Metamodels and Modeling Multiple Kinds of Information Systems

Randall M. Hauch

Chief Architect

metamatrix

presented at

OMG OBJECT MANAGEMENT GROUP

MDA, SOA and Web Services:
Delivering the Integrated Enterprise – Practice, not Promise

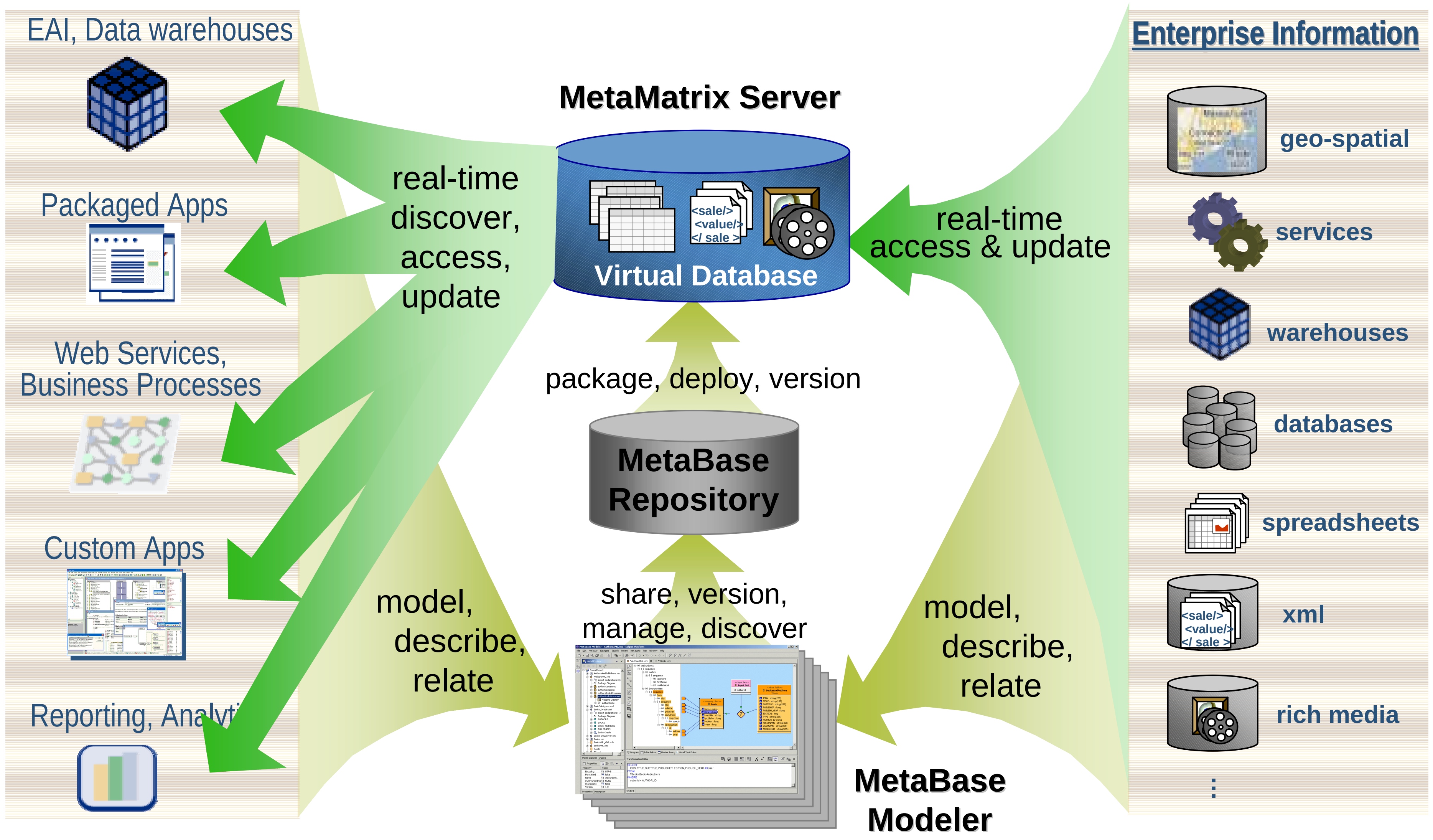
MetaMatrix Products at Work Today

Providing access to integrated information for …

- Government
- Financial services
- Telecommunications
- Life Sciences
- Manufacturing
- Pharmaceuticals

