View anti pattern

The Open Session in View (OSIV) pattern is an anti-pattern that allows Hibernate sessions (or JPA EntityManager sessions) to remain open beyond the service layer and into the view layer of a web application. This is typically done to facilitate lazy loading of associations in the view layer, ensuring that the data required for rendering views is available even after the service layer has completed its work and committed the transaction.

Here's how the OSIV pattern works:

1. **Open and Close Session in View Layer**:
   In the view layer (e.g., in a controller in a web application), the Hibernate session or EntityManager session is opened at the beginning of the request handling process and closed after the view is rendered. This ensures that the session remains open while the view is being rendered, allowing lazy-loaded associations to be initialized if needed.

2. **Service Layer and Database Transaction**:
   The view layer typically interacts with the service layer to perform business logic and access the database. When the service layer is called, it opens a database transaction to execute the necessary operations. Once the service layer has completed its work and the transaction is committed, the database transaction is closed.

3. **Lazy Loading in View Layer**:
   Because the Hibernate session remains open in the view layer, lazily fetched associations can be initialized when accessed in the view templates or UI components. This ensures that the required data is loaded on-demand, based on the needs of the view being rendered.

However, there are several drawbacks to using the OSIV pattern:

1. **Increased Database Load**:
   As mentioned in the explanation, keeping the Hibernate session open in the view layer can lead to increased load on the database server. Each SQL statement triggered by the view layer is executed in auto-commit mode, leading to additional database transactions and increased overhead.

2. **Transaction Management Complexity**:
   Managing transactions and session state across multiple layers of the application can introduce complexity and make it harder to reason about the application's behavior. This can lead to issues such as resource leaks, inconsistent transaction boundaries, and difficulties in troubleshooting.

3. **Potential for Performance Issues**:
   The OSIV pattern can lead to performance issues, especially in high-traffic or

concurrent environments, due to the increased database load and potential for resource contention. It may also hinder scalability and degrade overall application performance.

Due to these drawbacks, the OSIV pattern is considered an anti-pattern in modern application development. Instead, best practices recommend separating concerns and keeping transaction management within the service layer, while using other techniques such as eager fetching, DTO projection, or data prefetching to optimize data retrieval and avoid lazy loading in the view layer.