Keeping Service Oriented Computing Simple

MW4SOC
November 2007

P. M. Melliar-Smith
University of California, Santa Barbara

Service Oriented Computing

- Service Oriented Computing allows programs to interact with each other on demand without being tightly coupled together
  - Internet allows easy communication
  - Globalization and offshoring require enterprises to communicate

Standards

- Several standards for distributed computing between enterprises have been established recently, in particular
  - Web Services
    - SOAP
    - REST

Prior Standards

- These were not the first standards for distributed enterprise computing
- Earlier standards were
  - CORBA
  - J2EE/EJB
  - .Net/C#

The Slippery Slope

- The standards:
  - Grow
  - Add features
  - Become more complex
  - Eventually reach a level of complexity at which typical enterprise programmers find them very difficult to use

Incompatible Objectives

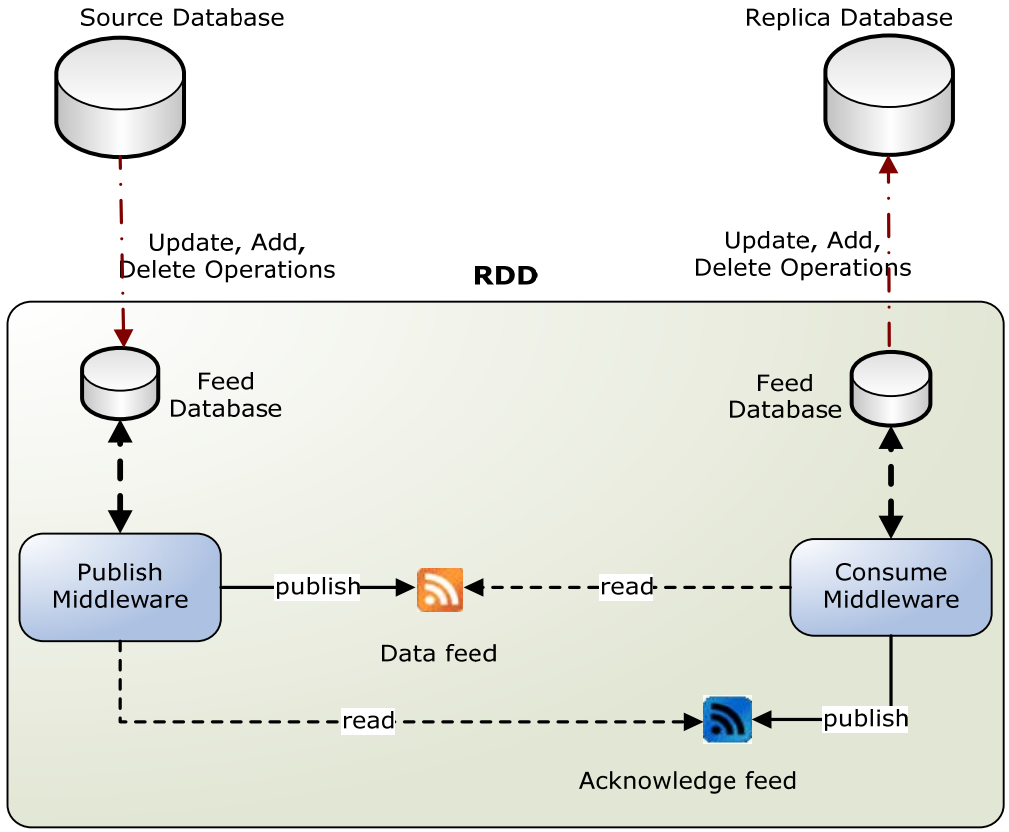
- **Interoperability** between programs written by different programming teams using different hardware platforms, programming languages and database systems, who were not even aware of each other's existence when the programs were written
  - Interoperability requires isolation and independence between the programs written by the different programming teams
- **Efficiency**, particularly low latency in accessing services and data from other enterprises
  - Efficiency has typically been interpreted as requiring direct procedural access to remotely provided services and data, thus reducing the isolation and independence of the programs

Distributed Programming Strategies

- Most distributed systems use one of three strategies:
  - Message Passing
  - Remote Procedure Call (e.g., SOAP)
  - Remote Data Access (e.g., REST)
- There is a fourth alternative
  - Distribution of database records to mirror databases on remote computers, with local access to those records by application programs on the remote computers