MetaMatrix:Model-driven information integration

Understand, relate, harmonize, rationalize, and use



MetaMatrix

Model-driven information integration

- An integrated, Eclipse-based model development environment (MDE) to model & relate multiple kinds of enterprise information
  - Use multiple domain-specific modeling languages: Relational, UML, Web Services, XML, XML Schema, Taxonomies, Ontologies, etc.
  - Enterprise-class metadata management system with seamless access via JDBC, ODBC and SOAP
- Model-driven and real-time integration of enterprise information
  - Use the models to integrate and provide real-time access (including transactional updates) of disparate enterprise information
  - Supports nearly any kind of information source and stores
  - Highly efficient and optimized for high-volume, high availability
- Enable model-driving other applications and systems
  - Through rich metadata management, packaging and access mechanisms

Standards & Open Source Projects

- OMG standards
  - Meta Object Facility (MOF) with standard & custom metamodels
  - XMI, UML2, CWM, etc.
- W3C standards
  - XML, XML Schema, SOAP, UDDI, HTTP, XPath, XQuery, OWL, etc.
- Supports and uses Java standards
  - JDBC, J2EE, JMS, etc.
- Supports SQL-92 (plus)
- Committed to and built on top of open source projects
  - Eclipse, EMF, Apache, Doug Lea’s Concurrency, and others

More on standards and open-source tomorrow afternoon

Why Model-Driven?

Abstracts the intent from execution

- Models declare what the intent is and what should happen
  - The “how” is determine later
- Interpretation, optimization and execution is left to the runtime engine
  - Access – providing multiple APIs
  - Push-down – do as much work as close to the data as possible
  - Orchestration – coordinating the interaction with multiple sources, including transactional updates
  - Caching – enable reuse of already integrated information
  - Safety – guard against unexpected and unacceptable uses
  - Scalability – handling massive volumes of information and large numbers of concurrent consumer requests
  - Performance – choosing appropriately and using the optimized algorithms for integrating information
  - Security – policies applied at runtime applied heterogeneously
  - Traceability – expose and ensure compliance
  - Improvements – engines can be improved without changes in the models

Model Driving Systems

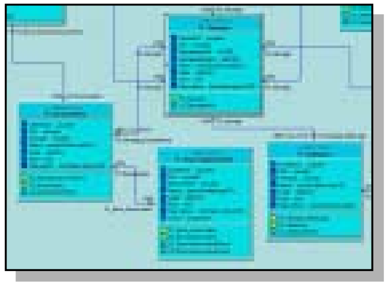
Examples

- Define how information is to be integrated
  - Deploy models to an integration engine for real-time access & updates
  - Use the models to discover, understand & harmonize enterprise information
  - Use the models in a portal with a user interface that is dynamically driven
- Define the configuration and makeup of a web portal
  - The model can be used directly as the configuration descriptor
  - The model can be transformed into a product-specific configuration artifact
- Define BI reports
  - The model (or another artifact generated from it) can be used by the BI tool to determine the report details before execution
- Define the structure of an organization
  - Use the specific constructs of the organization
  - The model can be deployed & treated as a standard enterprise data source
- Many other examples!

Modeling Systems

Each system has it's own semantic concepts

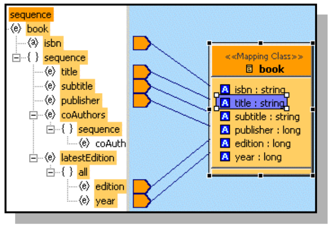
Relational



XML

```
<customer>
   <name>John Smith</name>
   <streetAddress>12 Main Street</streetAddress>
   <city>Springfield</city>
</customer>
```

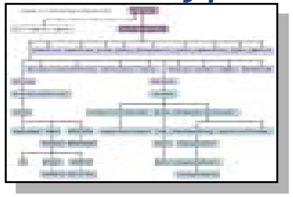
Transformations



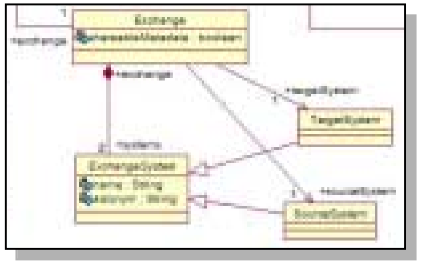
sequence
book
isbn
sequence
title
subtitle
publisher
coAuthors
sequence
coAuth
latestEdition
all
edition
year

