattern
SELECT * FROM cypher('fao', $$
    MATCH pattern
    WHERE conditions
    RETURN results
$$) as (column1 agtype, column2 agtype, ...);

-- With parameters
SELECT * FROM cypher('fao', $$
    MATCH (c:Country {area_code: $code})
    RETURN c.name
$$, json_build_object('code', '156')) as (name agtype);
```

Essential Setup (Run Once Per Session)

```sql
-- Load AGE extension
LOAD 'age';
SET search_path = ag_catalog, "$user", public;
```

Common Query Patterns

Finding Available Data

sql

```sql
-- What commodities are tracked?
SELECT * FROM cypher('fao', $$
    MATCH (i:Item)
    RETURN DISTINCT i.item_code, i.name
    ORDER BY i.name
    LIMIT 50
$$) as (code agtype, name agtype);

-- What countries have production data?
SELECT * FROM cypher('fao', $$
    MATCH (c:Country)-[:PRODUCES]->()
    RETURN DISTINCT c.area_code, c.name
    ORDER BY c.name
$$) as (code agtype, name agtype);

-- What years have trade data?
SELECT * FROM cypher('fao', $$
    MATCH ()-[t:TRADES_WITH]->()
    RETURN DISTINCT t.year
    ORDER BY t.year DESC
$$) as (year agtype);
```

Production Queries

sql

```sql
-- Top wheat producers in 2023
SELECT * FROM cypher('fao', $$
    MATCH (c:Country)-[p:PRODUCES]->(i:Item)
    WHERE i.name = 'Wheat'
      AND p.year = 2023
      AND c.area_code !~ '^5'
    RETURN c.name, p.value, p.unit
    ORDER BY p.value DESC
    LIMIT 10
$$) as (country agtype, production agtype, unit agtype);

-- Production with element details (joining back to PostgreSQL)
WITH prod_data AS (
    SELECT * FROM cypher('fao', $$
        MATCH (c:Country {area_code: '156'})-[p:PRODUCES]->(i:Item)
        WHERE p.year = 2023
        RETURN c.name, i.name, p.element_code_id, p.value, p.unit
    $$) as (country agtype, item agtype, element_id agtype, value agtype, unit agtype)
)
SELECT
    pd.country::text,
    pd.item::text,
    e.element as measurement_type,
    pd.value::float,
    pd.unit::text
FROM prod_data pd
JOIN elements e ON e.id = (pd.element_id::text)::integer
ORDER BY pd.value::float DESC;
```

Trade Analysis

```sql
-- Trade relationships for a commodity
SELECT * FROM cypher('fao', $$
    MATCH (exporter:Country)-[t:TRADES_WITH]->(importer:Country)
    WHERE t.year = 2023
      AND t.item_code_id = $item_id
    RETURN exporter.name, importer.name, t.value, t.unit
    ORDER BY t.value DESC
    LIMIT 20
$$, json_build_object('item_id',
    (SELECT id FROM item_codes WHERE item_code = '0111' LIMIT 1)
)) as (exporter agtype, importer agtype, value agtype, unit agtype);

-- Multi-hop trade paths
SELECT * FROM cypher('fao', $$
    MATCH path = (origin:Country {area_code: '076'})-[:TRADES_WITH*1..3]->(dest:Country)
    WHERE ALL(r IN relationships(path) WHERE r.year = 2023)
    RETURN
        [n IN nodes(path) | n.name] as route,
        length(path) as hops,
        relationships(path)[-1].value as final_trade_value
    ORDER BY hops, final_trade_value DESC
    LIMIT 10
$$) as (route agtype, hops agtype, value agtype);
```

Hybrid SQL-Graph Queries

Combining Aggregations with Graph Traversal

```sql
-- Top producers with their trade network size
WITH production_stats AS (
    SELECT
        area_code_id,
        SUM(value) as total_production
    FROM production_crops_livestock
    WHERE year = 2023
      AND element_code_id = (SELECT id FROM elements WHERE element_code = '5510')
    GROUP BY area_code_id
    ORDER BY total_production DESC
    LIMIT 20
)
SELECT
    ac.area as country,
    ps.total_production,
    (SELECT * FROM cypher('fao', $$
        MATCH (c:Country {sql_id: $country_id})-[:TRADES_WITH]->()
        RETURN count(*) as trade_partners
    $$, json_build_object('country_id', ps.area_code_id))
    as (partners agtype))::text::integer as trade_partners
FROM production_stats ps
JOIN area_codes ac ON ac.id = ps.area_code_id
ORDER BY ps.total_production DESC;
```

Using Foreign Tables with Graph Results

sql