Data Distribution

- The data to be communicated to other enterprises are stored in a local database
- The data are published as an Atom or RSS feed
- When a consumer receives the feed, it stores the data in a local database
- The application programs at the consumer access the data from the local database
- The local database at the consumer is a mirror of the database at the source



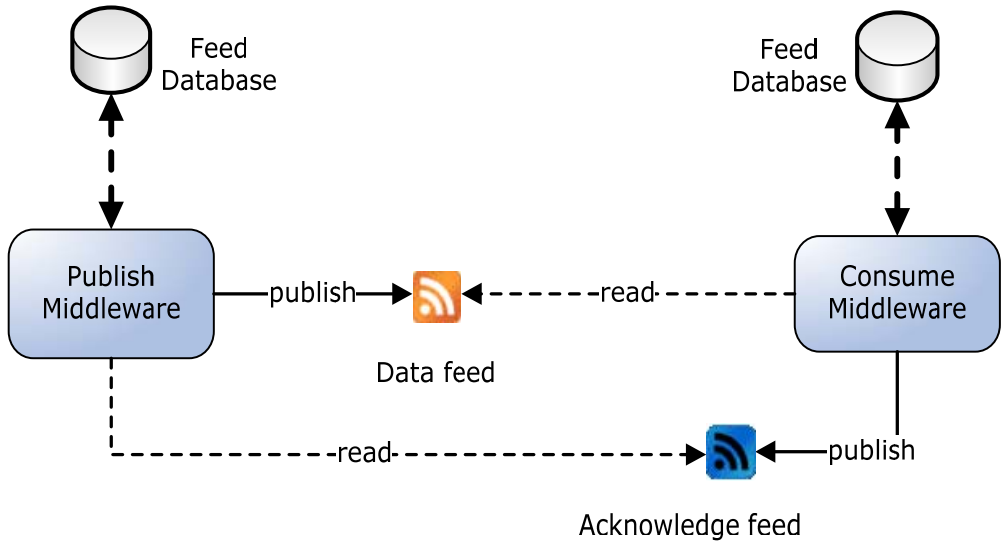
Experimental Implementation

Implementation by Firat Kart, Ph.D. student at UCSB

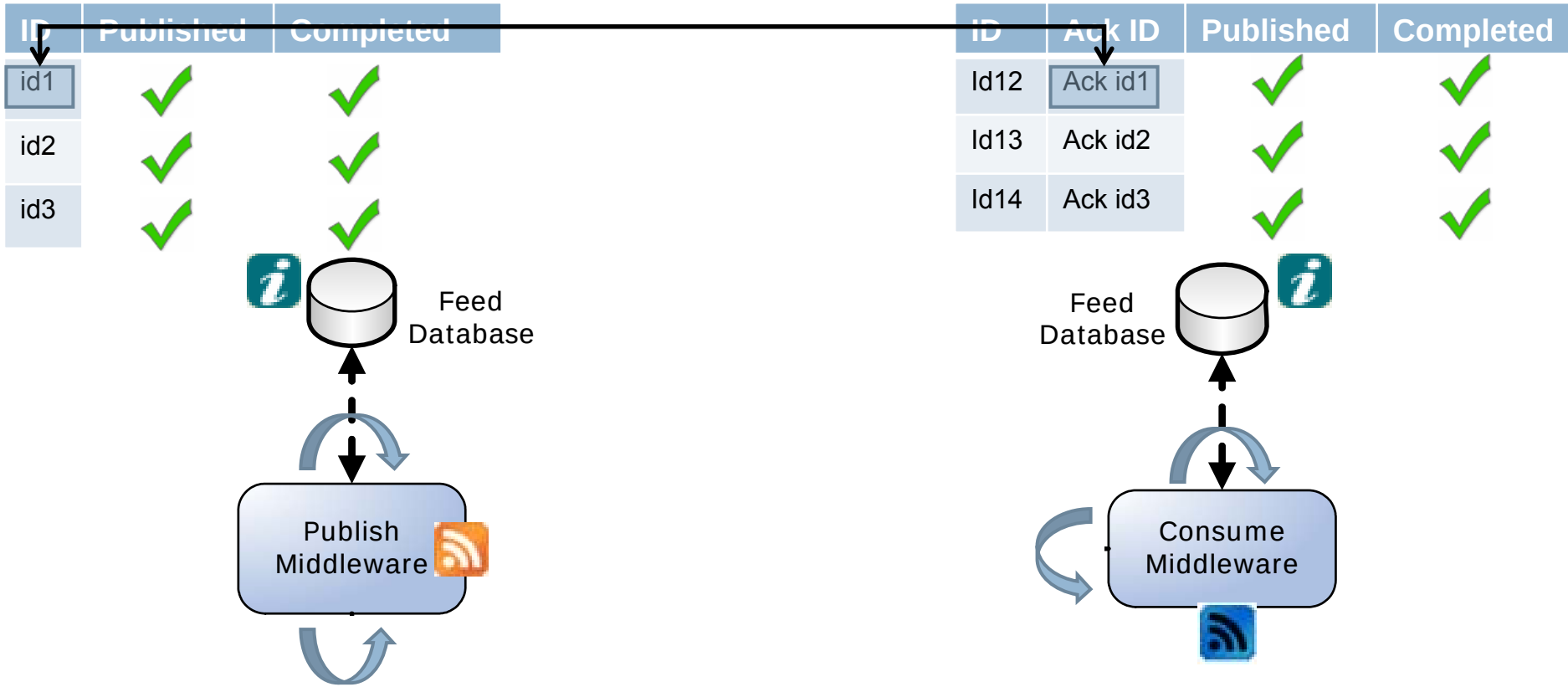
- Reliable Data Distribution (RDD)
  - Reliable communication protocol for Atom
- Consistent Data Replication (CDR)
  - Data replication protocol using RDD
  - Mirrored local copy of the publisher's data is stored at the consumer(s)
- Database Aggregation (DBA)
  - Automatic runtime schema conversion

