## 1. ドメインロジックパターン(Domain Logic Patterns)

- トランザクションスクリプト(Transaction Script)
　ビジネスロジックを 1 つ 1 つ個別のプロシージャで実装する。

- ドメインモデル(Domain Model)
　ビジネスロジックをデータと振る舞いを結び付けたオブジェクトの集合で実装する。

- テーブルモジュール(Table Module)
　テーブル、ビューの行全体の視点からビジネスロジックを実装する。

- サービスレイヤ(Service Layer)
　粗粒度のサービスを提供する層を生成して、アプリケーション境界を作成する。

## 2. データソースアーキテクチャパターン(Data Source Architectural Patterns)

- テーブルデータゲートウェイ(Table Data Gateway)
　テーブルに対するゲートウェイの提供する。

- ロウデータゲートウェイ(Row Data Gateway)
　ロウに対するゲートウェイの提供する。

- アクティブレコード(Active Record)
　ロウをラップして、ロウ操作とドメインロジックを提供する。

- データマッパ(Data Mapper)
　オブジェクトとデータベースのデータをマッピングし、これらとマッパ自身を分離する。

## 3. O-R 振る舞いパターン(Object-Relational Behavioral Patterns)

- ユニットオブワーク(Unit of Work)
　同時並行処理の問題を解決するために、ビジネストランザクション内で影響を受けるオブジェクトを記録し、更新内容の調停を行なう。

- アイデンティティマップ(Identity Map)
　ビジネストランザクションで読み出されたオブジェクトをマップに登録し、2 回目以降のアクセスに対してマップの内容を返送することにより 1 度だけ読み出しが行われることを保証する。

- 遅延ロード(Lazy Load)
　オブジェクトに最低限のデータのみを持たせ、それ以外のデータは必要になった時点で取りに行く。

## 4. O-R 構造パターン(Object-Relational Structural Patterns)

- アイデンティティフィールド(Identity Field)
　メモリ上のオブジェクトとデータベースを関連付けるために、オブジェクトにデータベースの ID を保持する。

- 外部キーマッピング(Foreign Key Mapping)
　オブジェクト間の関連をデータベーステーブルの外部キーによる関連にマップする。

- 関連テーブルマッピング(Association Table Mapping)
　多対多の関連を持つテーブル間の関連を、外部キーを格納したテーブルを追加することで解決する。

- 従属マッピング(Dependent Mapping)
　親クラスが、子クラスのデータベースマッピングも一緒に行う。

- エンベデッドバリュー(Embedded Value)
　オブジェクトを別のオブジェクトのテーブルのフィールドにマップする。

- 直列化 LOB(Serialized LOB)
　オブジェクトのグラフを直列化して 1 つのテーブルのフィールドに格納する。

- 単一テーブル継承(Single Table Inheritance)
　継承ツリー内の全クラスのフィールドを持つ 1 つのテーブルに、継承ツリー内のオブジェクトをマップして継承を表現する。

- クラステーブル継承(Class Table Inheritance)
　継承ツリー内の各クラスに 1 つのテーブルを対応させて継承を表現する。

- 具象テーブル継承(Concrete Table Inheritance)
　継承ツリー内の各具象クラスに 1 つのテーブルを対応させて継承を表現する。

- マッパ継承(Inheritance Mappers)
　継承マッパ自身を継承を使って実装する。

## 5. O-R メタデータマッピングパターン (Object-Relational Metadata Mapping Patterns)

- メタデータマッピング(Metadata Mapping)
　O-R マッピングの情報をメタデータに保存する。

- クエリオブジェクト(Query Object)
　データベースに対するクエリをオブジェクトで表現する。

- リポジトリ(Repository)
　ドメインオブジェクトに対してコレクションのようにアクセスできるインターフェースを提供して、ドメインとマッピング層との間の仲介をする。

## 6. Web プレゼンテーションパターン (Web Presentation Patterns)

- モデルビューコントローラ(Model View Controller)
　ユーザインターフェースとのやりとりを、3 つの役割に分割する。

- ページコントローラ(Page Controller)
　Web サイトのページやアクションリクエストを受け付けるオブジェクト。

- フロントコントローラ(Front Controller)
　Web サイトへのリクエストを 1 つのコントローラで処理する。

- テンプレートビュー(Template View)
　HTML の中にマーカを埋め込むことで、HTML の中に動的に情報を差し込む。

- 変換ビュー(Transform View)
　ドメインデータを工レメントごとに変換して、HTML を組み立てる。

- 2 段階ビュー(Two Step View)
　まず論理的なページを生成した後、それを HTML に変換する。

- アプリケーションコントローラ(Application Controller)
　アプリケーションの画面の遷移を制御する。

## 7. 分散パターン(Distribution Patterns)

- リモートファサード(Remote Facade)
　細粒度のオブジェクトの上に粗粒度の面をかぶせて、ネットワークを通した呼び出し効率を向上させる。

- データ転送オブジェクト(Data Transfer Object)
　複数の処理の間でまとめてデータを運ぶオブジェクトを作成してメソッド呼び出しの回数を削減する。

## 8. オフライン同時並行パターン (Offline Concurrency Pattenrs)

- オフライン楽観的ロック(Optimistic Offline Lock)
　同時並行で処理されるビジネストランザクション間の競合を検出してロールバックする仕組み。

- オフライン悲観的ロック(Pessimistic Offline Lock)
　同時並行で処理されるビジネストランザクション間の競合を、1 時点では 1 ビジネストランザクションしかデータにアクセスできないようにして、防ぐ仕組み。

- 粗粒度ロック(Coarse-Grained Lock)
　関連するオブジェクト全体を 1 つのロックオブジェクトでロックする。

- 暗黙ロック(Implicit Lock)
　フレームワークやレイヤスーパータイプ側で、自動的にオフラインロックを獲得する。

## 9. セッションステートパターン (Session State Pattenrs)

- クライアントセッションステート(Client Session State)

クライアント側にセッションステートを保持する。

- サーバセッションステート(Server Session State)
セッションステートをサーバ側に直列化された状態で保持する。

- データベースセッションステート(Database Session State)
セッションステートをデータベース上のコミットされたデータとして保持する。

## 10.基本パターン(Base Patterns)

- ゲートウェイ(Gateway)
外部のリソース、システムへのアクセスをカプセル化する。

- マッパ(Mapper)
サブシステム間のデータのやりとりを仲介し、サブシステムの独立性を保つ。

- レイヤスーパータイプ(Layer Supertype)
ある層を構成する全てのタイプから、共通のスーパータイプを抽出する。

- 分離インターフェース(Separated Interface)
実装からインターフェースを別のパッケージに分離する。

- レジストリ(Registry)
共通に使用されるオブジェクトやサービスを問い合わせるためのオブジェクト。

- 値オブジェクト(Value Object)
金銭や日付範囲などを表わす小さなオブジェクトで、同値性を参照の一致で判断できないもの。

- マネー(Money)
エンタープライズアプリケーションで頻繁に使用される、金額をオブジェクトにカプセル化して統一的に扱えるようにする。

- スペシャルケース(Special Case)
特定のケースに対応するために、継承クラスを導出する。

- プラグイン(Plugin)
クラス間のリンクをコンパイル時ではなく、実行時に設定を読んで解決する。

- サービススタブ(Service Stub)
サービスへの依存を排除して、テストを容易にする。

- レコードセット(Record Set)
表形式のデータのメモリ上での表現。