**📂 Data Retention Workflow Documentation**

**🖼️ Visual Flowchart**

*Animated GIF-style diagram illustrating the full data retention process including scheduled/manual execution, validation, logging, and rollback.*

**🏗️ Database Schema Changes**

**✅ Existing Table: entity (Unchanged)**

| **Column Name** | **Type** | **Description** |
| --- | --- | --- |
| entity\_id | STRING | Unique ID for the entity (primary key) |
| table\_name | STRING | Logical table name |
| datasource\_id | STRING | System or source identifier |
| last\_updated | DATETIME | Timestamp of the last update |
| ... | ... | ... |

**🆕 New Table: data\_retention\_entity\_config**

| **Column Name** | **Type** | **Description** |
| --- | --- | --- |
| config\_id | STRING/UUID | Primary key for config entry |
| entity\_id | STRING | Foreign key to entity.entity\_id |
| retention\_enabled | BOOLEAN | Flag to enable/disable retention for this entity |
| retention\_period | INT | Retention period in years |
| cluster\_type | STRING | Preferred cluster type for deletion (e.g., memory\_optimized, storage\_optimized) |
| max\_workers | INT | Max number of parallel workers for deletion jobs |
| updated\_by | STRING | User who last updated this config |
| updated\_on | DATETIME | Timestamp of last update |
| notes | STRING | Optional notes or comments |

**🆕 New Table: data\_retention\_log**

| **Column Name** | **Type** | **Description** |
| --- | --- | --- |
| entity\_id | STRING | Unique identifier of the entity |
| table\_type | STRING | Type of table (e.g., current, history, main, log) |
| validation\_status | STRING | Status (e.g., SUCCESS, FAILED, REVERTED) |
| delete\_count | INT | Number of records deleted |
| message | STRING | Log message (error, status, etc.) |
| execution\_time | TIMESTAMP | Time of deletion execution |
| pipeline\_run\_id | STRING | ADF pipeline run identifier |

**🔁 ADF Pipeline: data\_retention\_deletion**

**🔧 Parameters**

* malcode – Market/Region identifier
* entity – Target entity/table
* execution\_flag – Type of run: SCHEDULED or MANUAL

**🔄 Execution Flow**

1. **Trigger**: Initiated quarterly or on-demand via ADF
2. **Entity Check**: Query data\_retention\_entity\_config for retention-enabled entities (retention\_enabled = true)
3. **Task Queue**: Add eligible entries to the retention task queue
4. **Cluster Selection**:
   * Fetch cluster\_type and max\_workers from data\_retention\_entity\_config
   * Select cluster accordingly:
     + **Memory Optimized** cluster (e.g., Azure Standard\_E16as\_v4 or AWS r5d.4xlarge) recommended for:
       - Large tables (millions/billions of rows)
       - Heavy delete operations with Delta CDF validation
     + **Storage Optimized** clusters for less memory-intensive, high storage throughput tasks
5. **Databricks Notebook Execution**:
   * Build delete queries based on retention period
   * Pre-delete record counts from:
     + table\_current
     + table\_history
     + table\_main
     + table\_log
6. **Perform Deletion** per table in parallel up to max\_workers
7. **Validation with Delta CDF**:
   * Confirm expected number of rows deleted
8. **Rollback Logic**:
   * If validation fails, revert Delta table to previous version immediately
9. **Logging**:
   * Log results into data\_retention\_log
   * Send metrics and alerts to Datadog

**💻 Manual Execution (Ad Hoc)**

* Same pipeline logic triggered manually via ADF UI/API
* User inputs parameters: entity, malcode, execution\_flag=MANUAL
* Allows testing, audit-triggered runs, or emergency data cleanup

**🆕 ADF Pipeline: update\_data\_retention\_period**

**🎯 Purpose**

* Controlled update of retention configs with auditability and approval

**🔧 Parameters**

* entity
* new\_retention\_period
* chg\_request\_id
* requested\_by
* approved\_by

**🔒 Approval Process**

* Requires proper Change Request ID and approval before update
* Updates recorded in data\_retention\_entity\_config with audit columns

**💻 Infrastructure Design & Cluster Recommendation**

**🧠 Cluster Types and Selection Criteria**

| **Cluster Type** | **VM Examples** | **Use Case** | **Why?** |
| --- | --- | --- | --- |
| Memory Optimized | Azure Standard\_E16as\_v4 AWS r5d.4xlarge | Large datasets with billions of rows Heavy Delta operations, CDF validation | High RAM to cache data and speed up queries and deletes Reduces IO latency and improves shuffle performance |
| Storage Optimized | Azure Standard\_L80s AWS i3.4xlarge | Large volume data but less intensive compute needed | Large local SSD storage Good for bulk IO operations, but less memory |

**🛠️ Recommended Settings**

* **Workers**: 4 to 8 workers minimum for parallelism; tune based on job duration and cluster utilization
* **Auto-scaling**: Enabled to save cost during idle times
* **Photon engine**: Enabled for Delta Lake optimization
* **Auto-termination**: 30 minutes idle timeout

**📊 Monitoring, Observability, and Alerts**

**📈 Datadog Metrics**

* Number of records deleted by entity and table type
* Execution duration and cluster resource usage
* Validation status and rollback events

**🛎️ Alerts**

* Failure in deletion or validation process
* Manual execution triggers
* Change approval events

**✅ Summary**

* Using a **separate config table linked by entity\_id** provides flexibility and clarity.
* Cluster type and worker concurrency are configurable per entity for efficient resource utilization.
* Memory optimized clusters are recommended for heavy delete and Delta CDF validation on large tables.
* Full audit and change control built into the pipeline and config management.
* The design supports both scheduled and ad hoc retention enforcement with robust validation and rollback.

@startuml

title Data Retention Pipeline Flow

start

:Trigger - Quarterly;

:Query DataRetentionConfig table;

:Populate task queue per entity;

partition "Task Scheduling" {

:Wait for cluster availability;

:Check for active ingestion tasks;

if (No ingestion running?) then (Yes)

:Prioritize & schedule tasks;

else

:Delay task scheduling;

endif

}

partition "Deletion Process" {

:Check if CDF is enabled;

if (CDF enabled?) then (Yes)

:Log current Delta version;

:Get count of records to delete;

:Delete records from tables;

else

:Log task as failed;

stop

endif

}

partition "Validation" {

:Validation Approach 1;

:Verify count using CDF;

if (Counts match?) then (Yes)

:Proceed to Logging;

else

:Trigger rollback to previous Delta version;

stop

endif

:Validation Approach 2;

:Delete first, then check max dates from CDF;

:Final count check;

if (Count correct?) then (Yes)

:Proceed to Logging;

else

:Trigger rollback;

stop

endif

}

partition "Logging" {

:Log actions and outcomes;

:Send metrics to Datadog;

}

partition "Future Enhancements" {

:If deleted count = 0;

:Check min effective date from config;

:Update or log date;

:Schedule only if retention > min date;

}

stop

@enduml