Key Design Patterns:

1. **Strategy Pattern** for changing the processor's mode dynamically.
2. **Factory Method Pattern** for database connection management.
3. **Interface Segregation and Dependency Inversion Principles** to ensure flexibility and extendability.

UML Class Diagram Overview

The diagram will include the following classes and interfaces:

1. **DataProcessor (Abstract Class)**
    - **Attributes:**
        - mode: ProcessingMode
        - database: Database
    - **Methods:**
        - configure(mode: ModeIdentifier, db: DatabaseIdentifier): void
        - process(data: DataPoint): void
2. **ProcessingMode (Interface)**
    - **Methods:**
        - process(data: DataPoint, db: Database): void
3. **DumpMode (Class)**
    - Implements ProcessingMode
    - **Methods:**
        - process(data: DataPoint, db: Database): void
4. **PassthroughMode (Class)**
    - Implements ProcessingMode
    - **Methods:**
        - process(data: DataPoint, db: Database): void
5. **ValidateMode (Class)**
    - Implements ProcessingMode
    - **Methods:**
        - process(data: DataPoint, db: Database): void

6. **Database (Interface)**
    - **Methods:**
        - connect(): void
        - insert(data: DataPoint): void
        - validate(data: DataPoint): boolean
7. **PostgresDatabase (Class)**
    - Implements Database
    - **Methods:**
        - connect(): void
        - insert(data: DataPoint): void
        - validate(data: DataPoint): boolean
8. **RedisDatabase (Class)**
    - Implements Database
    - **Methods:**
        - connect(): void
        - insert(data: DataPoint): void
        - validate(data: DataPoint): boolean
9. **ElasticDatabase (Class)**
    - Implements Database
    - **Methods:**
        - connect(): void
        - insert(data: DataPoint): void
        - validate(data: DataPoint): boolean

Description of Core Interactions:

- **DataProcessor:**
    - Acts as the central controller of the system.
    - The configure method updates the mode and database fields based on the provided identifiers.

    - The process method delegates behavior to the currently configured mode and database.

- **ProcessingMode Implementations:**
    - Each mode class (e.g., DumpMode, PassthroughMode, ValidateMode) implements the specific logic for handling data based on the current mode.
- **Database Implementations:**
    - Each database class (e.g., PostgresDatabase, RedisDatabase, ElasticDatabase) implements its own logic for connecting, inserting, and validating data.