```sql
-- Get detailed flag information for production relationships
WITH flagged_production AS (
    SELECT * FROM cypher('fao', $$
        MATCH (c:Country)-[p:PRODUCES]->(i:Item)
        WHERE p.year = 2023
          AND p.flag_id IS NOT NULL
        RETURN c.name, i.name, p.flag_id, p.value
        LIMIT 100
    $$) as (country agtype, item agtype, flag_id agtype, value agtype)
)
SELECT
    fp.country::text,
    fp.item::text,
    fp.value::float,
    f.flag,
    f.description as data_quality
FROM flagged_production fp
JOIN flags f ON f.id = (fp.flag_id::text)::integer;
```

Time Series Analysis

sql

```sql
-- Production trends with moving average
WITH yearly_data AS (
    SELECT * FROM cypher('fao', $$
        MATCH (c:Country {area_code: '156'})-[p:PRODUCES]->(i:Item {item_code: '0111'})
        WHERE p.year >= 2010
        RETURN p.year, p.value
        ORDER BY p.year
    $$) as (year agtype, value agtype)
)
SELECT
    year::integer,
    value::float as production,
    AVG(value::float) OVER (
        ORDER BY year::integer
        ROWS BETWEEN 2 PRECEDING AND 2 FOLLOWING
    ) as moving_avg
FROM yearly_data;

-- Year-over-year growth
WITH production_pairs AS (
    SELECT * FROM cypher('fao', $$
        MATCH (c:Country {area_code: '076'})-[p1:PRODUCES]->(i:Item)
        MATCH (c)-[p2:PRODUCES]->(i)
        WHERE p2.year = p1.year + 1
          AND p1.year >= 2020
        RETURN i.name, p1.year, p1.value, p2.year, p2.value
    $$) as (item agtype, year1 agtype, value1 agtype, year2 agtype, value2 agtype)
)
SELECT
    item::text,
    year1::integer as year,
    value1::float as current_value,
    value2::float as next_value,
    ROUND(((value2::float - value1::float) / value1::float * 100)::numeric, 2) as growth_pct
FROM production_pairs
WHERE value1::float > 0
ORDER BY item::text, year1::integer;
```

Advanced Patterns

Finding Data Gaps

sql

```sql
-- Countries missing recent data
WITH recent_producers AS (
    SELECT DISTINCT (country::text) as country_name
    FROM cypher('fao', $$
        MATCH (c:Country)-[p:PRODUCES]->()
        WHERE p.year = 2023
        RETURN c.name as country
    $$) as (country agtype)
),
older_producers AS (
    SELECT DISTINCT (country::text) as country_name
    FROM cypher('fao', $$
        MATCH (c:Country)-[p:PRODUCES]->()
        WHERE p.year = 2020
        RETURN c.name as country
    $$) as (country agtype)
)
SELECT o.country_name as missing_recent_data
FROM older_producers o
LEFT JOIN recent_producers r ON o.country_name = r.country_name
WHERE r.country_name IS NULL
ORDER BY o.country_name;
```

Cross-Dataset Analysis

```sql
-- Countries that produce but don't trade
WITH producers AS (
    SELECT * FROM cypher('fao', $$
        MATCH (c:Country)-[:PRODUCES]->(i:Item {item_code: '0111'})
        WHERE c.area_code !~ '^5'
        RETURN DISTINCT c.area_code, c.name
    $$) as (code agtype, name agtype)
),
traders AS (
    SELECT * FROM cypher('fao', $$
        MATCH (c:Country)-[t:TRADES_WITH]->()
        WHERE t.item_code_id = $wheat_id
        RETURN DISTINCT c.area_code
    $$, json_build_object('wheat_id',
        (SELECT id FROM item_codes WHERE item_code = '0111')
    )) as (code agtype)
)
SELECT p.name::text as produces_but_doesnt_trade
FROM producers p
LEFT JOIN traders t ON p.code = t.code
WHERE t.code IS NULL
ORDER BY p.name::text;
```

Performance Optimization

Use Indexes on Graph Properties

```sql
-- Check what indexes exist
SELECT * FROM pg_indexes
WHERE schemaname = 'fao';

-- Create property indexes if needed
CREATE INDEX idx_country_area_code
ON fao.country USING btree ((properties->>'area_code'));

CREATE INDEX idx_trades_year
ON fao.trades_with USING btree ((properties->>'year'));
```

Limit Early in Queries

```sql
-- Good: Filter in graph query
SELECT * FROM cypher('fao', $$
    MATCH (c:Country)-[p:PRODUCES]->(i:Item)
    WHERE p.year = 2023
    RETURN c.name, sum(p.value) as total
    ORDER BY total DESC
    LIMIT 10
$$) as (country agtype, total agtype);