Reliable Data Distribution

- Publish Middleware
  - Uses Web server to publish data feeds
  - Reads feeds from consumers
- Consume Middleware
  - Reads feeds that publisher publishes
  - Uses Web server to publish acknowledgement feeds

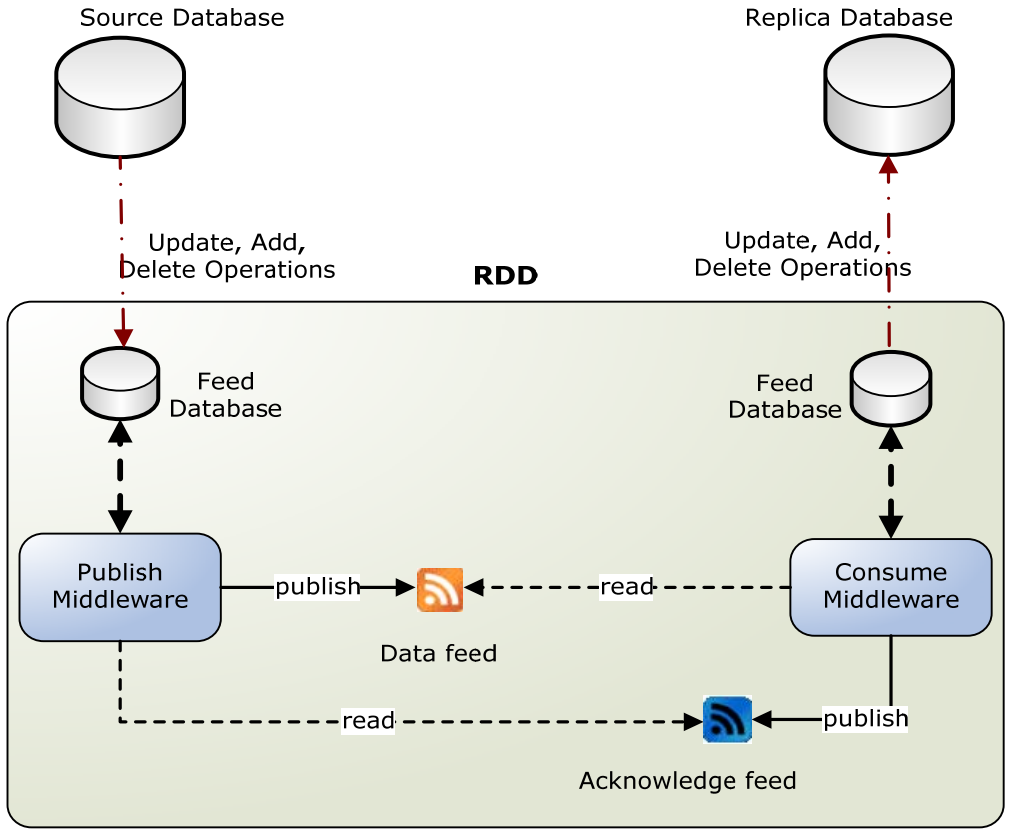


