**Descripción funcional del flujo de agregación en**

**DTOComposer**

El EventConsumer actúa como el punto único de entrada de todos los eventos provenientes del broker (Kafka), y es responsable de enrutar cada evento hacia el DTOComposer correspondiente. Dentro del DTOComposer, existen múltiples agregadores especializados que componen diferentes partes del DTO CustomerPosition. Estos agregadores encapsulan la lógica necesaria para interpretar los eventos y realizar la transformación o actualización del DTO.

Los escenarios cubiertos en este patrón son:

1. Un evento actualiza un solo campo del DTO

El evento es recibido, validado, y enruta directamente a un agregador específico (ej. BalanceAggregator) que actualiza el campo correspondiente. Luego, se envía la actualización al DTOPersist.

1. Un evento actualiza múltiples campos del DTO

El evento es procesado por más de un agregador, por ejemplo, tanto CardAggregator como AccountAggregator, en paralelo o en flujo coordinado. Ambos agregadores actualizan partes distintas del DTO.

1. Múltiples eventos actualizan un solo campo del DTO

El EventConsumer agrupa los eventos relacionados por tipo y cliente. Una vez que se ha recolectado suficiente información, los eventos son enviados al AccountAggregator para consolidar la actualización.

1. Múltiples eventos actualizan múltiples campos del DTO

Este es el escenario más complejo. Se produce una agregación combinada que alimenta simultáneamente a varios agregadores (ej. CardAggregator, BalanceAggregator, InsuranceAggregator). El resultado conjunto es un DTO completo o parcialmente actualizado que se persiste en Oracle y/o Redis.

Una vez que todos los datos han sido consolidados, DTOPersist los almacena en los repositorios correspondientes (Redis JSON como cache L1 y Oracle Converged Database como persistencia L2), y opcionalmente se genera un nuevo evento usando el EventGenerator.

sequenceDiagram

participant Kafka as Kafka Broker

participant EventConsumer

participant EventRouter

participant DTOComposer

participant BalanceAggregator

participant CardAggregator

participant AccountAggregator

participant DTOPersist

participant Redis

participant Oracle

participant EventGenerator

Kafka->>EventConsumer: Consume Event (e.g., AccountUpdated)

EventConsumer->>EventRouter: Route by eventType + customerId

EventRouter-->>DTOComposer: Dispatch event to DTOComposer

alt Single field update

DTOComposer->>BalanceAggregator: Update balance

BalanceAggregator->>DTOPersist: Forward updated DTO

end

alt Multiple field update from one event

DTOComposer->>CardAggregator: Update card info

DTOComposer->>AccountAggregator: Update account info

CardAggregator->>DTOPersist: Partial DTO

AccountAggregator->>DTOPersist: Partial DTO

end

alt Multiple events for single field

loop Grouping by customerId

EventConsumer->>DTOComposer: Aggregate events

end

DTOComposer->>AccountAggregator: Consolidated account update

AccountAggregator->>DTOPersist: Updated DTO

end

alt Multiple events for multiple fields

loop Grouping multiple events

EventConsumer->>DTOComposer: Aggregated stream

end

DTOComposer->>BalanceAggregator: Update balance

DTOComposer->>CardAggregator: Update cards

DTOComposer->>AccountAggregator: Update accounts

BalanceAggregator->>DTOPersist: Partial DTO

CardAggregator->>DTOPersist: Partial DTO

AccountAggregator->>DTOPersist: Partial DTO

end

DTOPersist->>Redis: Cache updated CustomerPosition

DTOPersist->>Oracle: Persist updated CustomerPosition

alt Generate new event

DTOPersist->>EventGenerator: Trigger event (e.g., CustomerPositionUpdated)

EventGenerator->>Kafka: Publish new event

end