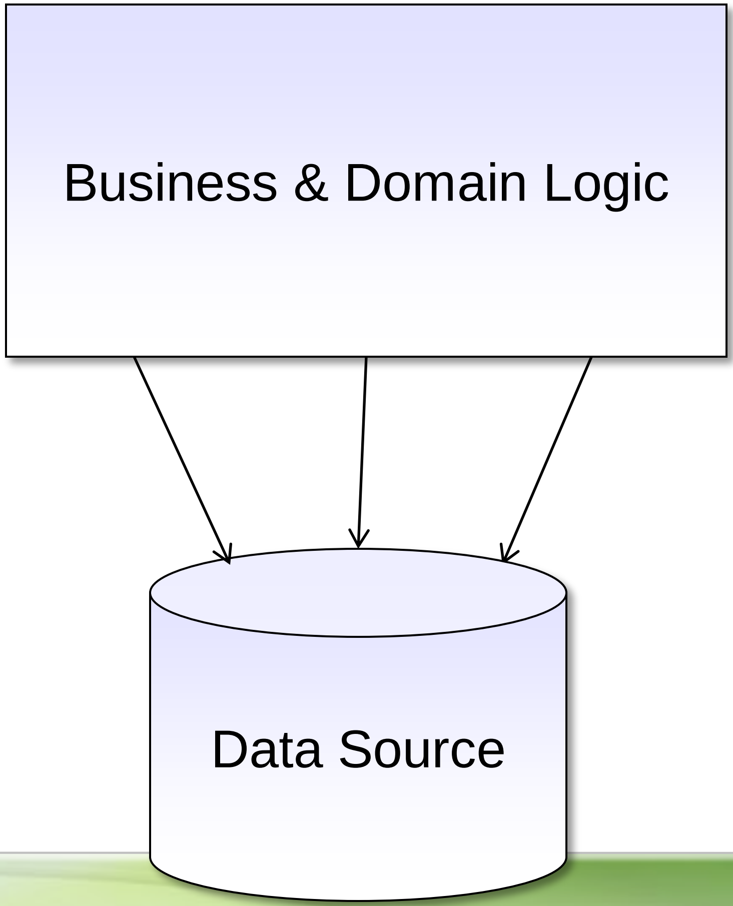
Repository Pattern

A pattern for data access

Why?

Repository Pattern

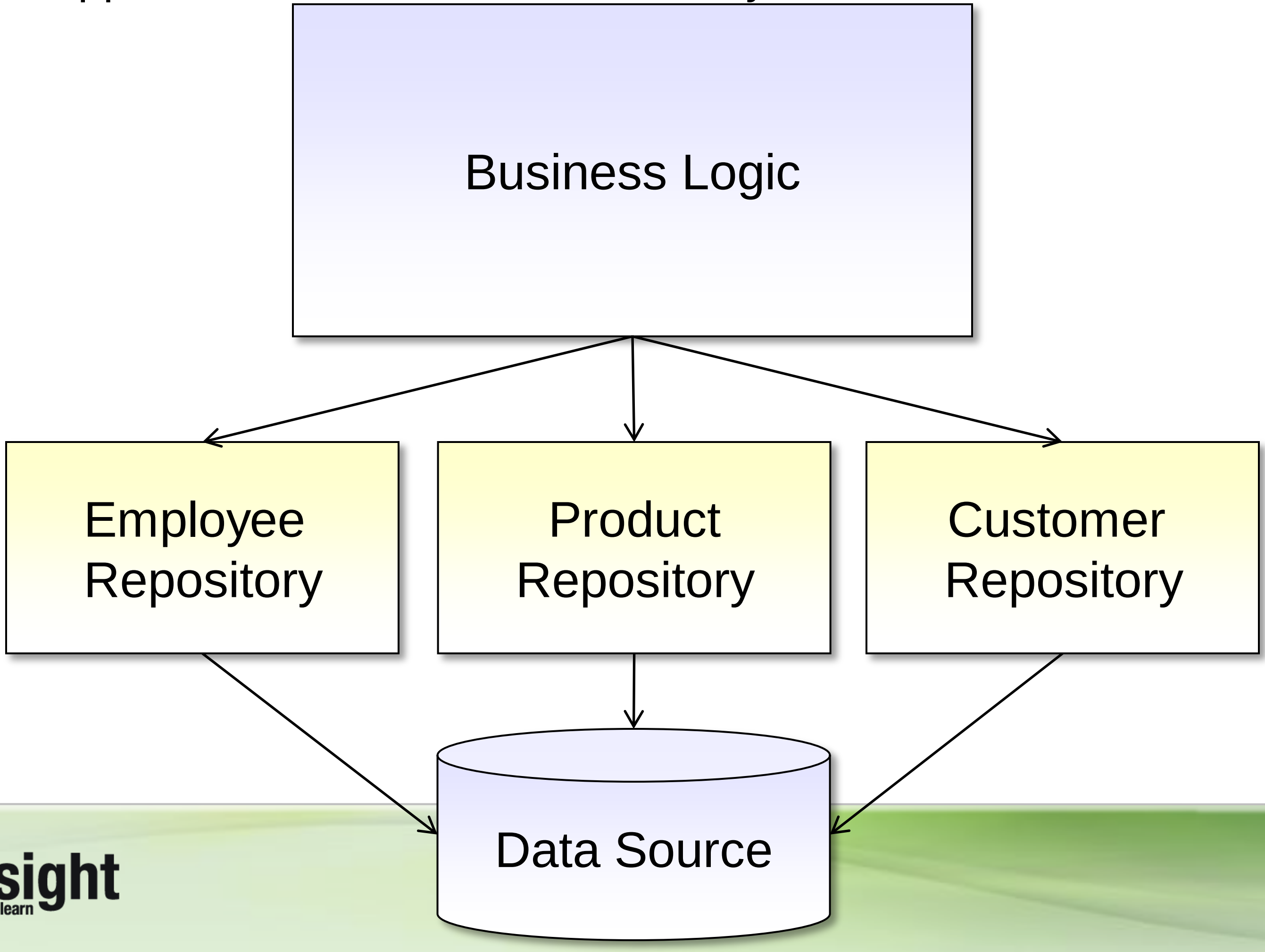
- **Separate business code from data access**
  - Separation of concerns
  - Testability



