PolarLake

Oracle XMLDB

PolarLake Background

Founded in 2003

Offices in Dublin, London, New York

XML integration heritage – patents in XML processing

Warren Buckley – CTO and Founder

PolarLake Clients

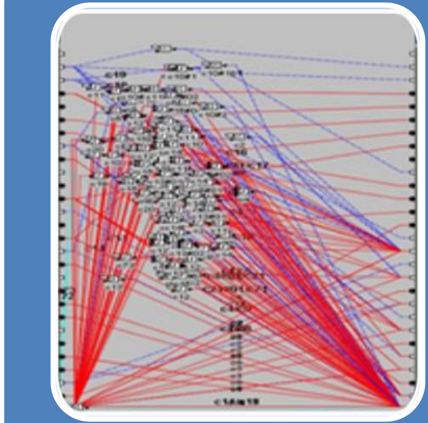
6 of the top 10 Investment Banks

2 of the top 5 Prime Brokers

2 of the top 10 Asset managers

Control Pricing and Reference Data Distribution and Integration with PolarLake

Pricing & Reference Data Management and Distribution



The need for PolarLake?

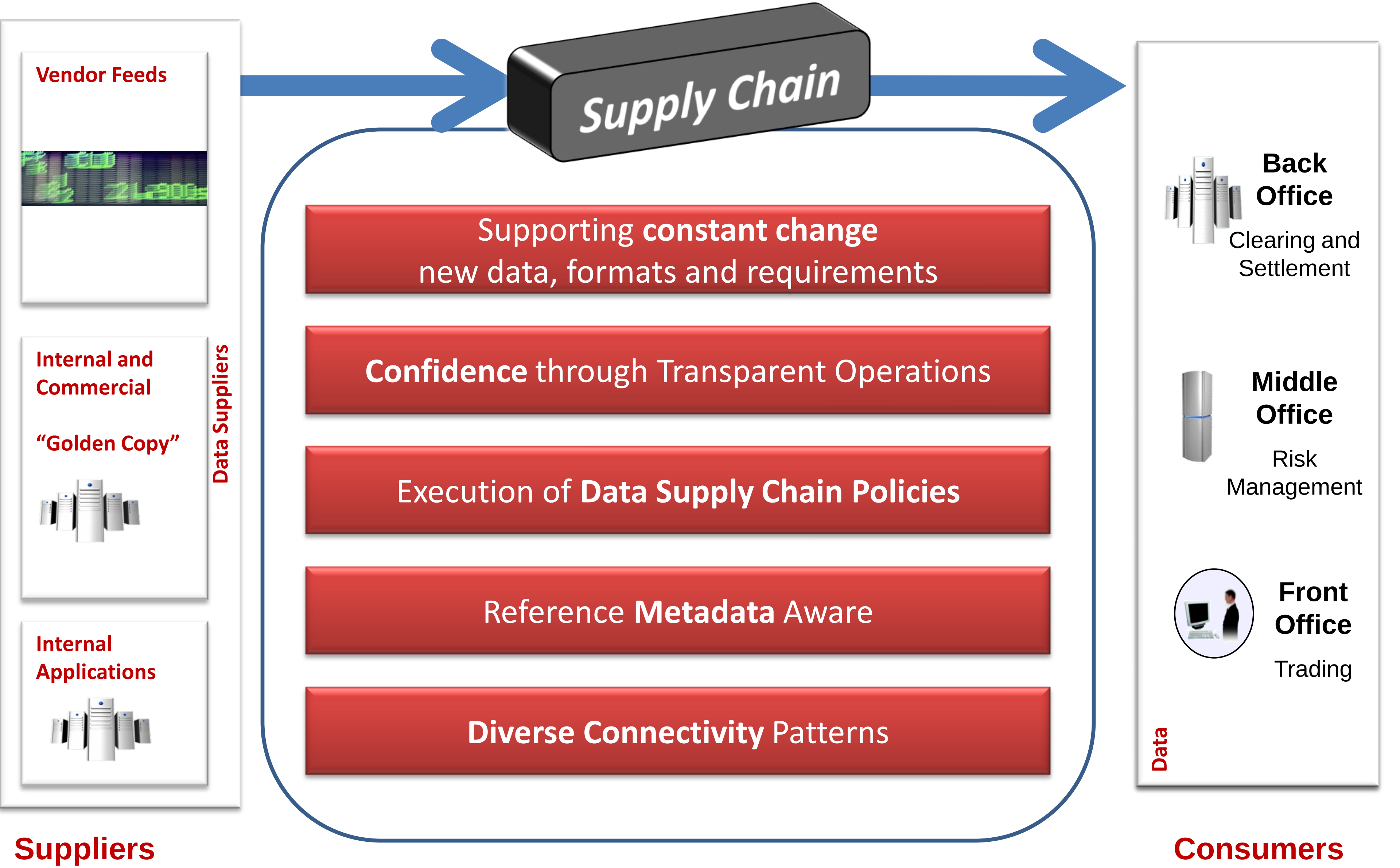
- Frustration with complexity, cost and time to market of ETL/EAI/DB replication solutions to distribute Pricing & Reference Data Distribution and Trade Integration
- Inflexibility of fixed data models
- Experienced Trade and Reference Data integration specialists