Reliable Data Distribution

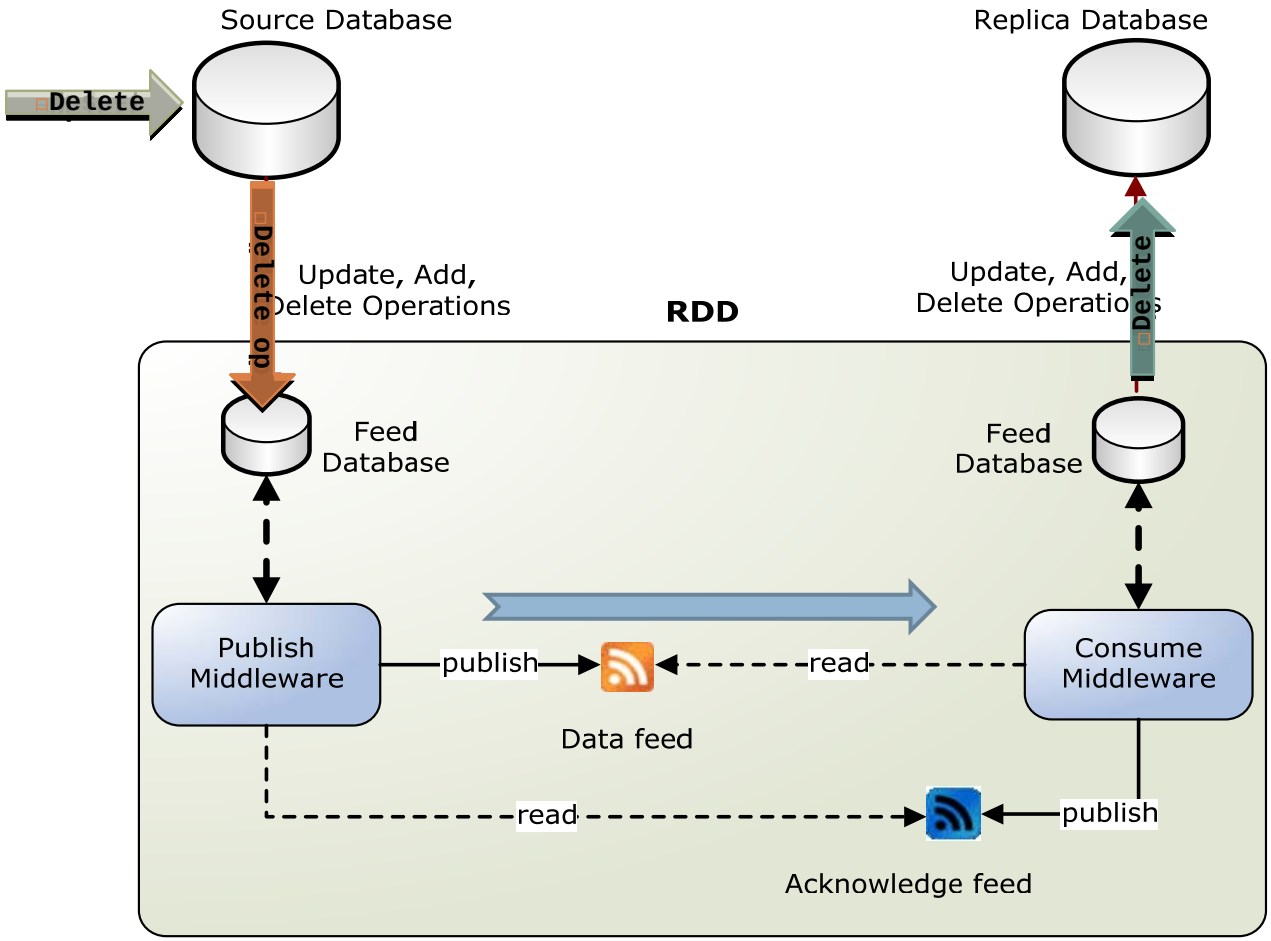


Consistent Data Replication

- Replicates information in source database to multiple target databases
- Uses RDD as the communication protocol
- Augments original database with additional triggers (UPDATE, INSERT, DELETE operations)