Intent

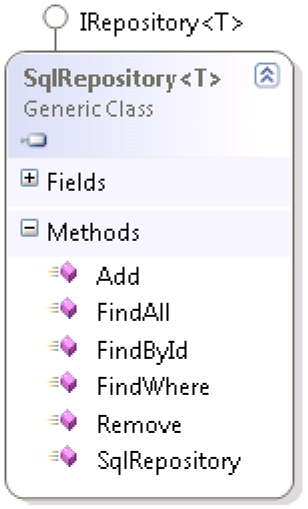
Repository Pattern

- **Encapsulate data access**
  - Data appears to live inside an in-memory collection



Demo

Repository Pattern


IRepository<T>

SqlRepository<T>
Generic Class

Fields

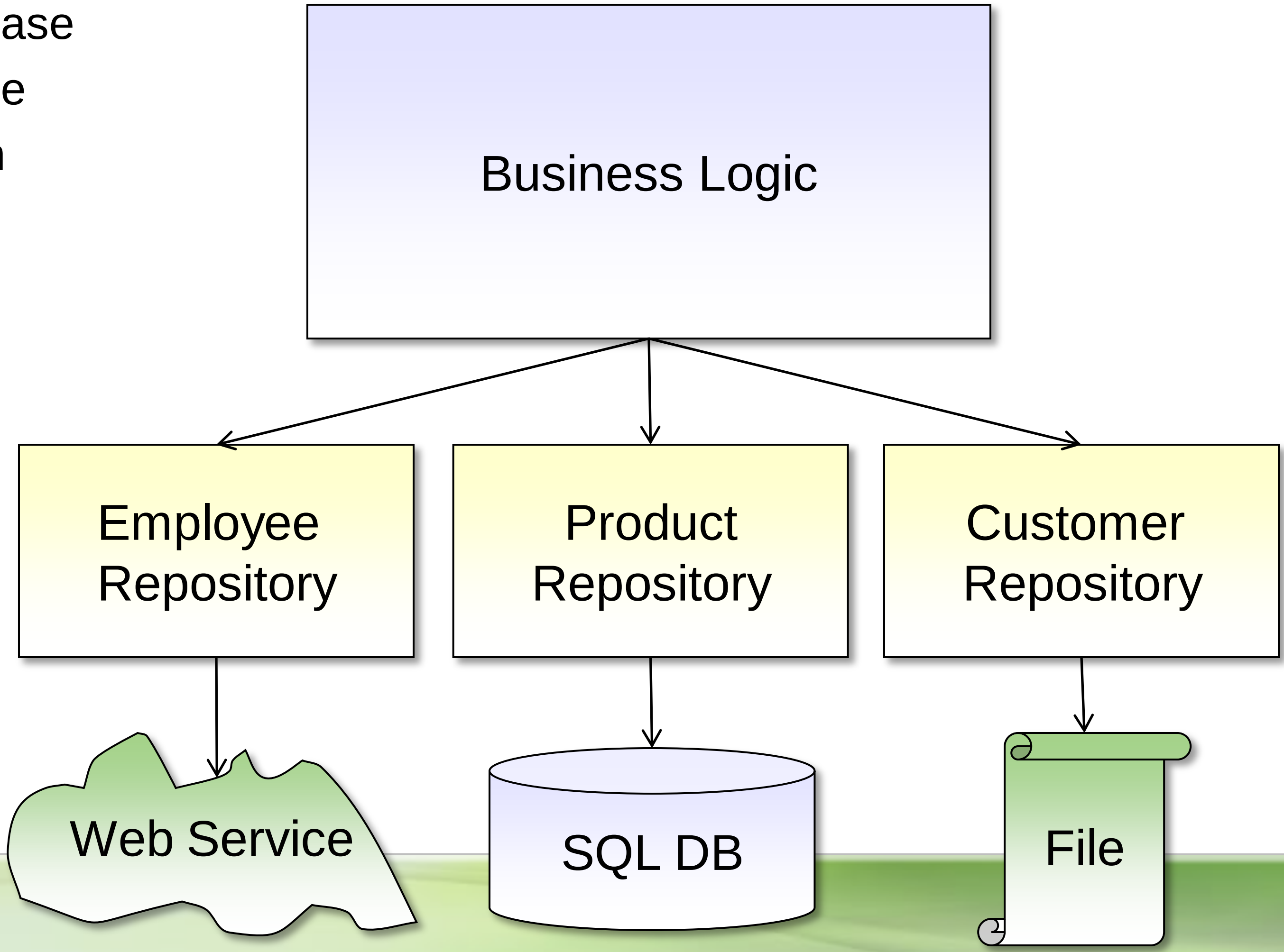
Methods
- Add
- FindAll
- FindById
- FindWhere
- Remove
- SqlRepository

```
var model = _repository.FindAll()
                       .Include("TimeCards")
                       .OrderBy(e => e.HireDate);
```

