**Problem statement**

**1. Context & Overview**

The U.S. tariff system has gone through **three distinct policy phases**:

* **President 1**: Imposed **item-level tariffs** on 10 countries.
* **President 2**: Expanded the tariff policy to **4 additional countries** at the item level.
* **President 3**: Introduced **component-level tariffs**, meaning duties now apply to materials (e.g., steel, aluminum) within products.

**Challenge**: This evolution demands a flexible system capable of understanding and applying different layers of rules, including historical tariffs, item structure (BoM), and changing political directives.

**2. Requirements**

**Functional Requirements**

* **Tariff Rule Definition**: Define rules by country, item, or component.
* **Calculation Engine**: Dynamically calculate total tariff based on the applicable rules.
* **Component Mapping**: Link finished goods to raw materials using a Bill of Materials (BoM).
* **Versioning**: Support rule changes over time (e.g., per presidency).
* **Integration**: Connect with external import/export or logistics systems.
* **Auditing & Reporting**: Track policy changes and report on impact.

**Non-Functional Requirements**

* **Extensibility**: Add new rule types, components, or countries easily.
* **Performance**: Complete tariff calculation in under 200ms.
* **Security**: Enforce role-based access for rule editing vs. viewing.
* **Scalability**: Handle a growing number of rules and item-component combinations.
* **Data Integrity**: Validate country codes, rates, and hierarchies.

**3. High-Level Design (HLD)**

External Systems

External Systems

Admin Console

Network Layer

Load Balancer

API Gateway

Microservice

Layer

Tariff Rule Manager

Policy Manager

Message Queues (Policy, Items, Components, Tariff Rules, Updates)

BOM Resolver

Tariff Engine

Data Layer

Redis Cache (Frequently Accessed Data)

Relational DB (Tariff Rules, Items, Components, Policies, Audit Logs)

Analytics & Reporting

### Detailed Explanation of Components and Interactions:

1. **External World:**
   * **Admin Console:** The web-based user interface for policymakers to define, test, draft, and deploy tariff rules. It interacts with the system via the Load Balancer and API Gateway.
   * **External Systems (Import/Export, Logistics):** Other enterprise systems that need to integrate with the tariff calculation engine to get final duty amounts or provide item/component data. They also interact via the Load Balancer and API Gateway.
2. **Network & Edge Layer:**
   * **Load Balancer (LB):** (e.g., Nginx, AWS ELB, Azure Load Balancer, GCP Load Balancing)
     + **Role:** Distributes incoming network traffic across multiple servers (in this case, instances of the API Gateway) to ensure high availability and responsiveness. It prevents any single point of failure and improves overall system performance by spreading the load.
     + **Reasoning:** Essential for handling a growing number of requests ("Scalability") and ensuring continuous service availability even if one API Gateway instance fails.
   * **API Gateway (AG):** (e.g., Kong, AWS API Gateway, Spring Cloud Gateway)
     + **Role:** Acts as the single entry point for all client requests to the microservices. It handles cross-cutting concerns such as authentication, authorization (Role-Based Access Control), rate limiting, request routing, and potentially caching or transformation.
     + **Reasoning:** Enhances "Security" by centralizing access control, simplifies client interactions by providing a unified API, and improves "Scalability" by offloading common tasks from individual microservices.
3. **Core Microservices Layer:**
   * This layer consists of independent, loosely coupled services, each responsible for a specific business capability. They communicate primarily via internal API calls (HTTP/gRPC) or asynchronously via a Message Queue.
   * **Tariff Rule Service (TR):** Manages the lifecycle (CRUD) of tariff definitions.
   * **Tariff Engine (TE):** The core logic for dynamically calculating tariffs based on item, country, and components. It orchestrates calls to BoM Resolver and Policy Manager.
   * **BoM Resolver (BR):** Provides the Bill of Materials for any given item, translating it into its constituent components.
   * **Policy Manager (PM):** Manages policy versions (e.g., presidential terms) and associates tariff rules with them, enabling historical lookups and future simulations.
   * **Analytics Service (AS):** Collects and processes data related to tariff calculations, rule changes, and policy impacts for auditing and reporting.
4. **Data Layer:**
   * **Relational Database (DB):** (e.g., PostgreSQL, MySQL, SQL Server)
     + **Role:** The primary persistent storage for all structured data, including TariffRule, Item, Component, PolicyVersion, Item\_Component (join table), and AuditLog.
     + **Reasoning:** Chosen for its strong "Data Integrity" (ACID properties, schema enforcement), support for complex queries, and transactional capabilities, which are crucial for accurate financial calculations and consistent rule management.
   * **Redis Cache (R):**
     + **Role:** An in-memory data store used for caching frequently accessed data (e.g., active tariff rules, common BoM structures) to reduce database load and improve read performance.
     + **Reasoning:** Directly addresses the "Performance" non-functional requirement by enabling tariff calculations to complete under 200ms by minimizing database round-trips for hot data.