<<Mapping Class>>
book
isbn : string
title : string
subtitle : string
publisher : long
edition : long
year : long

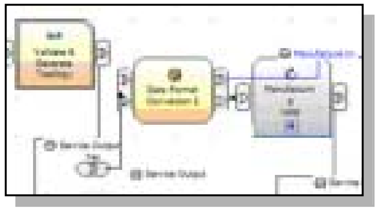
Datatypes



Domain
[UML/ER]



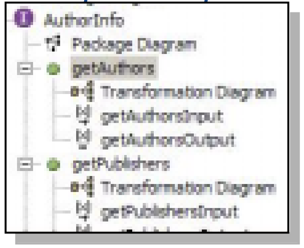
Processes
[BPM/BPEL]



Generic Typed Relationships

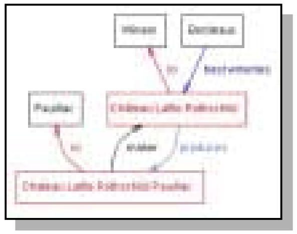
Relationship Role
Relationship Type
Relationship Role
RoleName
AllowedType
Multiplicity
Name
RoleName
AllowedType
Multiplicity

Web Services
[WSDL]



AuthorInfo
Package Diagram
getAuthors
Transformation Diagram
getAuthorsInput
getAuthorsOutput
getPublishers
Transformation Diagram
getPublishersInput

Ontologies
[OWL/RDF]



Taxonomies

Organizations & Business Models

Environmental Management
Law Enforcement
Litigation and Judicial Activ
Correctional Activities
Health
Homeland Security
Income Security
General Retirement and
Unemployment Compen
Housing Assistance
Food and Nutrition Ass
Survivor Compensation

metamatrix

Modeling Multiple Types of Systems

Use one modeling language – or multiple?

- Many modeling tools work with only one
  - Typically UML or Entity-Relational
  - Some applications are used more like file editors than modeling tools
- Systems to be modeled can take many forms
  - Relational tables, views, procedures
  - XML elements, attributes, namespaces
  - Object-oriented classes, properties, operations, associations
  - Simple and complex datatypes
  - Services, components, messages
- Multiple domain-specific languages are best
  - Use the modeling language best suited for the system being described
  - Leverage native artifact structures (e.g., DDL, WSDL, XSD, OWL, etc.)

Modeling Multiple Types of Systems

Using multiple modeling languages

- A “metamodel” defines a domain-specific modeling language
  - Accurate and precise
  - Structured
- Supporting multiple metamodels allow for multiple modeling languages
  - Domain-specific models are easily understood by users
  - Makes possible mixing and matching
  - Enables treating heterogeneous models in homogeneous ways
- Use metamodels to drive behavior of modeling environment
  - Menus, wizards, views are all driven with the metamodels
  - Object construction is driven by metamodels
  - Validation is driven by metamodels
  - Diagramming is driven by metamodels
  - Models are serialized to files using metamodels and XMI rules

Modeling Multiple Types of Systems

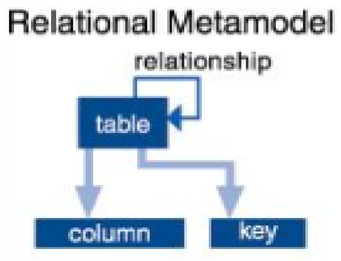
Some of the modeling standards

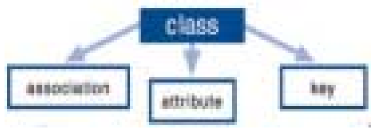
- Meta-Object Facility (MOF)
  - Defines an architecture for modeling
  - Used to create “metamodels” or “domain-specific modeling languages” that define syntax and semantics of models
- XML Metadata Interchange (XMI)
  - Defines rules that dictate how models defined with MOF are serialized to and from XML files
  - Not a single format, but rather patterns for defining formats (XML Schemas) in terms of the metamodels
- Common Warehouse Metamodel (CWM)
  - Defines metamodels for various types of information systems
  - Relational, record, hierarchical, OLAP, etc.
- Unified Modeling Language (UML)
  - The well-known metamodel for object-oriented systems
- OMG’s Model Driven Architecture™ (MDA™)
  - Defines architecture for using models to drive systems

