Data Programming beyond ADO.NET ObjectSpaces and Neo

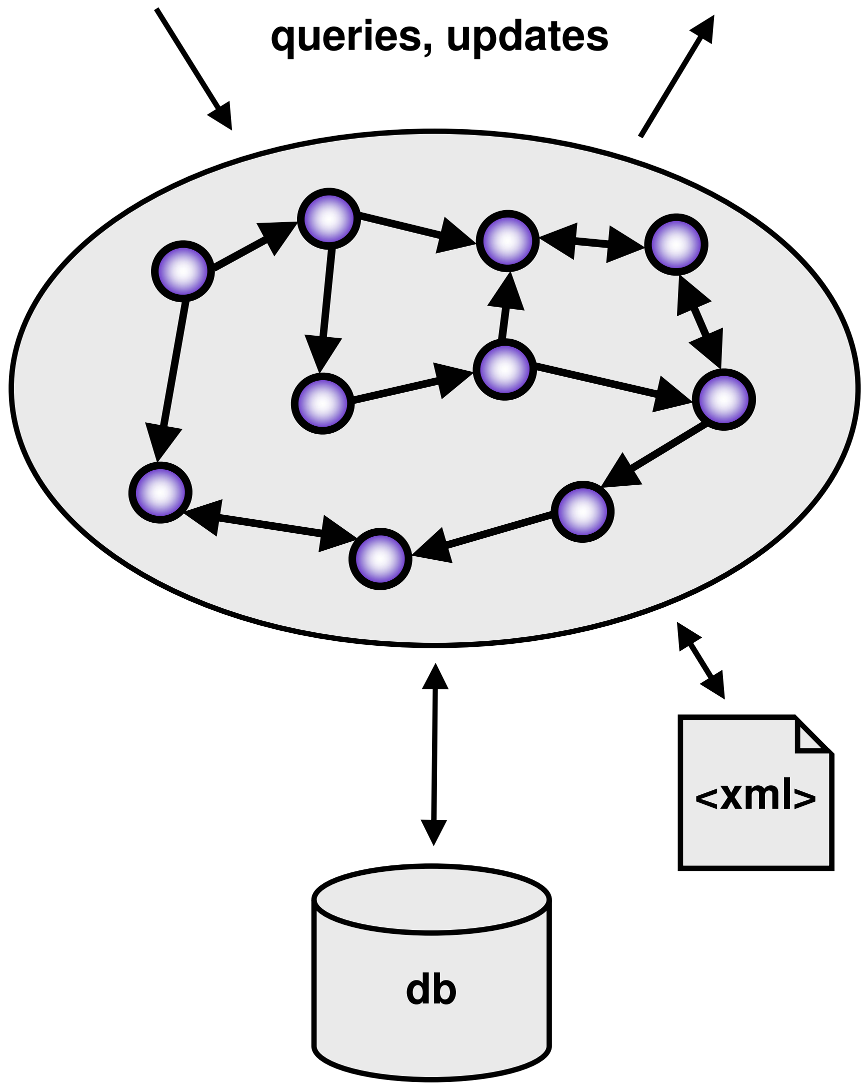
Erik Dörnenburg
ThoughtWorks UK

O/R Mapping Frameworks

- When do you need O/R mapping frameworks?
  - Domain-Driven Design
  - Strong business logic layer
- Access and manipulate data in terms of domain objects, e.g. Customer, Order, Address
- Declarative mapping between object model and relational tables
- Separation between domain model (business logic) and infrastructure code (data access logic)
- Higher level layer than ADO.NET DataReaders

ObjectSpaces

Architecture



Application code

ObjectSpace & Domain Objects

Mapping files
- rsd/osd/msd

Domain Objects

- Be as transparent as possible
- POCOs – Plain Old CLR Objects
  - Relations: 1-1, 1-many, many-many
  - Inheritance, Properties/Fields, etc.
- Current Limitations:
  - Need of an empty constructor (can be private)
  - ObjectList/ObjectHolder to achieve delay loading
- Define and initialise data model / schema
  - Currently with Object Persistence class

Queries

- ObjectSpaces uses OPath as a query language
- Defined over the exposed domain model
  - Can use public properties over private fields
- Natural for the developer
- Able to express complex queries
  - Relationships navigation
  - Set restriction based navigation
- Similar to XPath but not the same!

Materialisation

- Direct Query
  - Multiple explicit queries for the objects needed
- Span (a.k.a Eager Loading)
  - Materialising connected objects when the primary ones are loaded
- Delay loading (a.k.a. Lazy Loading)
  - Materialising connected objects just when they are accessed
- DbObjectReader
  - Read object from any DbReader