The PolarLake Benefits

- **Control & Confidence** in delivery – end to end transparency of data and trade supply chain
- **Time to Market** – up to 95% rule generation, 75% less time, 80% less resources than traditional approaches
- **Manage ongoing complexity** – purpose built for change and unstructured data. Practical approach to Data Management

Data Supply Chain Management - Objectives

PolarLake



Vendor Feeds

Internal and Commercial

“Golden Copy”

Internal Applications

Data Suppliers

Suppliers

Supply Chain

Supporting **constant change**
new data, formats and requirements

Confidence through Transparent Operations

Execution of **Data Supply Chain Policies**

Reference **Metadata** Aware

Diverse Connectivity Patterns

Back Office

Clearing and Settlement

Middle Office

Risk Management

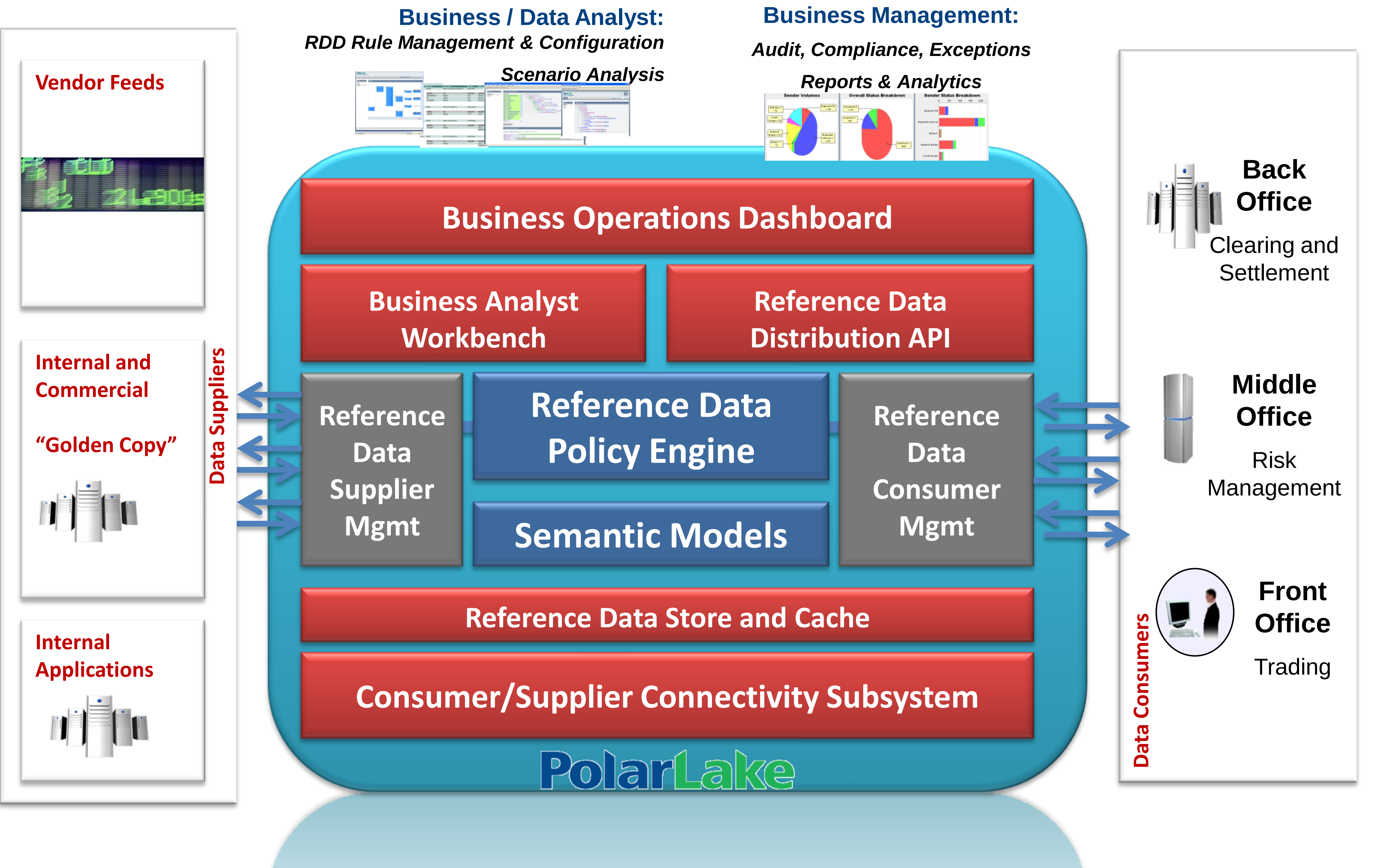
Front Office

Trading

Data

Consumers

Pricing, Reference Data & Trade Integration



Vendor Feeds

Internal and Commercial

“Golden Copy”

Internal Applications

Data Suppliers

Business / Data Analyst:

RDD Rule Management & Configuration

Scenario Analysis

Business Management:

Audit, Compliance, Exceptions

Reports & Analytics

Business Operations Dashboard

Business Analyst Workbench

Reference Data Distribution API

Reference Data Supplier Mgmt

Reference Data Policy Engine

Semantic Models

Reference Data Consumer Mgmt

Reference Data Store and Cache

Consumer/Supplier Connectivity Subsystem

PolarLake

Back Office

Clearing and Settlement

Middle Office

Risk Management

Front Office

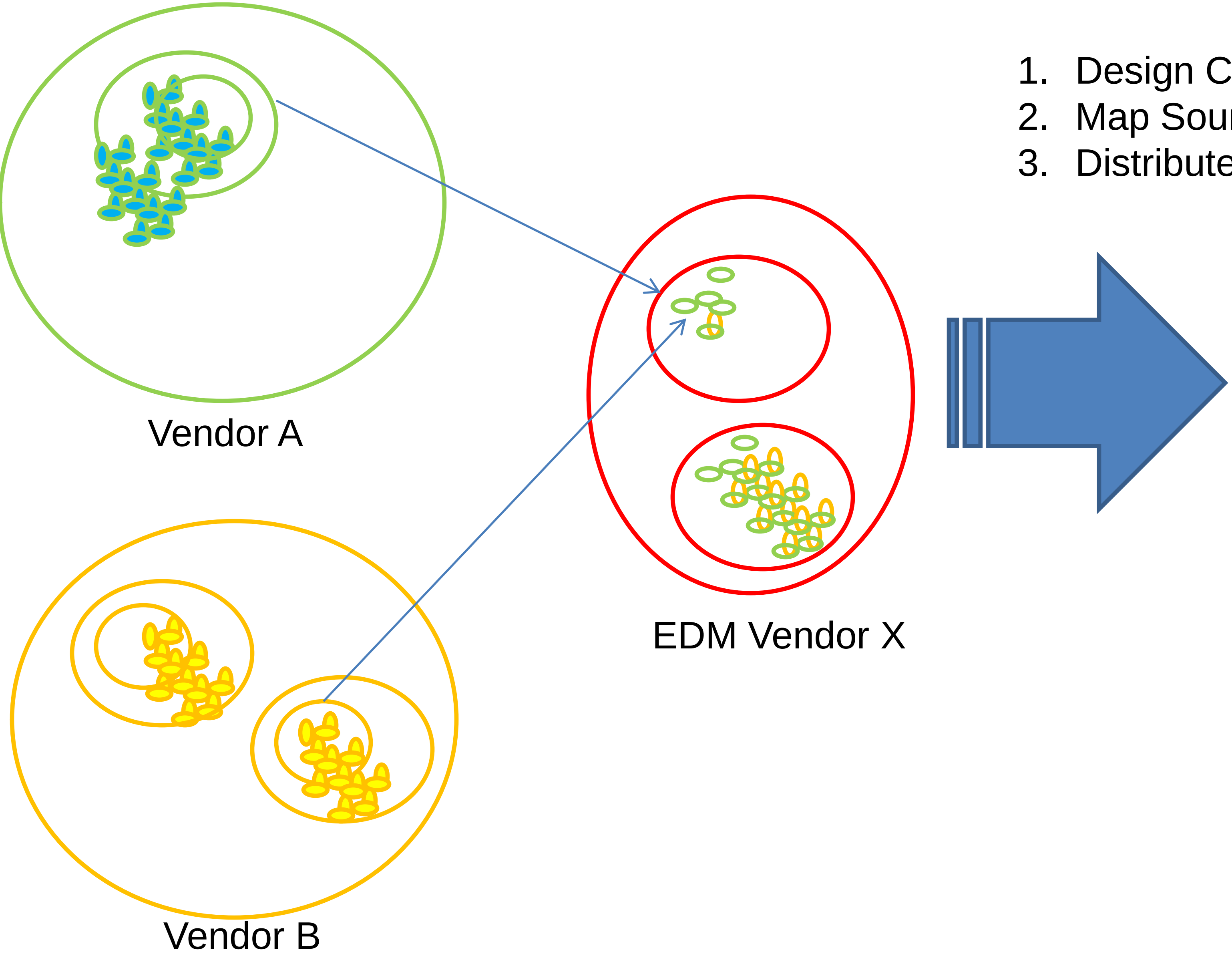
Trading

Data Consumers

PolarLake and Oracle

Why Oracle XMLDB

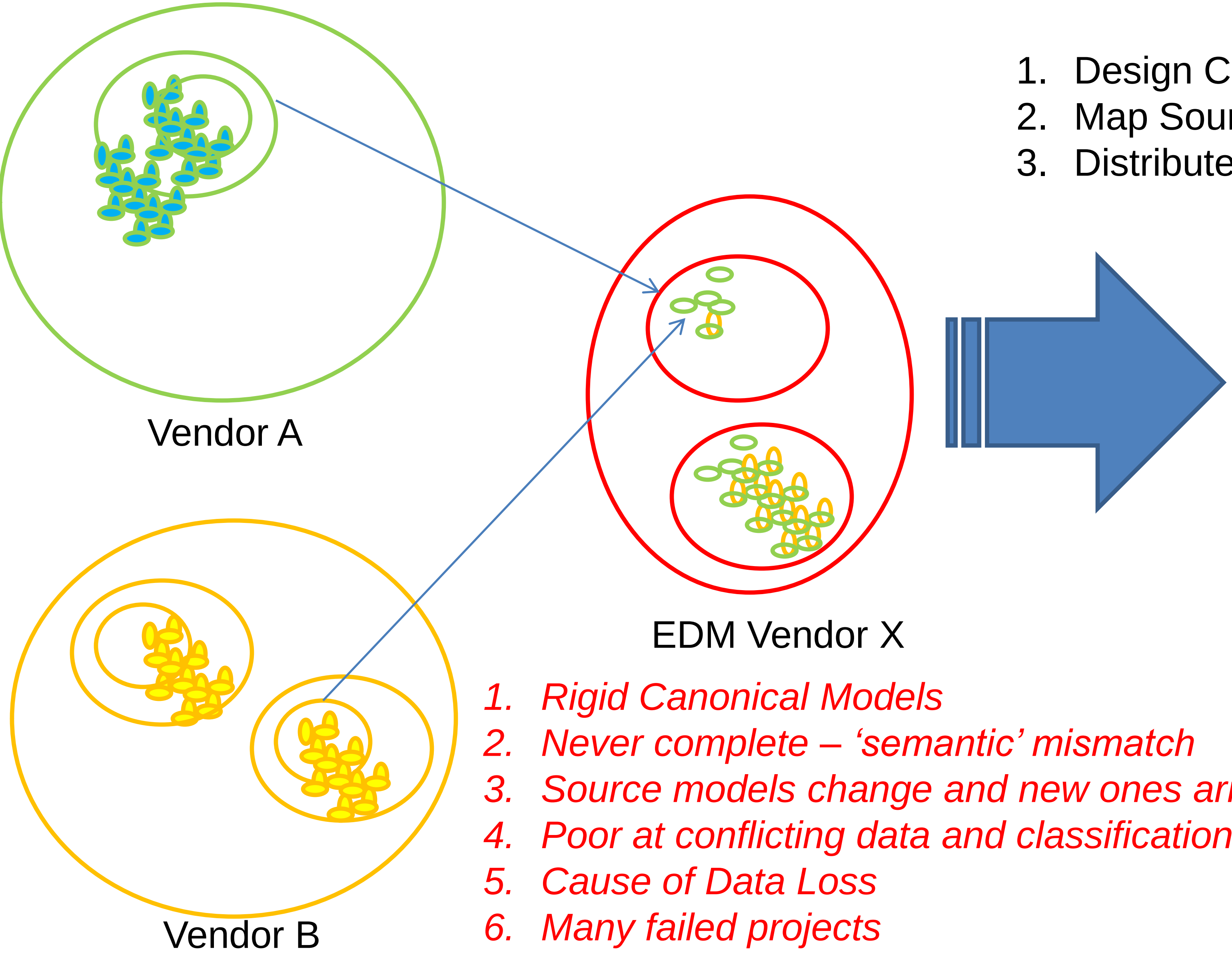
EDM – The default approach



1. Design Canonical Model
2. Map Sources to Model
3. Distributed from Model

EDM – The default approach has issues