Modeling Multiple Types of Systems

Architectural foundations

- Strong foundations for metadata and modeling
  - OMG's Meta Object Facility (MOF) – standard
  - Eclipse's Modeling Framework (EMF) – programmatic implementation of MOF
- Define and manage metadata in the context of *models*
- Models have types – *metamodels*
  - Relational, Web Services, XML, UML, etc
  - Some models can have multiple types (but probably one primary)
- With MOF and EMF, all model representations have a common 'master model'
  - This is the MOF meta-metamodel (EMOF) or Ecore
  - *This allows all models to be defined, edited and managed consistently, by the same software*





Modeling Multiple Types of Systems

An Integrated Model Development Environment …

Allows enterprises to define, capture, manipulate, organize, share, control, and manage

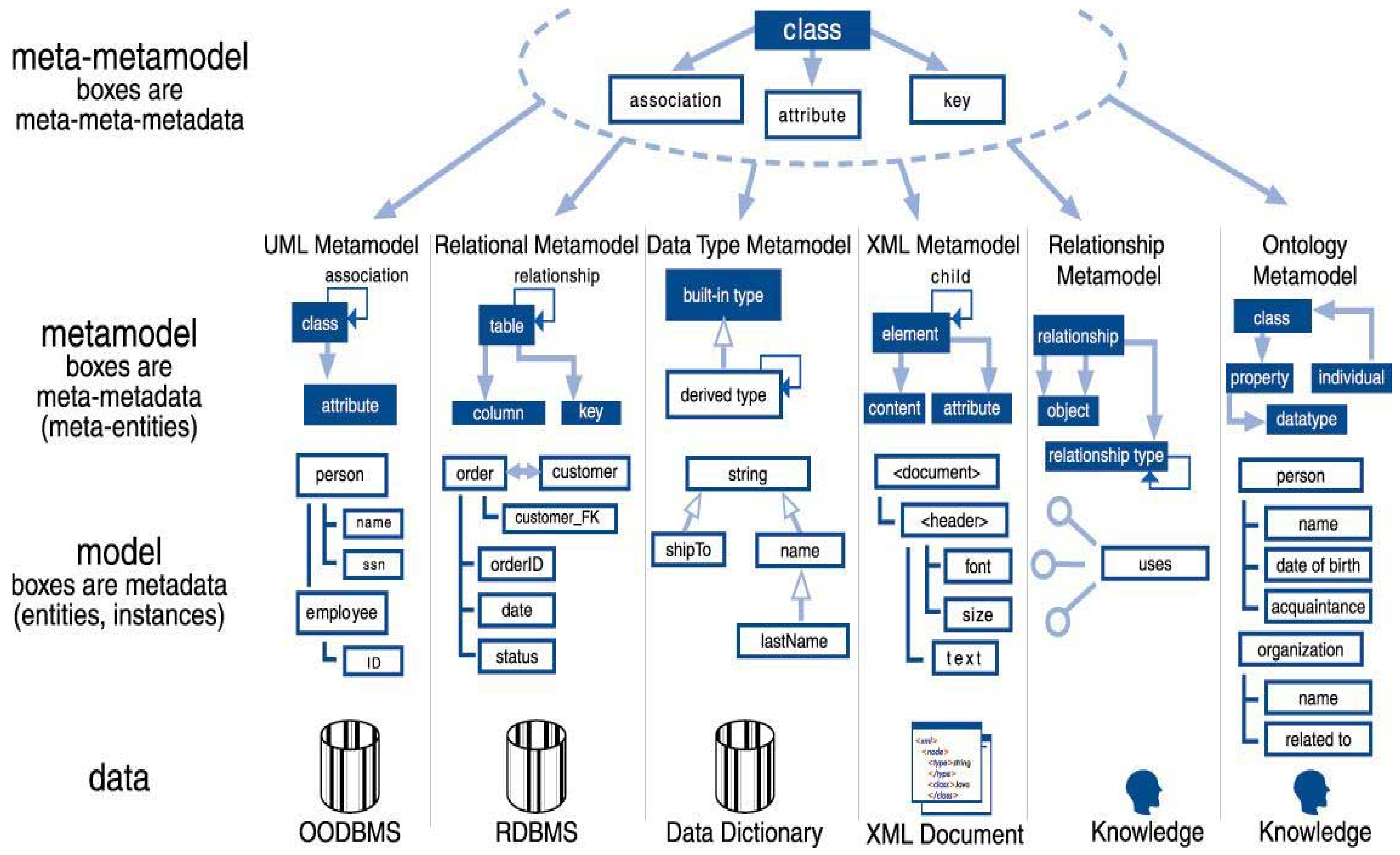
heterogeneous metadata

in a

homogenous manner

Using a Model-Based Approach

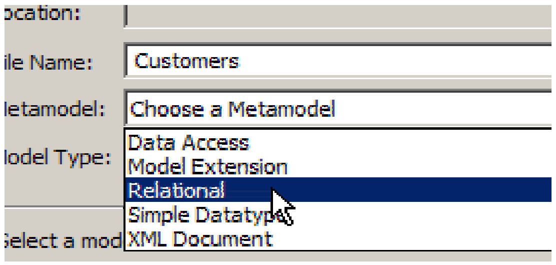
With domain-specific languages (metamodels)



Model Development Environment

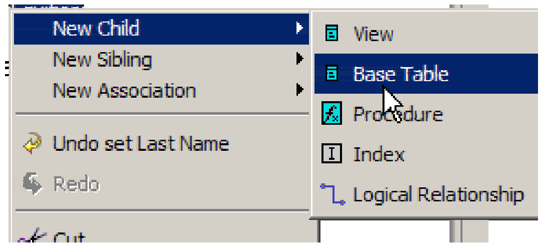
Using metamodels to drive behavior

Wizards

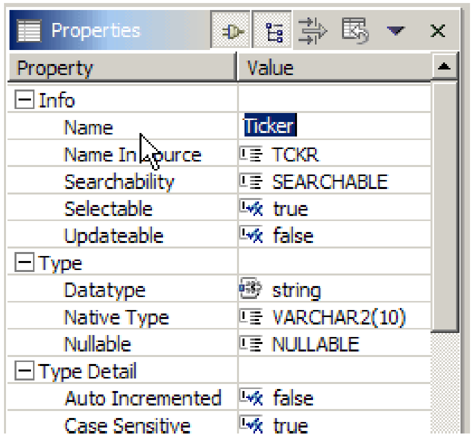


ocation:	
ile Name:	Customers
letamodel:	Choose a Metamodel
lodel Type:	Data Access Model Extension Relational Simple Datatypes XML Document
elect a mod	

Menus

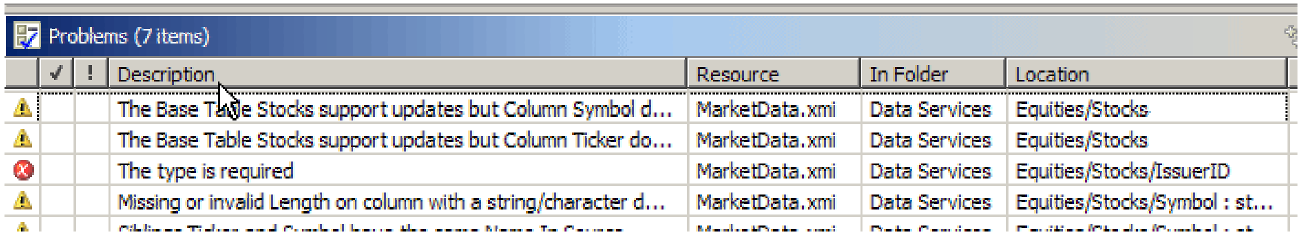


Views



Property	Value
⊟ Info	
Name	Ticker
Name In Source	TCKR
Searchability	SEARCHABLE
Selectable	true
Updateable	false
⊟ Type	
Datatype	string
Native Type	VARCHAR2(10)
Nullable	NULLABLE
⊟ Type Detail	
Auto Incremented	false
Case Sensitive	true