-- Avoid: Filter after graph query
SELECT * FROM (
    SELECT * FROM cypher('fao', $$
        MATCH (c:Country)-[p:PRODUCES]->(i:Item)
        RETURN c.name, p.year, p.value
    $$) as (country agtype, year agtype, value agtype)
) sub
WHERE year::integer = 2023
LIMIT 10;
```

Common Issues and Solutions

Issue: Duplicate Nodes from Different Sources

```sql
-- Problem: Multiple USA nodes
SELECT * FROM cypher('fao', $$
    MATCH (c:Country)
    WHERE c.area_code = '231'
    RETURN c.name, c.source_dataset
$$) as (name agtype, source agtype);

-- Solution: Filter by source dataset
SELECT * FROM cypher('fao', $$
    MATCH (c:Country)
    WHERE c.area_code = '231'
      AND c.source_dataset = 'area_codes'
    RETURN c.name
$$) as (name agtype);
```

Issue: Element Name Lookups

```sql
-- Don't use element names in relationships
-- Relationships store element_code_id

-- Correct approach:
WITH element_lookup AS (
    SELECT id, element, element_code
    FROM elements
    WHERE element LIKE '%Production%'
)
SELECT * FROM cypher('fao', $$
    MATCH (c:Country)-[p:PRODUCES]->(i:Item)
    WHERE p.element_code_id = ANY($elem_ids)
      AND p.year = 2023
    RETURN c.name, i.name, p.value
$$, json_build_object('elem_ids',
    ARRAY(SELECT id FROM element_lookup)
)) as (country agtype, item agtype, value agtype);
```

Issue: Type Casting

```sql
-- AGE returns agtype - cast for calculations
SELECT
    (country::text) as country_name,
    (value::text)::float as production_value,
    ROUND((value::text)::numeric, 2) as rounded_value
FROM cypher('fao', $$
    MATCH (c:Country)-[p:PRODUCES]->()
    WHERE p.year = 2023
    RETURN c.name as country, p.value as value
    LIMIT 5
$$) as (country agtype, value agtype);
```

Useful Query Templates

Explore Graph Structure

```sql
-- Count nodes by label
SELECT * FROM cypher('fao', $$
    MATCH (n)
    RETURN labels(n)[0] as label, count(*) as count
$$) as (label agtype, count agtype);

-- Count relationships by type
SELECT * FROM cypher('fao', $$
    MATCH ()-[r]->()
    RETURN type(r) as rel_type, count(*) as count
$$) as (rel_type agtype, count agtype);

-- Sample nodes with properties
SELECT * FROM cypher('fao', $$
    MATCH (n:Country)
    RETURN properties(n)
    LIMIT 5
$$) as (props agtype);
```

Data Quality Checks

```sql
-- Find relationships with missing values
SELECT * FROM cypher('fao', $$
    MATCH ()-[r:PRODUCES]->()
    WHERE r.value IS NULL OR r.value = 0
    RETURN type(r), count(*) as null_count
$$) as (rel_type agtype, count agtype);

-- Verify foreign key integrity
WITH orphaned AS (
    SELECT * FROM cypher('fao', $$
        MATCH ()-[p:PRODUCES]->()
        WHERE p.flag_id IS NOT NULL
        RETURN DISTINCT p.flag_id as flag_id
    $$) as (flag_id agtype)
)
SELECT o.flag_id::text::integer as orphaned_flag_id
FROM orphaned o
LEFT JOIN flags f ON f.id = (o.flag_id::text)::integer
WHERE f.id IS NULL;
```

Best Practices

1. **Always check source_dataset** when accuracy matters
2. **Filter regional aggregates** with `area_code !~ '^5'` for country-only analysis
3. **Use parameters** for repeated queries to improve performance
4. **Cast agtype appropriately** for calculations and joins
5. **Leverage hybrid queries** - combine SQL aggregations with graph traversals
6. **Limit results during exploration** to avoid overwhelming result sets
7. **Join to PostgreSQL tables** for detailed reference data
8. **Create helper functions** for common patterns

Quick Reference

sql

```sql
-- Session setup
LOAD 'age';
SET search_path = ag_catalog, "$user", public;

-- Basic query
SELECT * FROM cypher('fao', $$ MATCH (n) RETURN n LIMIT 1 $$) as (n agtype);

-- With parameters
SELECT * FROM cypher('fao', $$
    MATCH (n:Country {area_code: $code})
    RETURN n.name
$$, '{"code": "156"}'::jsonb) as (name agtype);

-- Type casting
(agtype_value::text)::integer
(agtype_value::text)::float
(agtype_value::text)::boolean
agtype_value::text  -- for strings
```