1. Design Canonical Model
2. Map Sources to Model
3. Distributed from Model



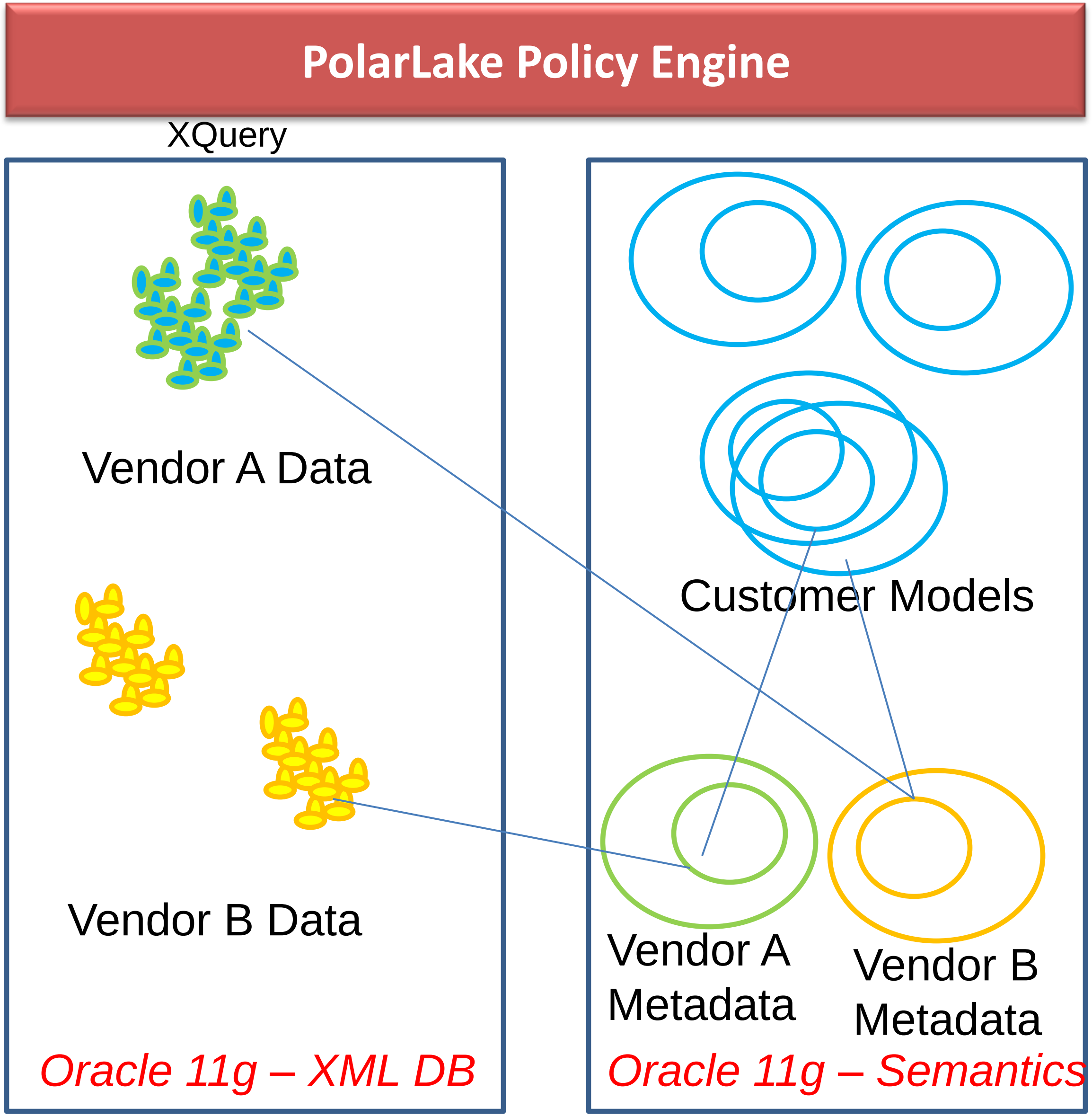
1. *Rigid Canonical Models*
2. *Never complete – 'semantic' mismatch*
3. *Source models change and new ones arrive*
4. *Poor at conflicting data and classifications*
5. *Cause of Data Loss*
6. *Many failed projects*