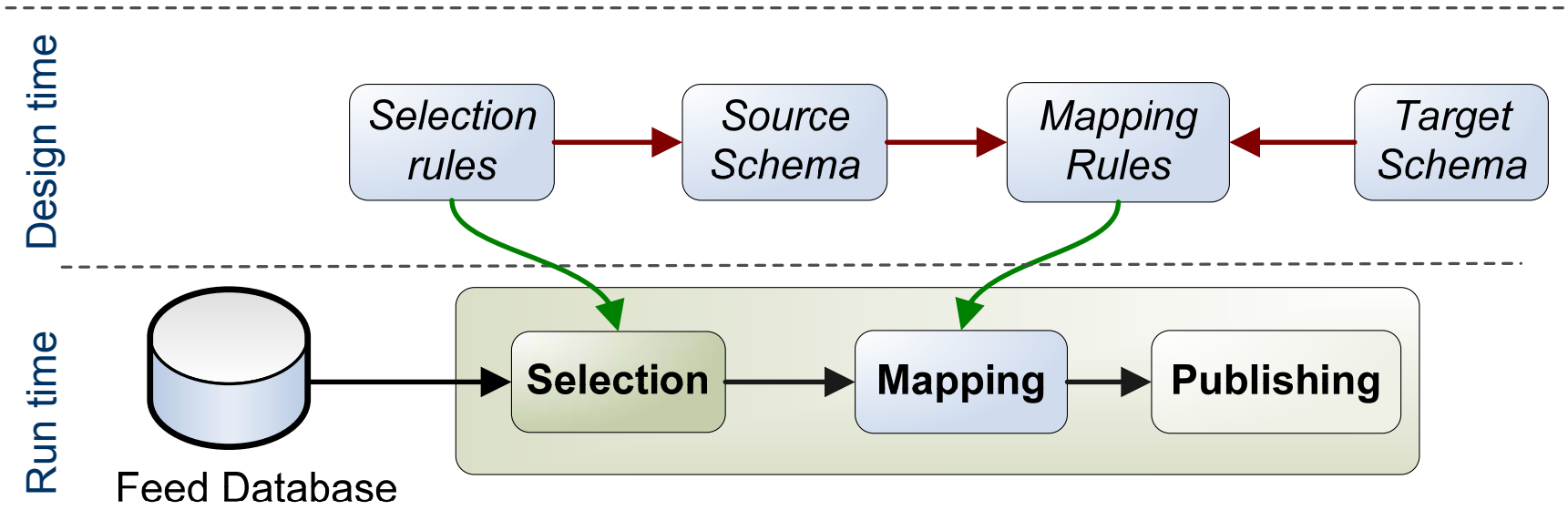


Consistent Data Replication



Database Aggregation

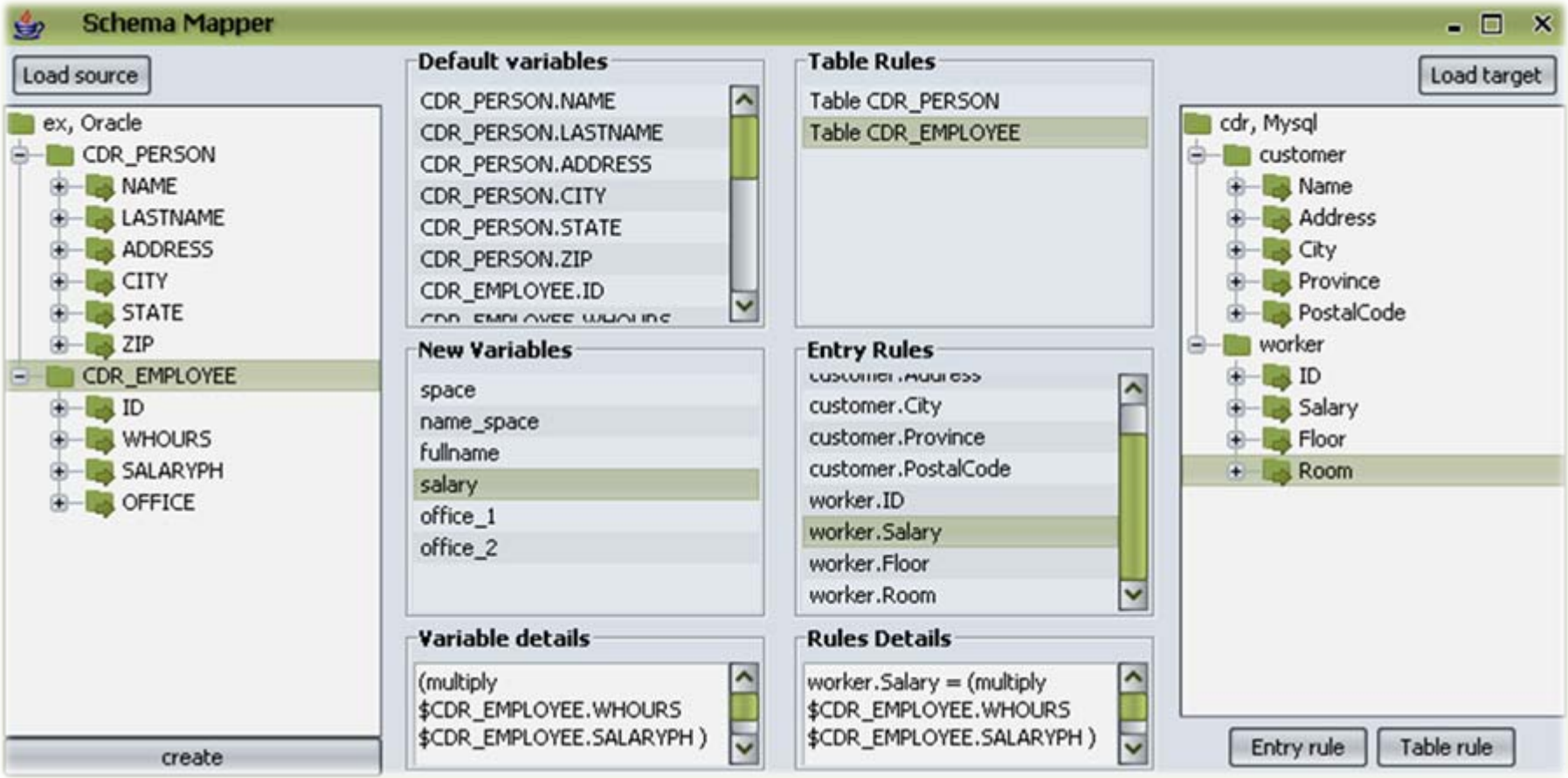
- Transform source database content into target database content
- Design time rules are used for mapping



Modules

- Selection module
  - Limited information to be published
- Mapping module
  - Transforms from source schema to target schema
  - Transformation
    - String functions
    - Logical operations
    - Math functions
- Publishing module
  - Generate Atom feed to be published

Rule Mapping Tool



Rules File

- Example XML rules file

```
<Rule>
  <Target>worker.Salary</Target>
  <Mapping>
    (multiply $EMPLOYEE.WHOURS $EMPLOYEE.SALARYPH )
  </Mapping>
</Rule>
<Rule>
  <Target>worker.Floor</Target>
  <Mapping>
    (split $EMPLOYEE.OFFICE - 1)
  </Mapping>
</Rule>
```

Implementation

- Rome tool to publish/consume Atom feeds
- Xstream to generate XML content
- Oracle database at the publisher
- MySQL databases at the consumers

ORACLE

MySQL

Simplicity

- Simplicity of design arises because each enterprise is free to define its own databases with its own schema
- Simplicity of application programming results because the application programs access only local data from a local database using that enterprise's own database schema
- If the enterprise must interact with a new enterprise, and that new enterprise uses a different data representation or database schema, there is no need to reprogram the application programs
- Isolation facilitates ease of programming

Interoperability

- Communicating enterprises use different hardware platforms, operating systems, programming languages and database systems. Inevitably, they use different data representations and database schemas
- Atom/RSS feeds are well established and highly interoperable, due to their use of XML with its tags and self-defining data
- Conversion of data in a database to data in a feed, and back again, is highly automated
- Isolation facilitates interoperability

Performance

- The time to propagate new or updated data from one enterprise to another is several seconds. This time is not significant because it occurs in the background, off the critical path of the applications
- The time required for applications to access data is the critical time for most applications
  All data are accessed from the local databases
- Isolation facilitates good performance

Security

- The enterprise decides which data to publish and make available to other enterprises
- The enterprise decides which feeds to access, and which data to convert into database records
- Remote sites do not access the data or programs of an enterprise directly
  - Security breaches can be greatly reduced
- Isolation facilitates security

Conclusions

- To avoid the slippery slope into complexity, we must consider alternative strategies
- We have proposed an alternative strategy
  - Distributing data to mirror databases
  - Application programs access data from their local databases
- Ease of programming for the applications is the primary objective

Discussion

Grazie

Teşekkürler!

Danke

Вы

धन्यवाद

شكرا

Thank You !

Gracias

Merci