***Handling the lack of isolation***

The I in ACID signifies isolation, ensuring that concurrent transactions yield results consistent with serial execution. However, sagas lack this property, potentially leading to anomalies where transactions interfere with each other's data. Despite this, developers often prioritize performance over isolation. While databases allow specifying isolation levels, real-world transactions often deviate from ideal ACID standards. To mitigate isolation issues in sagas, various countermeasures are employed, either implementing isolation at the application level or reducing business risks associated with isolation shortcomings. These strategies aim to eliminate anomalies or minimize their impact on business logic.

***Overview of anomalies***

The lack of isolation in sagas can lead to three types of anomalies:

* **Lost updates:** One saga overwrites changes made by another saga without considering them.
* **Dirty reads:** A saga reads data that is being updated by another saga before the update is completed.
* **Fuzzy/nonrepeatable reads:** Different steps of a saga read the same data and get different results due to updates made by other sagas.

Among these, lost updates and dirty reads are the most common and challenging. Lost updates occur when one saga ignores updates made by another, potentially leading to incorrect actions like shipping a canceled order. Dirty reads happen when a saga reads data that is in the process of being updated by another saga, potentially causing risks like allowing orders that exceed credit limits.

The saga transaction model lacks isolation, which can lead to anomalies affecting application behavior. Developers must design sagas to either prevent these anomalies or minimize their impact on business operations. One strategy is using \*\_PENDING states for Orders, signaling that the Order is being updated by a saga. This approach aligns with the concept of semantic lock countermeasures, as described in a 1998 paper by Lars Frank and Torben U. Zahle. This paper outlines several countermeasures for handling isolation issues, including semantic locks, commutative updates, pessimistic views, rereading values, version files, and dynamic concurrency mechanisms. The structure of a saga, as defined in the paper, includes compensatable transactions, pivot transactions (the decision point), and retriable transactions. These elements help in understanding and implementing effective countermeasures.

**Semantic Lock:**

An application-level lock. Ensures exclusive access to shared resources during transaction execution, preventing conflicting updates by multiple sagas.

**Commutative Updates:** Design update operations to be executable in any order.

It Allows for flexible execution order of transactions without affecting the final outcome, reducing the likelihood of conflicts.

**Pessimistic View:** Reorder the steps of a saga to minimize business risk.

Prioritize transaction ordering to minimize the impact of anomalies on business operations, ensuring critical actions are completed before less critical ones.

**Reread Value:** Prevent dirty writes by rereading data to verify it's unchanged before overwriting it.

It Validates data consistency before making updates, reducing the risk of overwriting changes made by concurrent transactions.

**Version File:** Record updates to a record to facilitate reordering.

Tracks changes made by transactions, allowing for adjustments in execution order to mitigate conflicts and ensure data consistency.

**By Value:**

Dynamically select the concurrency mechanism based on each request's business risk.

Adapts concurrency control mechanisms based on the criticality of the transaction, ensuring that high-risk operations receive stronger isolation guarantees.

In addition to these countermeasures, understanding the structure of a saga is crucial. As outlined in the aforementioned paper, a saga consists of three types of transactions:

**Compensatable transactions:** Transactions that can potentially be rolled back using a compensating transaction.

**Pivot transaction:** The decision point in a saga. If the pivot transaction commits, the saga proceeds until completion. It can be a transaction that's neither compensatable nor retriable, or the last compensatable transaction, or the first retriable transaction.

**Retriable transactions:** Transactions following the pivot transaction that are guaranteed to succeed, contributing to the saga's successful execution.