Validation

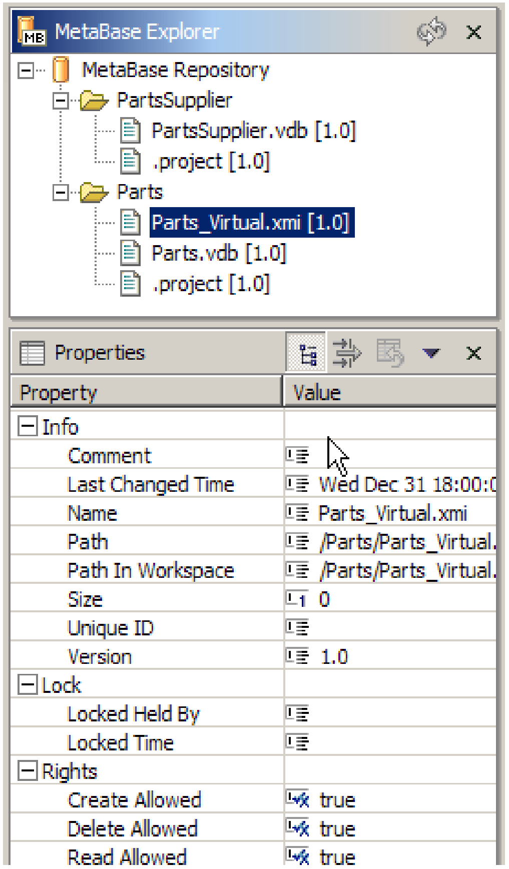


	✓	!	Description	Resource	In Folder	Location
⚠			The Base Table Stocks support updates but Column Symbol d…	MarketData.xmi	Data Services	Equities/Stocks
⚠			The Base Table Stocks support updates but Column Ticker do…	MarketData.xmi	Data Services	Equities/Stocks
⊗			The type is required	MarketData.xmi	Data Services	Equities/Stocks/IssuerID
⚠			Missing or invalid Length on column with a string/character d…	MarketData.xmi	Data Services	Equities/Stocks/Symbol : st…

Model Development Environment

Using metamodels to drive behavior

Repository Contents



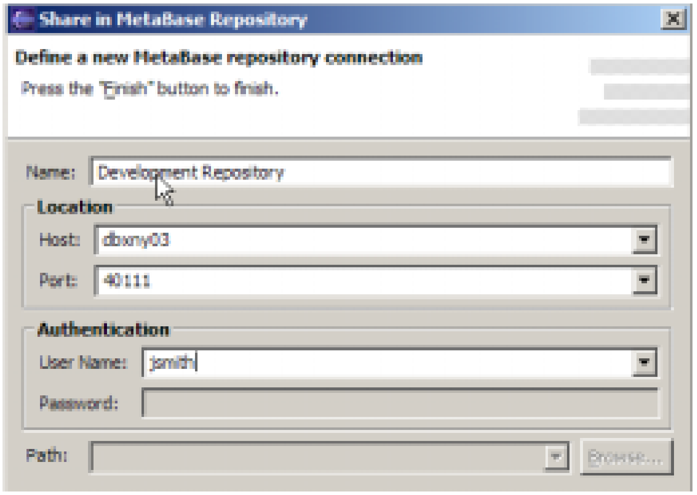
MetaBase Explorer

- MetaBase Repository
  - PartsSupplier
    - PartsSupplier.vdb [1.0]
    - .project [1.0]
  - Parts
    - Parts_Virtual.xmi [1.0]
    - Parts.vdb [1.0]
    - .project [1.0]

Properties

Property	Value
Info	
Comment	
Last Changed Time	Wed Dec 31 18:00:0
Name	Parts_Virtual.xmi
Path	/Parts/Parts_Virtual.
Path In Workspace	/Parts/Parts_Virtual.
Size	0
Unique ID	
Version	1.0
Lock	
Locked Held By	
Locked Time	
Rights	
Create Allowed	true
Delete Allowed	true
Read Allowed	true

Repository Connections



Share in MetaBase Repository

Define a new MetaBase repository connection

Press the "Finish" button to finish.

