Data Engine Thinking

by Roelant Vos and Dirk Lerner

Training Overview

Practical training with ready-to-use patterns to architect, implement and fully automate your data solution.

| Contents – Day 1

The first training day is focused on the essential concepts and architecture for the data solution, and what the overall objectives are for working with data.

There are various ways to design and model data solutions.

These are more than just technical alternatives: it is important to have a clear understanding of their underlying ideas, and how they impact the overall solution architecture.

The training content includes a refresher on intra-systems integration, implementation approaches, data behaviour and modelling for business process alignment, as well as the entities and constructs used in data solution modelling.

To start focusing on automation and code generation, the mechanism for capturing design metadata is also covered on the first day.

At the end of the day, we will have covered the main architecture building blocks ("patterns") for modelling, designing, and implementing a flexible data solution and how to capture this in design metadata.

Forenoon sessions (08:30–12:30):

- Session 1 – Introductions
- Session 2 – Pattern-based design

  Break – 15 min – Morning tea (10:45–11:00)
- Session 3 – Data solution architecture(s)

Afternoon sessions (13:30–17:00):

Lunch break (12:30–13:30)

- Session 4 – Data staging concepts

  Break – 15 min – Afternoon tea (15:00–15:15)
- Session 5 – Modelling concepts
- Session 6 – Metadata and code generation

The second training day covers the key areas of the *integration layer* – the heart of the solution.

We start with an overview of the Core Business Concept (CBC) pattern, the Natural Business Relationship (NBR) pattern, and delve into advanced topics of managing contextual data and the complexities around time-variance.

At this stage, the main entity types are available in the data solution, and the focus shifts to orchestration approaches – and how to embed these in a process control framework.

The control framework is an essential component to guarantee reliable information delivery and ties into the conceptual and technical implications of parallel loading and data consistency.

The collection of patterns will then be delivered using process automation, as part of release management and "DevOps".

Forenoon sessions (08:30–12:30):

- Session 1 – Core Business Concept pattern
- Session 2 – Natural Business Relationship pattern

  Break – 15 min – Morning tea (10:45–11:00)
- Session 3 – Context pattern, part 1 (introduction)

Afternoon sessions (13:30–17:00):

Lunch break (12:30–13:30)

- Session 4 – Context pattern, part 2 (historization, handling time-variant data)
- Session 5 – Control framework

  Break – 15 min – Afternoon tea (15:00–15:15)
- Session 6 – Testing
- Session 7 – Technical considerations
- Session 8 – Orchestration, workflows, and parallelism
- Session 9 – DevOps and versioning

Day 3 is focused on the delivery of information for consumption. This means investigating the transition from the integration layer to various forms of data delivery ("marts").

As part of the information delivery, the application of business logic and technical considerations are covered.

Data delivery requires a transformation of the timelines that are used, including different ways to consider historised data. The *bitemporal* approach is a key component of this step, and requires a significant amount of time on Day 3.

When all the parts of the solution are finally in place, we review the end-to-end solution, and to what extent the patterns supported by automation simplify its delivery.

Forenoon sessions (08:30–12:30):

- Session 1 – Temporality concepts

  Break - 15 min - Morning tea (10:45-11:00)

- Session 2 – Data delivery for consumption

Afternoon sessions (13:30–17:00):

Lunch break (12:30–13:30)

- Session 3 – Application of business logic

  Break - 15 min - Afternoon tea (15:00–15:15)

- Session 4 – Completing the solution

| Prerequisites

- Sufficient understanding of English (course language)
- Understanding of data engineering (e.g. Data Warehouse, ETL development)
- Good understanding of SQL (joins, window functions)
- Basic scripting/programming awareness (e.g. C#, Python)
- Familiarity with data modelling for Data Warehousing (CIF, Kimball, Data Vault, Anchor)

| Session modules

Pattern-based design

The direction to move away from manually creating data integration logic, towards a more pattern-based approach directed by the information model, is called Model Driven Design or Pattern-Based Design. Fundamentally this is about cultivating a mindset of flexibility in design by leveraging modular patterns and supporting technologies.

This session provides an introduction to the thinking behind data logistics generation and automation.

- Required components for model-driven / pattern-based design
- Styles of data logistics generation
- Guiding principles and requirements
- Family of hybrid modelling techniques
- Overview of modelling concepts and core entity types
- Overview of core implementation patterns

Data solution architecture

This session takes a step back and looks at the overall design. Now that there is a foundational understanding of the modelling approach and its intent, the end-to-end architecture can be explained. This provides a reference point for the advanced topics and explores design considerations, especially the interactions between different concepts. What needs to be done where, and what are the impacts of certain design decisions?