Applicability

Repository Pattern

- **Anytime you need data persistence**
  - SQL Database
  - Web service
  - File system



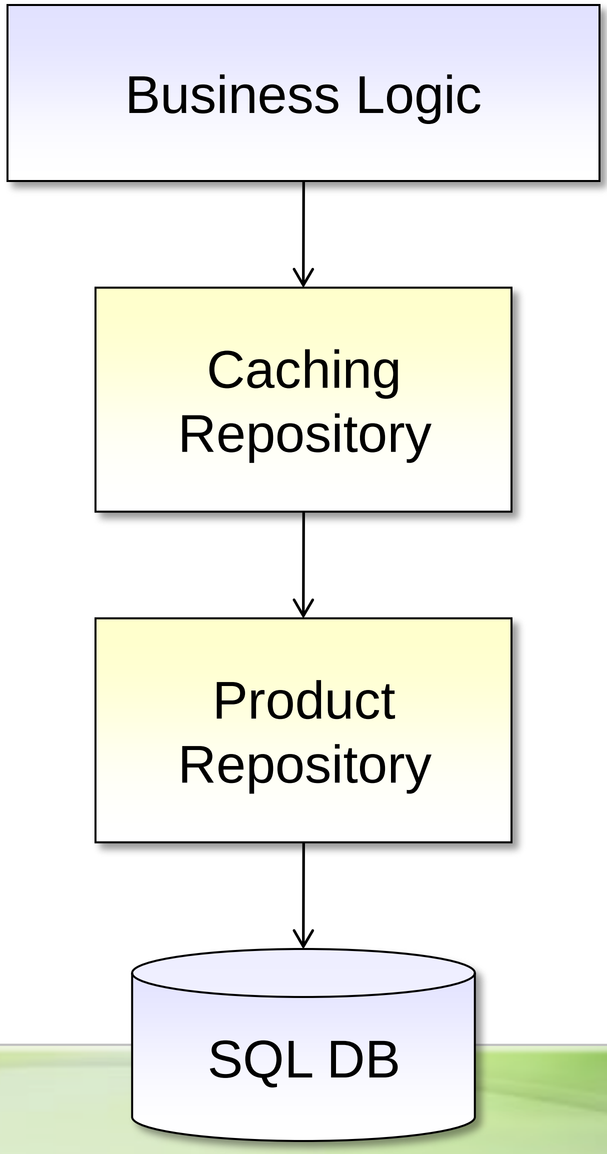
Consequences

Repository Pattern

- **Increased level of abstraction**
  - More classes, less duplicated code
  - Maintainability, flexibility, testability
- **Further away from data**
  - Shielded from infrastructure
  - Harder to optimize

Related Patterns

- Unit of Work
- Specification
- Identity Map
- Decorator



Summary

Repository Pattern

- **Keep business logic free of data access code**
- **Testability, maintainability**
- **Generic repositories**
  - IRepository<T>