Chapter 15: *Invalidate caches on database*

Invalidating caches on a database write is a strategy used to ensure that the cached data remains consistent with the actual data stored in the database. Here's a breakdown of why and how this process works:

1. The Problem of Stale Cache Data

- When data in the database is updated, any related cached data can become **stale** or **outdated** if it's not updated at the same time.
- For example, if you update a user's profile information in the database but don't update the cache, anyone accessing the cache would see outdated data until it's refreshed.

2. Why Invalidate the Cache on a Database Write?

- **Consistency**: Invalidation ensures that the cache doesn't hold old data that no longer matches the database.
- **Prevention of Conflicts**: When the cache and database are out of sync, applications might make decisions based on incorrect or outdated data, leading to errors or unexpected behavior.

3. How Cache Invalidation Works

- When a **write operation** (insert, update, or delete) occurs in the database, the application invalidates (removes) the corresponding entry in the cache.
- This can be done by deleting the specific cached item or by marking it as invalid.
- Once invalidated, future read requests will bypass the cache and pull the updated data from the database, re-populating the cache with the latest value.

4. Steps for Cache Invalidation on Write

1. **Write to the Database**: First, the application writes or updates the new data in the database.

2. **Invalidate the Cache**: Immediately after the write, the application removes or invalidates the related entry in the cache.
3. **Subsequent Reads Fetch Fresh Data**: The next time an application requests this data, it will check the cache, find it missing (due to invalidation), and query the database for the latest data. This fresh data is then cached again for future requests.

5. Example

- Suppose we have a cache storing user profile data, with each entry keyed by the user's ID.
- When a user updates their profile, the application:
  - Writes the new profile data to the database.
  - Invalidates the cache entry for that user's ID.
- This way, the next time a user's profile data is requested, the application pulls the latest information from the database and caches it again.

6. Benefits and Considerations

- **Benefits**: This approach keeps cached data consistent with the database and reduces the likelihood of errors due to outdated information.
- **Considerations**: Cache invalidation can add complexity, and in distributed systems, ensuring immediate cache consistency across servers may require additional mechanisms, like event-driven invalidation.

In summary, cache invalidation on database write is a way to ensure that the cache reflects the latest database changes, keeping the two sources in sync and preventing outdated data from being served to users.