Updates and Concurrency

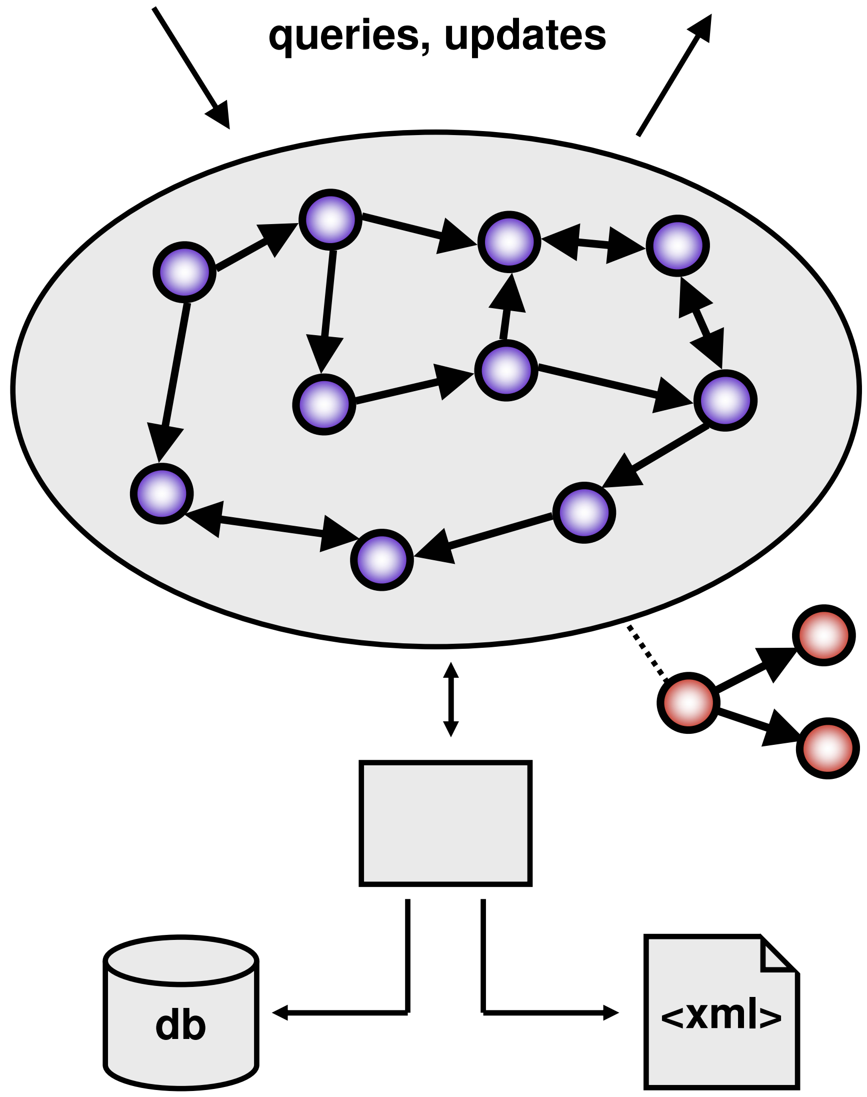
- Updates
  1. Modify your object model
  2. Call PersistentChanges
- Transactions
  - Wrap your code in BeginTransaction…CommitTransaction
- Optimistic concurrency: throw an exception if the data has changed in the data store. Alternatives:
  - Checking all the columns
  - Checking a subset of the columns
  - Using a timestamp field

ADO.NET and Neo

ADO.NET

- One of the major components of the .NET framework
- Can represent XML documents but is **not** XML-based
- Dynamic data API
- Support for a simplistic static API
- Change tracking
- Object association uses a relational model
- Relational-style meta-data and constraints
- Limited SQL generation

Neo – Architecture



Application code

ObjectContext & Entity Objects

EntityMap

DataStore

Neo – Design Goals

- Business entities are represented by objects
- One Entity Object class per database table
- One Entity Object per database row
- Transparent access to derived properties
- Transparent access to related entities
- Strongly typed API
- Automatic generation of database schema and class templates from a single XML file
- Separation of generated and custom code
- Full integration with ADO.NET framework

Domain Objects

- Model / schema defined in XML file
- Base class with attributes and relations is generated
    - Relations: 1-1, 1-many
- Main class contains business logic
- Current Limitations:
    - Need specific constructor, but lifecycle methods

Queries

- Neo uses Qualifiers to define queries
  - Very similar to OPath
- Defined over the exposed domain model
  - Can access private properties and transient attributes as well
- Natural for the developer
- Able to express complex queries
  - Relationships navigation
- FetchSpecifications allow further customisation:
  - Fetch limit, refresh, sort ordering

Materialisation

- Direct Query
  - Multiple explicit queries for the objects needed
- Span
  - not supported, yet
- Delay loading (a.k.a. Lazy Loading)
  - Materialising connected objects just when they are accessed
- Import and export of ADO.NET data bearing objects, i.e. DataSet and DataRow
- Transfer from parent context

Updates and Concurrency

- Updates
  1. Modify your object model
  2. Call PersistentChanges
- Optimistic concurrency: throw an exception if the data has changed in the data store. Alternatives:
  - Checking all the columns

Neo and ObjectSpaces

- Similar in approach and design
- Implementation quite different due to ADO.NET
- Very similar query languages
- ObjectSpaces uses POCOs with advanced mapping
- ObjectSpaces has more complete feature set:
  - Spans, inheritance, pessimistic concurrency are all missing from Neo – but it's Open Source!
- Neo is not coupled to data store
- Neo can ‘stack’ object contexts – great for rich clients

www.thoughtworks.com

erik@thoughtworks.com

neo.codehaus.org