**4. Low-Level Design (LLD)**

The data models are designed to capture the complexity of tariff rules at both item and component levels, support versioning, and link items to their components.

* **TariffRule:** Represents a single tariff definition.
  + tariff\_id: Unique identifier for the tariff rule (UUID).
  + level: Enum ("item" | "component") indicating whether the tariff applies to an item or a component.
  + entity\_id: Identifier for the item (ITEM123) or component (COMPONENT456).
  + country\_code: 3-letter ISO country code (e.g., "USA", "CHN").
  + tariff\_rate: Decimal representing the tariff percentage (e.g., 0.15 for 15%).
  + start\_date: Date from which the tariff rule becomes active (e.g., "2025-01-01").
  + end\_date: Optional date when the tariff rule expires. Null if perpetual.
  + policy\_version\_id: Foreign key linking to the PolicyVersion (e.g., "President 3 Policy").
  + status: Enum ("DRAFT" | "ACTIVE" | "INACTIVE" | "ARCHIVED").
  + created\_at: Timestamp of creation.
  + updated\_at: Timestamp of last update.
  + **Reasoning:** This model is central to defining tariff rules, allowing for granular control at item or component levels and associating them with specific policies or presidencies. The

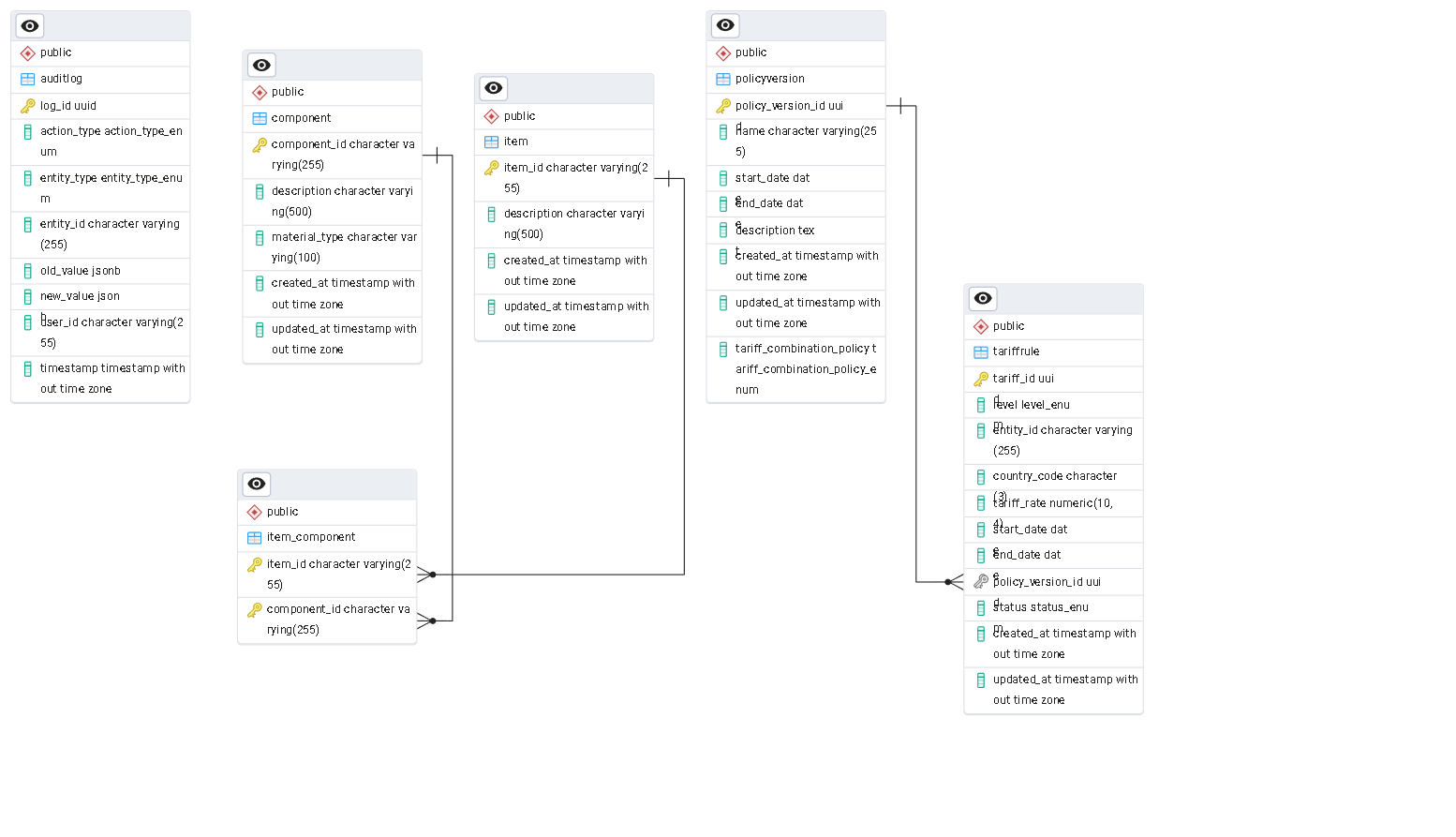
start\_date and end\_date support the "Versioning" requirement.

