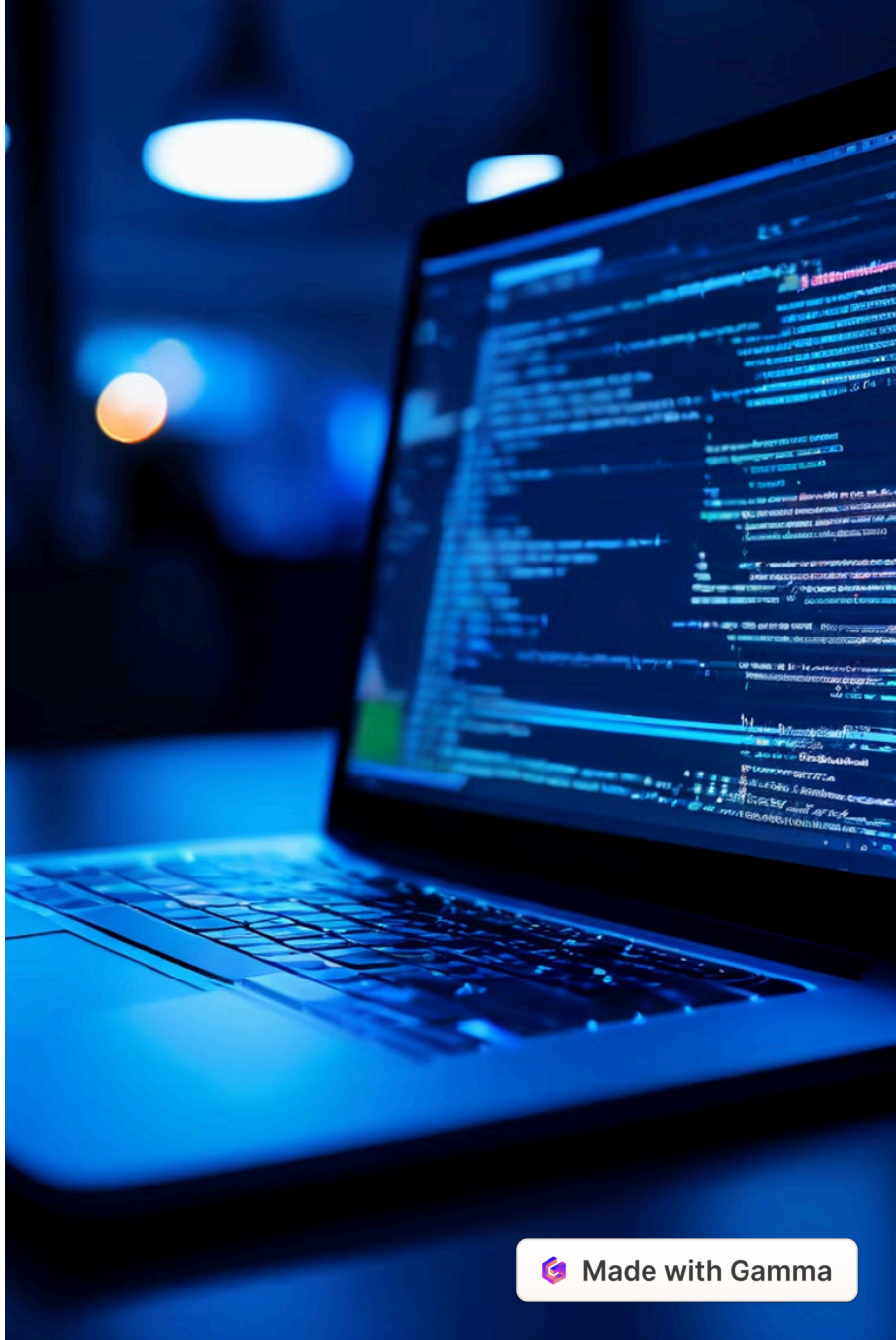
Entity Framework Mapping Strategies

Entity Framework provides a powerful Object-Relational Mapping (ORM) solution for .NET developers, allowing you to work with database tables using objects. One key aspect of Entity Framework is mapping strategies, which determine how your object model relates to your database schema.

by Fawzii Elfaramaawy



Overview of Mapping Strategies

Entity Framework offers three primary mapping strategies: Table-Per-Hierarchy (TPH), Table-Per-Type (TPT), and Table-Per-Concrete-Type (TPC).