Name: Development Repository

Location

Host: dbxny03

Port: 40111

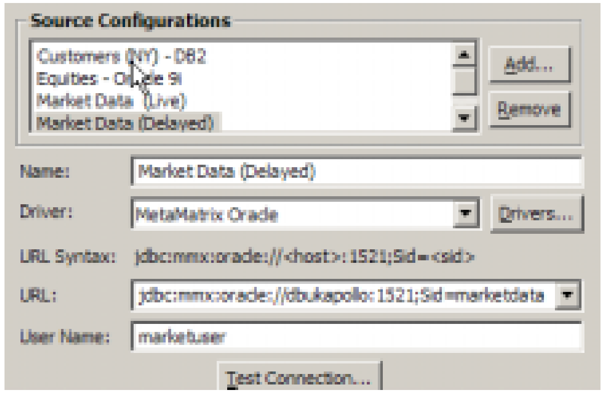
Authentication

User Name: jsmith

Password:

Path:

User information



Source Configurations

- Customers (NY) - DB2
- Equities - Oracle 9i
- Market Data (Live)
- Market Data (Delayed)

Name: Market Data (Delayed)

Driver: MetaMatrix Oracle

URL Syntax: jdbc:mmx:oracle://<host>:1521;Sid=<sid>

URL: jdbc:mmx:oracle://dbukapollo:1521;Sid=marketdata

User Name: marketuser

Model Development Environment

Extending existing metamodels

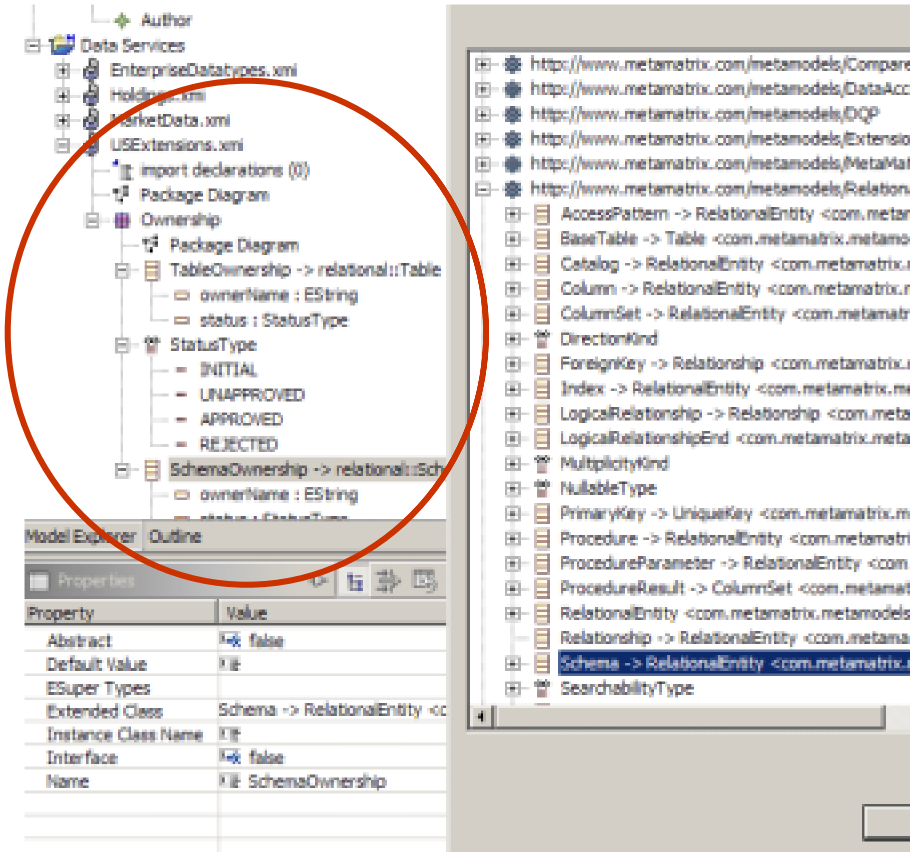
- Add custom fields (properties) to existing metaclasses
- Users can do this very dynamically
  - Extensions can be reused
  - Different models can use different extensions
  - Easily change extensions even after they are used
- Useful when an existing metamodel is structurally correct
  - Typically want do add some information
- The types of things being modeled doesn't really change
- Treated like any other property
  - Persistent, searchable, validated, etc.

