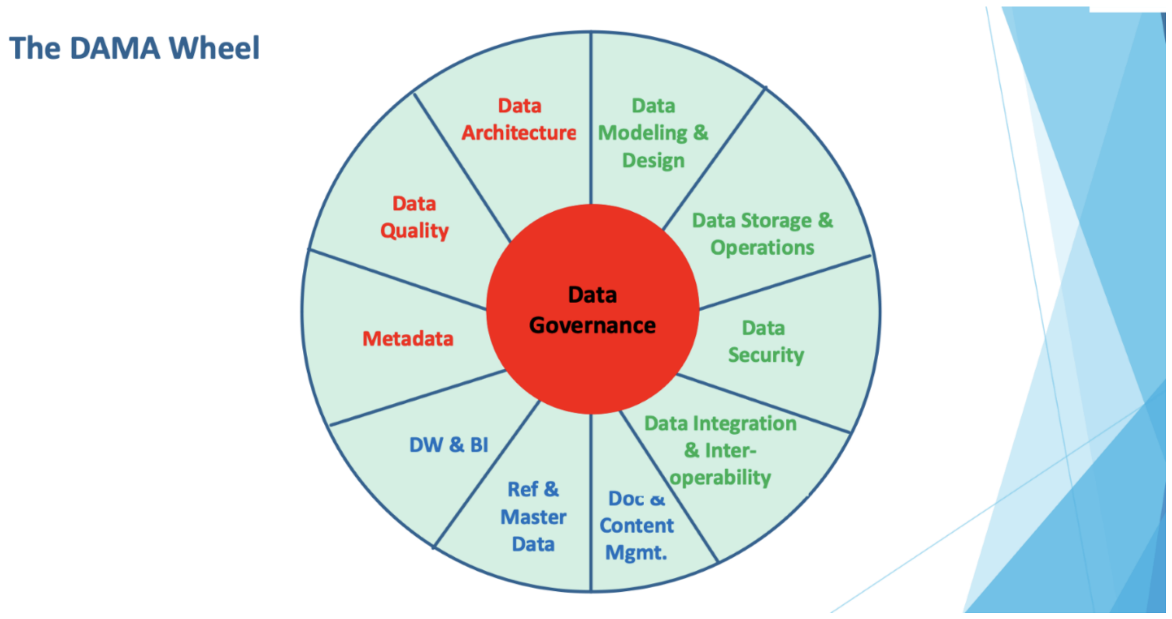
Data Governance Framework

Introduction

In our project, we have applied a Data Governance Framework using DAMA wheel concepts. From the DAMA wheel concepts, we chose below sub-domains that are applicable to the project.



1. Metadata Management.
2. Data Modeling and Design.
3. Data Security.

This involves principles like Audit, Quality, Formalization, Knowledge Retention, Risk reduction, and Clear Accountability. We have developed policies under each of these principles and developed procedures for those policies.

In our project, we have used each of the principles to make sure our data is well-behaved. We did this from the beginning when we created our data structure, and we have kept it up throughout the project.

For example, we made sure that all the information about our data, like where it came from and who changed it, was recorded. Each entity has a dedicated audit table, and the lifecycle of the data is recorded through it.

We also paid attention to how we name things and how we organize our data. We have made sure that everything is clear and matches up.

So, in simple terms, we have used these rules all the way through to manage our data properly, from the start of our project to now.

Dataset

The dataset which we are provided to work with is a Sample Superstore excel spreadsheet which comprises 9994 records representing an Order placed by a customer across 4 regions Central, West, East and South and their associated details.

1. Subdomain: Metadata management

1.1 Principle: Audit

1.1.1) Policies:

1. **Metadata Lineage Policy:** Every table should have associated metadata that should show the lineage of the respective data.
2. **Metadata Archival and Retrieval Policy:** Audit metadata tables should be archived every month and retrievable in 6 hours if needed.

1.1.2) Procedures:

1. Create a table with audit ID, <Primary key of the entity table>, and metadata columns like "createdAt", "updatedAt", "createdBy", and "updatedBy."
2. If there is a status change kind of structure in the entity table, it should possess a separate column in metadata like "statusChangedBy."
3. If some of the values in the entity table were always created/updated by the system user, then the respective metadata table does not need "createdBy", or "updatedBy" columns.
4. Auto expiry should be configured in all the metadata tables, and expired data should be pushed to the low-cost cloud storage.

1.2 Principle: Quality

1.2.1) Policies:

1. **Change communication:** All changes in the metadata should be communicated throughout the company to ensure accurate usage of metadata.

2. **Data Validation:** Metadata entries should undergo validation processes to identify and rectify errors or inconsistencies.
3. **Data Robustness:** Audit Metadata entries should only get updated if values of source entities are changed.

1.2.2) Procedures:

1. Business metadata should not have ambiguity between glossaries.
2. In the ERD diagram, we have implemented audit tables for each entity table. While implementing audit tables, we should implement them in such a way that metadata should get updated only if the value of corresponding entity tables are changed. It should not create new items in the audit table unnecessarily.
3. Operational Metadata and Business Metadata updation should be communicated through newsletters every week to both internal and external stakeholders. Changes should be applied only after communicating with the stakeholders.