1 TPH

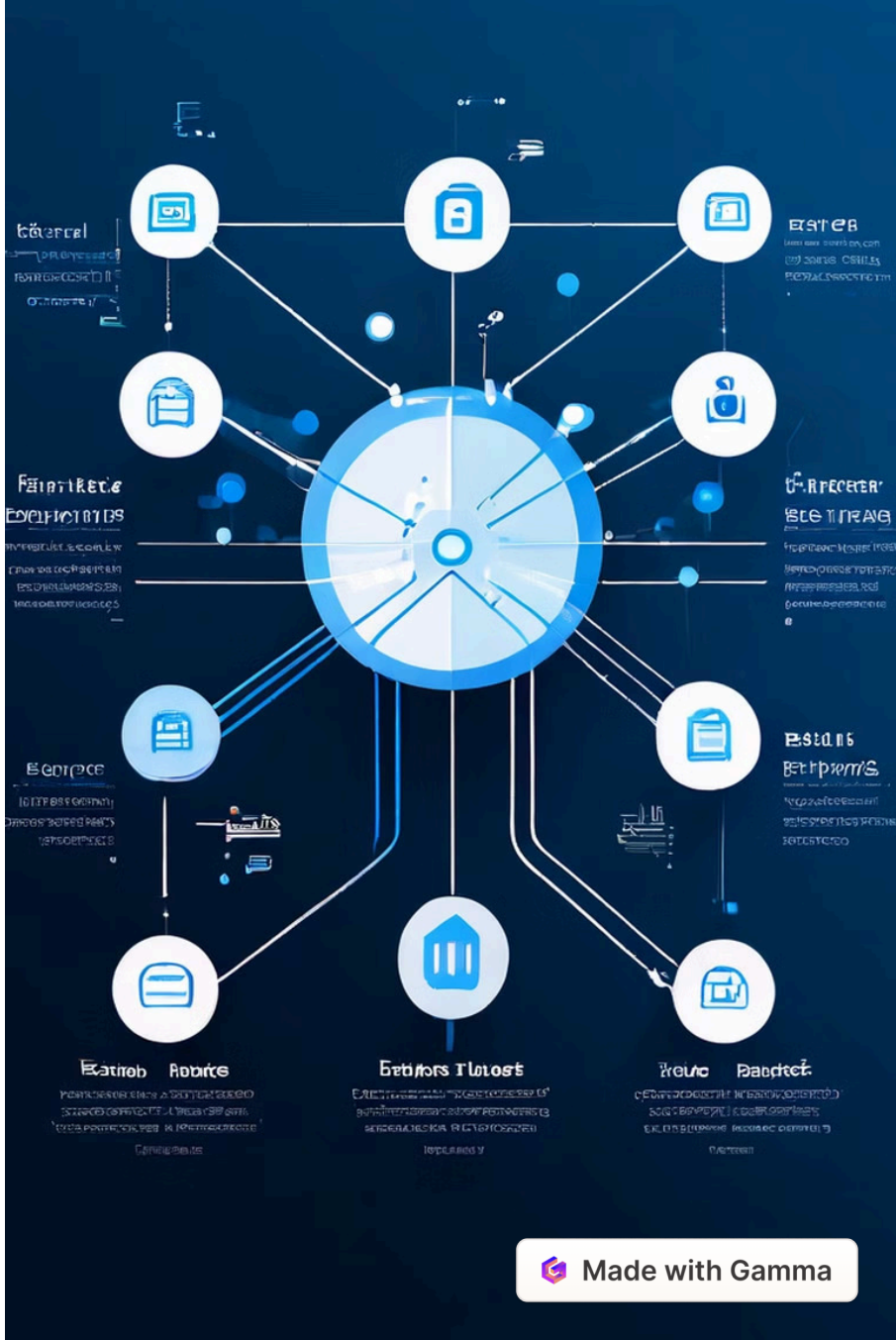
Maps all entities in a hierarchy to a single table, using inheritance in the database.