- Overview of layers and areas in a data solution architecture
- Architecture options and considerations
- Back-room and front-room operations
- Combining multiple platforms and technologies
- Specifics and requirements of each area
- Separation of concerns

Data staging concepts

Getting the data into the data solution is one of the most complex areas of the architecture. This session covers the fundamental concepts that need to be included in any design and

explores the impacts on subsequent layers.

- Different data staging approaches (patterns)
- Implications of date/time stamping (where/how to capture load dates)
- Key requirements for a Staging Layer
- Persistent Staging Area (PSA) considerations
- Preparing for near-real-time
- Supporting parallel processing
- Change Data Capture (CDC)

Design metadata

With the source and target models available, focus shifts to the design (mapping) metadata itself: what metadata is required, where it is located, and how it should be stored.

Regardless of whether you are developing a solution yourself or using commercial Data Warehouse Automation software, the metadata is the same.

- Overview of required patterns
- Overview of required metadata

Code generation

Design metadata captures what data needs to go where, why, and how. Defining a template drives how this will be delivered. This is a practically-oriented session that includes hands-on exercises.

- Defining code generation templates
- Merging design metadata with templates to generate code

Core Business Concepts pattern

The Core Business Concept (CBC) entities are the single most defining aspect of a data solution and heavily influence subsequent design decisions. They are also the most straightforward patterns to implement.

Given their critical role, it is essential that their data logistics "just work".

- Pattern, structure, and implementation
- Parallelism
- Technical types of Business Keys
- Key distribution approaches
- Metadata and code generation

Natural Business Relationship pattern

If CBCs are the "joints" of the model, Natural Business Relationships (NBRs) are the "bones". NBRs manage relationships between concepts and govern granularity and "Unit of Work".

- Pattern, structure, and implementation
- Recursive and clustering mechanisms
- Degenerate attributes
- Metadata for code generation

Context pattern

Context entities provide the details ("describe") for CBCs and NBRs. This is where time-variance is managed, and data changes are captured in time.

- Pattern, structure, and implementation
- Redundancy
- Row condensing (independent or CDC-based)
- Change merging
- Column scope
- End-dating
- Zero records
- Multi-Active (multi-variant) approaches

Technical considerations

Technology must be configured to support a robust and scalable solution. Correct indexing, partitioning, and parallelism have a huge impact.

- Compression
- Partitioning
- Indexing
- Filtering
- Referential integrity
- Error handling

Control framework

A control framework is essential for reliable data delivery, beyond simple auditing. It ensures consistency of delivered information.

- Transaction isolation at application level
- Grouping for execution of load processes
- Rollback and recovery

Testing

Reusable, generic tests are valuable for quality development and regression prevention in DevOps.

- Defining a testing framework
- Creating test cases
- Applying data checks without judgment

Orchestration, workflows, and parallelism

Parallel processing should load data as soon as it is available. This session covers design and implementation considerations.

- Batching vs independent execution
- Parallelism: impacts on solution design
- Process redundancy
- Referential integrity in parallel loading

DevOps and versioning

This session focuses on automating the workflow of activities to create and deploy integration processes. Release management impacts reliability and flexibility.

- Metadata integration
- Automation of code generation
- Automating process workflows
- Organising DevOps

Temporality concepts

Combining multiple time-variant tables into a single delivery is critical. The *point-in-time* (PIT) approach is explained in detail.

- Time-variance concepts (“date math”)
- How to join time-variant entities
- Timing issues and resolutions
- PIT pattern: structure and implementation
- Stacked vs continuous PIT approaches
- PIT for data virtualisation
- Load order impacts

Data delivery for consumption

The presentation layer is defined as anything fit-for-purpose. Its role is to clarify requirements over time rather than upfront.

- Dimension patterns
- Fact table patterns
- Types of history
- Handling timing issues
- Switching time perspectives

Application of business logic

Source data is rarely in ideal form. Alternative patterns are needed. Applying business logic within a clear architecture simplifies this and ensures flexibility.

- Business logic: front-room vs back-room
- Recording and managing business logic
- Handling competing interpretations
- Transformation options and considerations
- Driving keys
- Impacts of date/time selection