PolarLake Data Management

Data Loading

Generic XML Storage

Change Tolerant

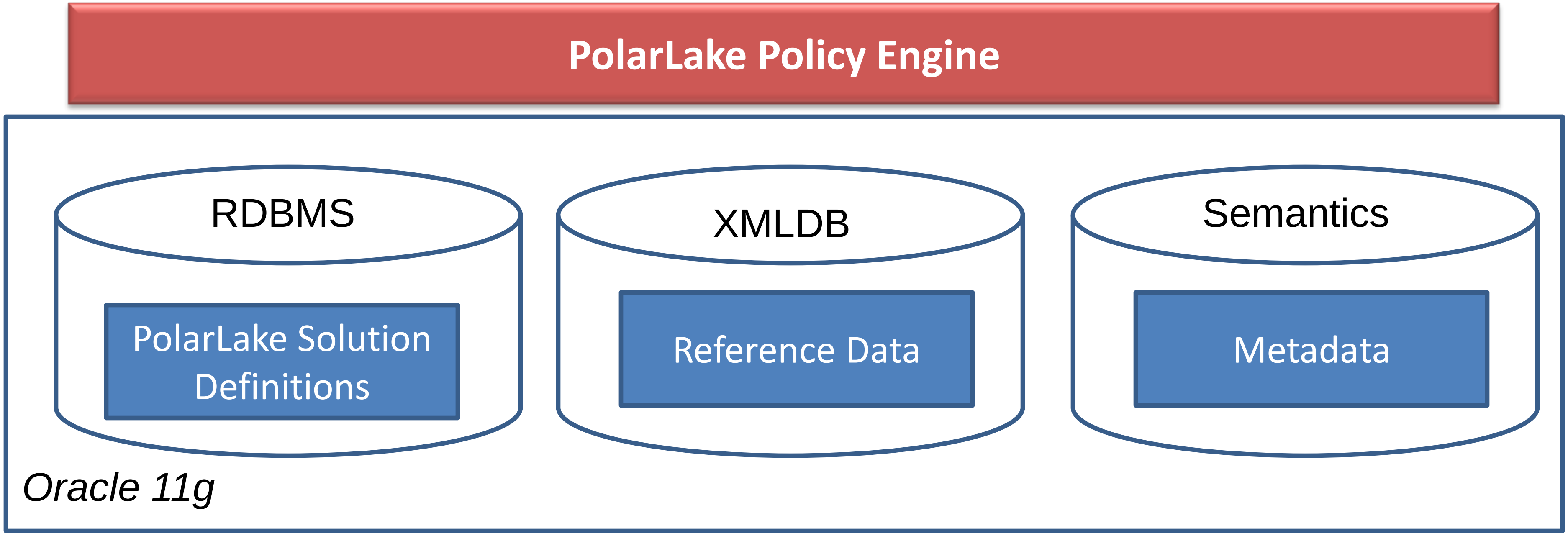


Linkages

Metadata imports

Classification imports

PolarLake and Oracle 11g



Scope

- Technology Platform
  - J2SE and J2EE
  - PolarLake Messaging Integrator (XML OS)
  - Primary port Linux (Solaris, Windows, IBM mainframes, etc.)
- Data volume
  - ~ 8 million documents a day
  - 4 types (pricing, ratings, reference, etc.)
  - Documents vary in size from 2K to 1M
    - trade management has less volume but more complex and larger documents)

Oracle XML DB

- Binary XML Storage
  - Unstructured XML Index
- XMLIndex
  - Performance improvements
- Java API –oracle.xdb.XMLType

- Combining XMLDB, Semantics and Oracle Text is extremely powerful.

Cross Storage Queries

```
SELECT rddsem.file, rddxml.doc, rddxml.src  FROM

        TABLE(SEM_MATCH(
                '(?filetype :contains :Issue)
                (?filetype :providedBy :Reuters)
                (?file rdf:type ?filetype)',
        SEM_Models('rdd'),
        SDO_RDF_Rulebases('RDFS'),
        SEM_ALIASES(SEM_ALIAS('','http://www.polarlake.com/rdd#')),
        null)) rddsem,

        XMLTable('
                for $i in ora:view("borec") return $i
        ' COLUMNS
        doc XMLTYPE PATH '.', src varchar2(100) PATH '/xml/@src') rddxml

where rddxml.src=rddsem.file
and contains(rddxml.doc,'$(Oracle) INPATH (xml/LONG_COMP_NAME) ') >0;
```

Conclusions

- XQuery and SPARQL joins are powerful.
- Performance improvements by pushing logic to the database tier.
- Mid Tier APIs look promising
- Need to separate XML Query and XML Construction.
- Need to separate Procedural and Declarative approaches.
- Careful crafting of XQuery for performance is essential.
- Careful design of paths in XML Indices required.
- Oracle 11g Release 2 - XML in the database has arrived *for real*.

Questions

warren.buckley@polarlake.com