2 TPT

Creates a separate table for each concrete type in a hierarchy, using inheritance in the database.

3 TPC

Creates a separate table for each concrete type, with no inheritance in the database.



Mapping Strategies in Entity Framework

Choosing the right mapping strategy depends on your application's specific needs and the complexity of your data model. Each strategy has its advantages and disadvantages.

TPH

Simplest approach with one table for all types.

Good for small hierarchies with similar properties.

TPT

Better for larger hierarchies with variations in properties.

Supports more flexible queries and data access.

TPC

Provides the most control over database structure.

Suitable for highly specialized and optimized queries.

Table-Per-Hierarchy (TPH) Mapping

In TPH, all entities in a hierarchy are mapped to a single table. The table contains columns for all properties of the base class and derived classes.

1

Advantages

Simplest and most efficient for simple hierarchies.

2

Disadvantages

Limited control over database structure.

3

Use Cases

Simple inheritance structures with minimal variations.

Table-Per-Type (TPT) Mapping

TPT maps each concrete type in a hierarchy to its own separate table. This allows for more flexibility in data storage and querying.

Advantages

Better for large hierarchies with varying properties.

Disadvantages

More complex to implement than TPH.

Use Cases

Complex hierarchies with significant differences between types.

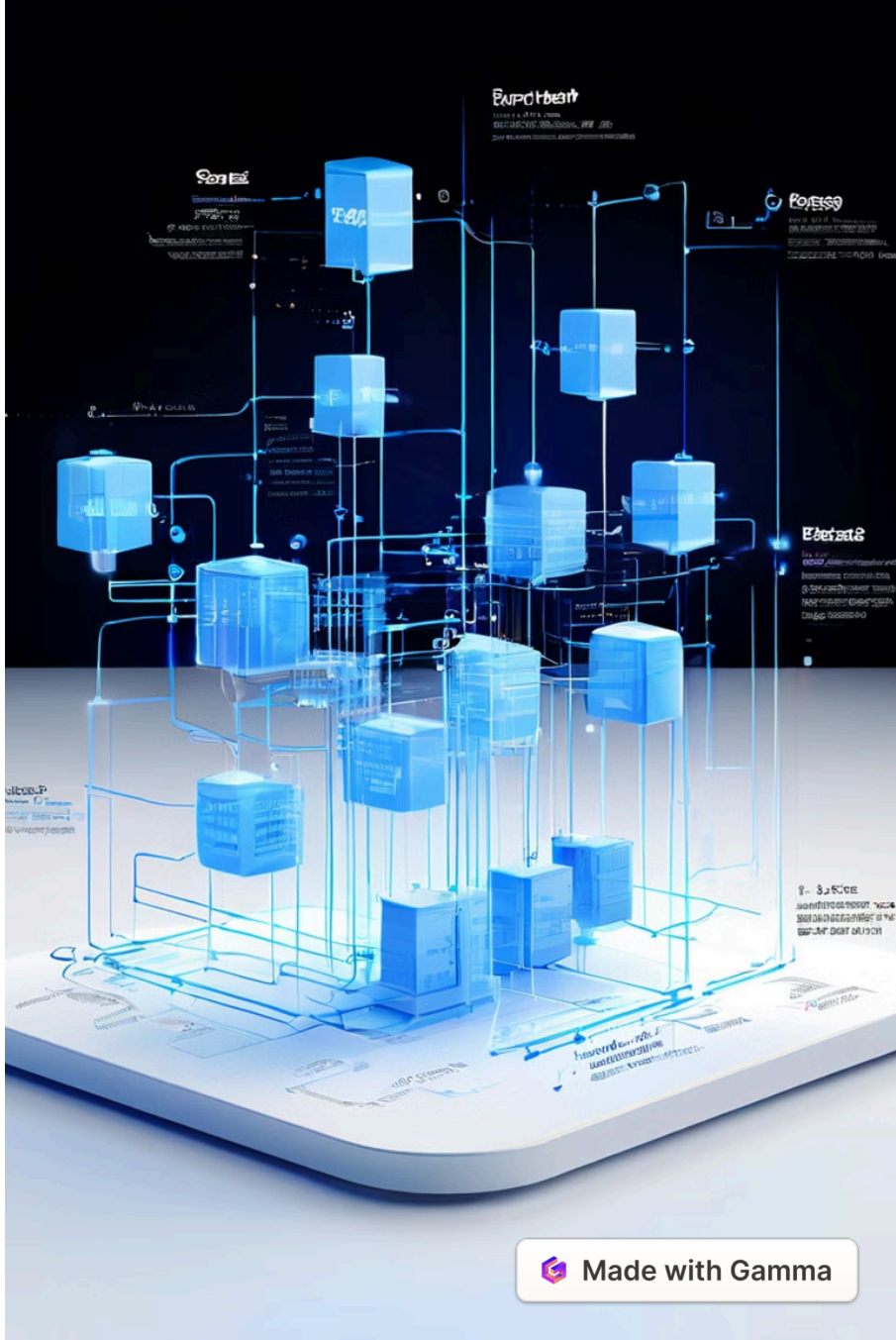
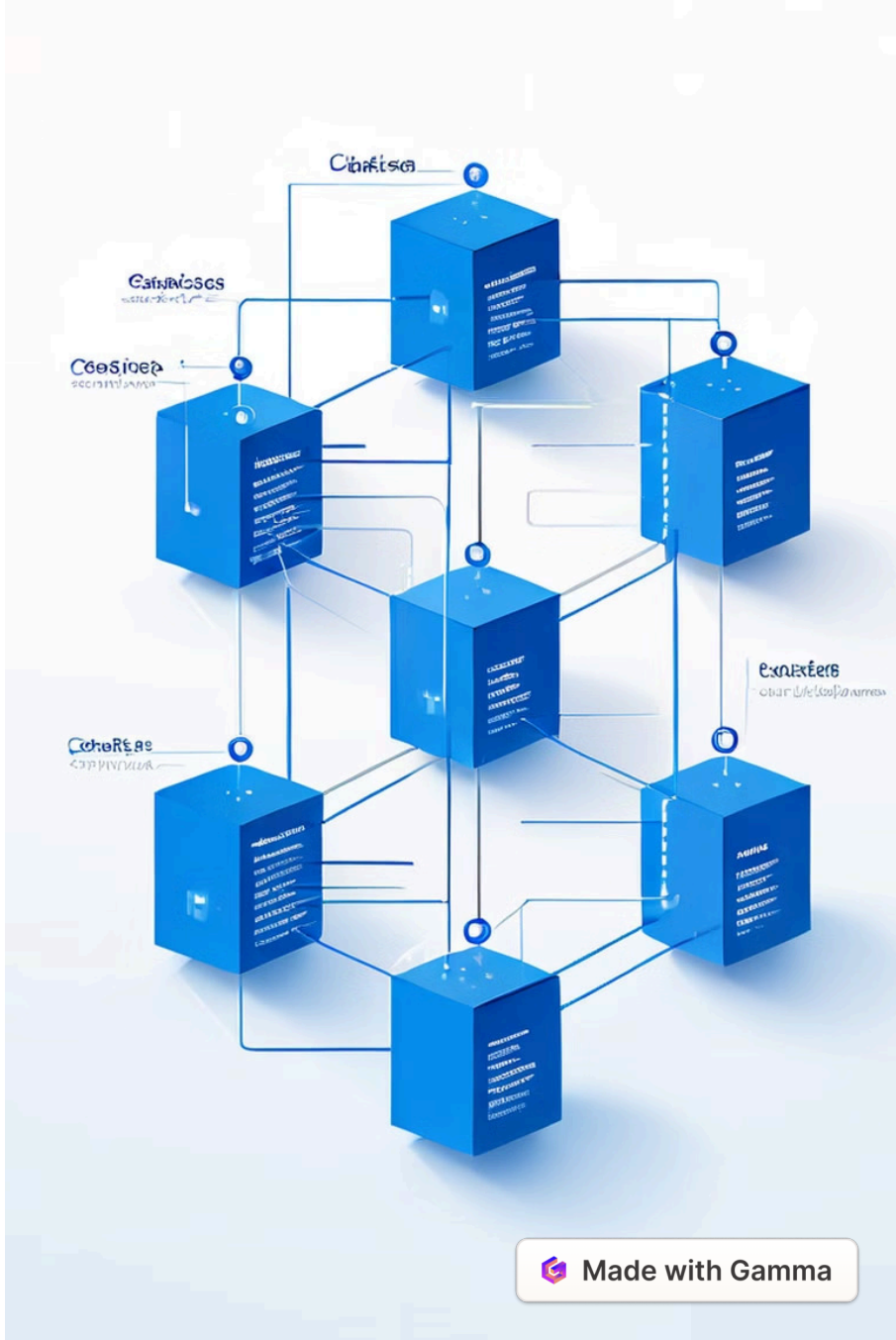
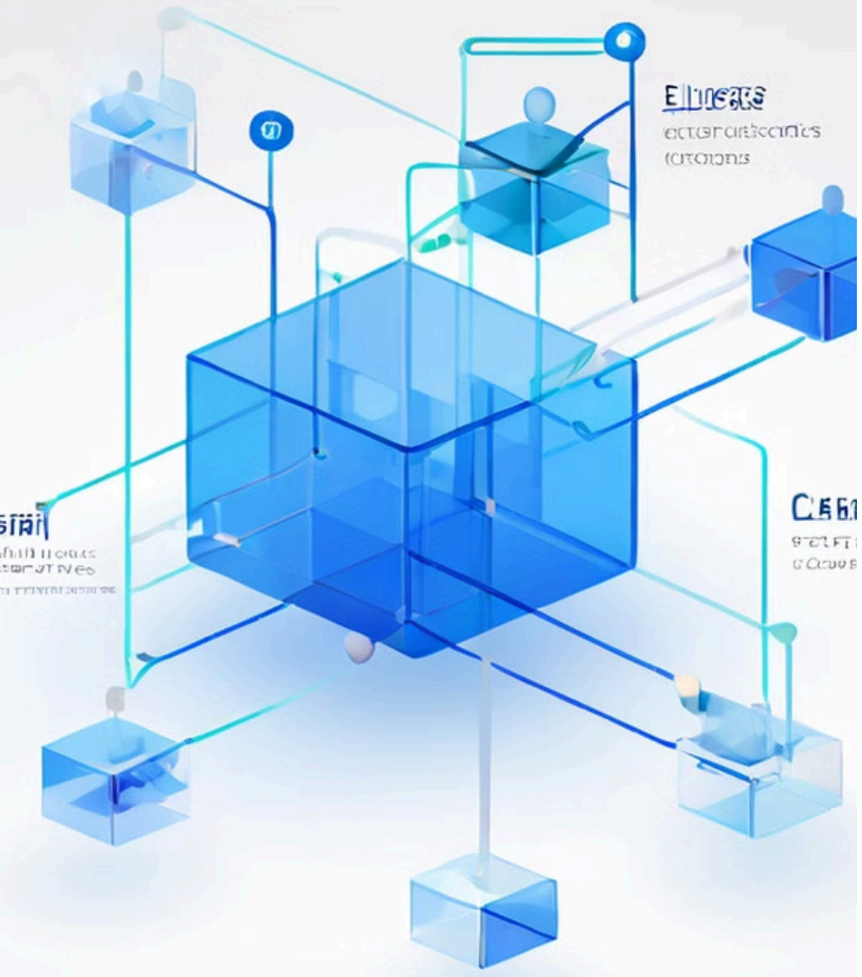


Table-Per-Concrete-Type (TPC) Mapping

TPC creates a separate table for each concrete type, with no inheritance in the database. This strategy provides the most control over the database structure.

Advantages	Most control over database structure.	Optimized queries and data access.
Disadvantages	Most complex to implement.	Limited inheritance support.
Use Cases	Highly specialized data models with performance requirements.	Large and complex data sets.





Advantages and Disadvantages of Mapping Strategies

The best strategy depends on your specific needs and requirements.

TPH

Simple implementation, ideal for small hierarchies.

TPT

Flexibility for complex hierarchies, better for varying properties.

TPC

Fine-grained control, optimized for performance.

Demonstration with Code Examples

Understanding the nuances of each strategy and how they work in practice is crucial for choosing the right approach for your application. Let's explore these strategies with code examples.