DWA_07.4 Knowledge Check_DWA7

1. Which were the three best abstractions, and why?
    - **<u>The view level</u>** is the highest level of abstraction and deals with how the data should be presented to the user. It defines the user's view of the database and provides a security mechanism by hiding sensitive information from unauthorized users.

    - **<u>The conceptual level</u>** is the middle level of abstraction and defines how data is organized and structured in the database. It provides a conceptual view of the database that is independent of any specific implementation.

    - **<u>The physical level</u>** is the lowest level of abstraction and deals with how data is stored on disk. It defines how data is physically arranged in the database and how it can be accessed by the operating system

---

2. Which were the three worst abstractions, and why?

    - **Overly complex abstractions:** When an abstraction becomes overly complex, it can be challenging to understand and modify. Over time, such abstractions may accumulate unnecessary conditionals and interleave vaguely associated ideas, making the codebase harder to maintain.

    - **Poorly designed abstractions:** Abstractions that were poorly designed from the start can lead to confusion and obfuscation. If the code no longer represents a single, common abstraction and instead consists of condition-laden procedures, it becomes harder to understand and easier to break.

    - **Inappropriate abstractions:** Sometimes, an abstraction that was initially suitable for a project may become inappropriate as the project evolves. Recognizing when an existing abstraction is no longer suitable is crucial. Inappropriate or misused abstractions can add unnecessary complexity and hinder maintenance efforts2.

3. How can The three worst abstractions be improved via SOLID principles.

- The Single-responsibility Principle (SRP) states that a class should have only one reason to change. This means that each class should have only one responsibility or job to do 1. By following this principle, developers can create classes that are easier to understand and maintain.
- The Open-closed Principle (OCP) states that classes should be open for extension but closed for modification 1. This means that developers should be able to add new functionality to a class without changing its existing code. By following this principle, developers can create software that is more flexible and easier to maintain.
- The Dependency Inversion Principle (DIP) states that high-level modules should not depend on low-level modules. Instead, both should depend on abstractions 1. This means that developers should create interfaces or abstract classes that define the behavior of their code. By following this principle, developers can create software that is more modular and easier to test.