2. Subdomain: Data Modeling and Design

2.1 Principle: Formalization

2.1.1) Policies:

1. **Entity Formalization Policy**: Entity addition should be formally moved from conceptual to logical to physical. The direct addition of tables without brainstorming the conceptual part is prohibited. It should not be entertained in any situations like tighter deadlines.
2. **Consistent Entity Naming and Matching Policy**: Entity names should be clear, descriptive, and free from ambiguity, making it easy for stakeholders to understand their purpose.
3. **Normalization Policy:** Ensure data is well-structured and normalized. For both OLTP and OLAP system, follow the appropriate standard design principles which are followed globally according to the type of systems.

2.1.2) Procedures:

1. Before adding a new table to the database, conduct a conceptual assessment to define its purpose, relevance, and business objectives. Once the conceptual assessment is complete, create a formal logical design that includes defining attributes, relationships, and constraints.

2. Ensure that entity names are consistently in singular form for clarity and ease of understanding. For example, if an organization has multiple databases storing customer information, this policy will require the "customer_id" attribute in each database uses the same data type (e.g., varchar), follows a consistent naming convention (e.g., "customer_id"), and enforces the same data integrity constraints (e.g., unique and not null)
3. In OLTP systems, maintain tables in at least the 3rd Normal Form (3NF) to reduce redundancy and preserve data integrity. For OLAP systems, prioritize sound design principles like sharding, etc. Optimize queries through techniques like indexing and caching, without sacrificing normalization. Compulsorily assign primary keys to enforce data uniqueness. Verify that constraints, such as foreign keys, align between physical and logical entities.

The database constructed in the previous exercises is an OLTP system, thus all the 6 tables were normalized to 3NF as per the Normalization policy. Also, all our entity names are singular, ambiguity-free, and Logical and physical names match together as per the naming policy. We have created the tables by doing an ERD diagram as per the Entity Formalization policy.

2.2 Principle: Knowledge Retention / Documentation

2.2.1) Policies:

1. **Change Management Policy:** Each change should be propagated from conceptual to physical entity. Entity versioning should be followed.
2. **Consistency across Documentation:** Templates should be standardized and versioned that should be followed across the teams. Ownership of documentation should be fixed and clearly stated.

2.2.2) Procedures:

1. Implement standardized templates for documenting data models and designs. These templates should include sections for essential information, such as data entity descriptions, relationships, attributes, and design rationale.
2. Enforce naming conventions for documents to ensure consistency. For example, use clear and consistent file naming and folder structures for organizing documentation.
3. Assign clear ownership of documentation to responsible individuals or teams. They should be accountable for keeping the documentation up to date and accurate.
4. Assign ownership of template to the role of the people. For example, SDE-3 should be taking care of maintaining templates of technical documentation, regardless of individual.

3. Subdomain: Data Security

3.1 Principle: Reduce risk by reducing exposure

3.1.1) Policies:

1. **Data Access Policy:** Each entity should be audited during the modeling phase and a list of roles that could do CRUDE operations should be determined before moving to a logical stage of the modeling phase.
2. **Minimal Exposure Policy:** No entities should be open and only exposed to necessary processes.
3. **Deletion Policy:** No master data can be deleted from the system, only archival is allowed.

3.1.2) Procedures:

1. During the modeling phase, conduct a comprehensive audit of each entity to understand its data sensitivity and access requirements. Document the findings to identify potential security risks.
2. For each entity, create a list of roles that require access for Create, Read, Update, Delete, and Execute (CRUDE) operations. Based on the crude matrix outcomes, Role-based access control should be implemented over all the entities.
3. Classify data entities based on sensitivity. Identify entities that contain highly sensitive data and limit their exposure to only essential processes and roles. Here in our sample_superstore database, customer entity data could be limited/obfuscated and should only be able to access based on the approvals.
4. Instead of deletion, master data should only be archived. Archived data should be securely stored, ensuring it is retrievable when needed but not actively accessible in the live system. Here in our sample_superstore database, "product" (Product Entity) that are no longer in the system can be archived and moved to cheaper storage like AWS Glacier.

3.2 Principle: Clear accountability

3.2.1) Policies:

1. **Data Ownership Accountability:** Clearly define data ownership for each data entity, assigning responsibility and accountability for its security and integrity.
2. **Incident Response Accountability and Communication:** Establish clear accountability for incident response actions and decision-making.

3.2.2) Procedures:

1. Assign a data owner for each data entity, specifying their roles and responsibilities for data security and integrity. For example, “Product” entity ownership could be assigned to the product catalog management team which should be responsible for the complete lifecycle of that data from creation to archival.
2. Maintain an inventory of all data entities and their respective data owners to ensure accountability is established across the data landscape. Also, communicate the changes in ownership across the company.
3. Form an incident response team with designated roles and responsibilities for incident handling. Communicate with the respective stakeholders as soon as possible. For example, If Customer entity data is leaked, then communicating with the customer is necessary along with the legislative bodies.