* **Item:** Represents a finished good.
  + item\_id: Unique identifier for the item (e.g., "ITEM123").
  + description: Human-readable description of the item (e.g., "Refrigerator").
  + components: Array of strings, where each string is a component\_id (e.g., ["STEEL\_A", "ALUMINUM\_B"]).
  + created\_at: Timestamp of creation.
  + updated\_at: Timestamp of last update.
  + **Reasoning:** This model establishes the Bill of Materials (BoM), crucial for component-level tariff calculations.
* **Component:** Represents a raw material or sub-part.
  + component\_id: Unique identifier for the component (e.g., "STEEL\_A").
  + description: Human-readable description (e.g., "High-grade Steel Sheet").
  + material\_type: (e.g., "steel", "aluminum", "plastic").
  + created\_at: Timestamp of creation.
  + updated\_at: Timestamp of last update.
  + **Reasoning:** Separating components into their own entity allows for independent management and consistent referencing across items.
* **PolicyVersion:** Represents a distinct policy phase (e.g., a presidential term).
  + policy\_version\_id: Unique identifier (e.g., UUID or "President1\_Term").
  + name: Descriptive name (e.g., "President 1 Policy", "President 2 Expansion").
  + start\_date: Date when the policy became effective.
  + end\_date: Optional date when the policy ended.
  + description: Details about the policy (e.g., "Imposed item-level tariffs on 10 countries").
  + created\_at: Timestamp of creation.
  + updated\_at: Timestamp of last update.
  + **Reasoning:** Essential for the "Versioning" requirement and for associating

TariffRule instances with specific historical contexts.

Code attached for reference

**4.1 Database Design**



**5.Forward-Thinking Enhancements Integration**

The proposed design lays a strong foundation for the suggested enhancements:

* **Simulation Engine:** Can leverage the Tariff Rule Service (in DRAFT status) and the Tariff Engine to calculate tariffs based on hypothetical rules without impacting live policies. The

Policy Manager can manage different simulation scenarios.

* **AI Suggestions:** The Analytics Service can provide historical trade data and tariff impacts, which can feed into an AI/ML model. The AI model could then suggest tariff changes, with suggestions being ingested by the

Tariff Rule Service (as DRAFT rules) for review in the Admin Console.

* **Conflict Resolution Engine:** The combine\_tariff function within the Tariff Engine is the designated point for implementing sophisticated conflict resolution logic. This logic could be configured via the

Policy Manager or even be AI-driven.

* **Staging & Approval Workflow:** The status field in the TariffRule table (DRAFT, ACTIVE) and the Admin Console provide the necessary components for a multi-stage approval workflow. Rules can be drafted, reviewed, approved, and then activated.
* **Multilateral Agreements:** The data model for TariffRule and PolicyVersion is flexible enough to incorporate rules stemming from multilateral agreements (e.g., by adding a treaty\_id to TariffRule and linking it to a new Treaty table). The

Tariff Engine would need to incorporate logic to prioritize or combine rules from such agreements.

**6. Conclusion**

This system design provides a modular, scalable, and flexible solution for managing the evolving U.S. tariff system. By separating concerns into distinct services, using a robust database design, and defining clear APIs, the system is well-equipped to handle current requirements and easily integrate future enhancements. The focus on versioning, auditing, and performance ensures transparency, control, and compliance in a dynamic trade environment.