Model Development Environment

Extending existing metamodels

- Define the extensions
  - Define the properties
  - Define the target metaclasses

- Additional properties appear on appropriate instances



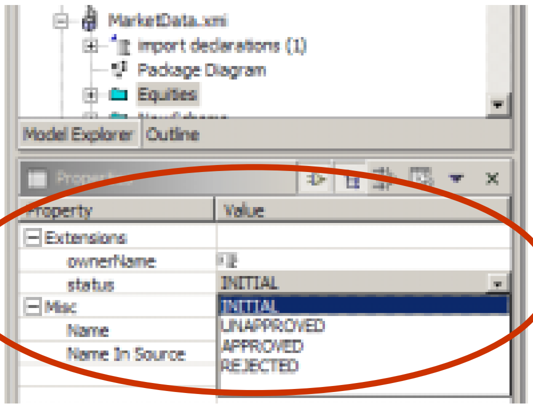
```
Author
Data Services
  EnterpriseDatatypes.xmi
  Holdings.xmi
  MarketData.xmi
  USExtensions.xmi
    import declarations (0)
    Package Diagram
    Ownership
      Package Diagram
      TableOwnership -> relational::Table
        ownerName : EString
        status : StatusType
      StatusType
        INITIAL
        UNAPPROVED
        APPROVED
        REJECTED
      SchemaOwnership -> relational::Sch
        ownerName : EString
```

Model Explorer | Outline

Properties

Property	Value
Abstract	false
Default Value	
ESuper Types	
Extended Class	Schema -> RelationalEntity <c
Instance Class Name	
Interface	false
Name	SchemaOwnership

```
http://www.metamatrix.com/metamodels/Compare
http://www.metamatrix.com/metamodels/DataAcc
http://www.metamatrix.com/metamodels/DQP
http://www.metamatrix.com/metamodels/Extensio
http://www.metamatrix.com/metamodels/MetaMat
http://www.metamatrix.com/metamodels/Relation
  AccessPattern -> RelationalEntity <com.metar
  BaseTable -> Table <com.metamatrix.metamo
  Catalog -> RelationalEntity <com.metamatrix.
  Column -> RelationalEntity <com.metamatrix.r
  ColumnSet -> RelationalEntity <com.metamatr
  DirectionKind
  ForeignKey -> Relationship <com.metamatrix.
  Index -> RelationalEntity <com.metamatrix.me
  LogicalRelationship -> Relationship <com.meta
  LogicalRelationshipEnd <com.metamatrix.meta
  MultiplicityKind
  NullableType
  PrimaryKey -> UniqueKey <com.metamatrix.m
  Procedure -> RelationalEntity <com.metamatri
  ProcedureParameter -> RelationalEntity <com
  ProcedureResult -> ColumnSet <com.metamat
  RelationalEntity <com.metamatrix.metamodels
  Relationship -> RelationalEntity <com.metama
  Schema -> RelationalEntity <com.metamatrix.
  SearchabilityType
```



```
MarketData.xmi
  import declarations (1)
  Package Diagram
  Equities
```

Model Explorer | Outline

Properties

Property	Value
Extensions	
ownerName	
status	INITIAL
Misc	INITIAL
Name	UNAPPROVED
Name In Source	APPROVED
	REJECTED

Model Development Environment

Adding new metamodels

- When the system being modeled can't be effectively and easily described using an existing modeling language
  - Constructs and syntax of system are unique or sufficiently different
  - People that are modeling the system don't need to translate concepts in their heads
- Extends the existing Modeler behavior for the new type of system
  - Same ways to create, edit, change, and refactor objects
  - Same validation framework
  - Same wizards, importers, exporters
  - Ability to reference models defined by other metamodels (e.g., relational, generalized relationships, XSD, ontologies, etc.)
- New components can always be added
  - Views and editors (e.g., diagrams, forms, etc.)
  - Wizards (e.g., importers, exporters)
  - Analysis tools

Eclipse Consortium

Partial list of members

ACTUATE.

IBM.

intel.

QNX Build a more reliable world

ERICSSON

hp invent

MONTAVISTA SOFTWARE

SAP

Accelerated Technology A Mentor Graphics Division

The new face of COBOL ACUCORP

ASC

Agitar™ SOFTWARE

Aonix

Borland

Canyon Blue

genuitec

HITACHI Inspire the Next

ILOG Changing the rules of business™

INNOOPRACT

Instantiations

JBoss™

MERCURY GLOBAL ALLIANCE PARTNER

Metanology

MICRO FOCUS

mks

mValent

Novell.

eclipse plugin central

Fraunhofer Institut für Offene Kommunikationssysteme

OMG OBJECT MANAGEMENT GROUP

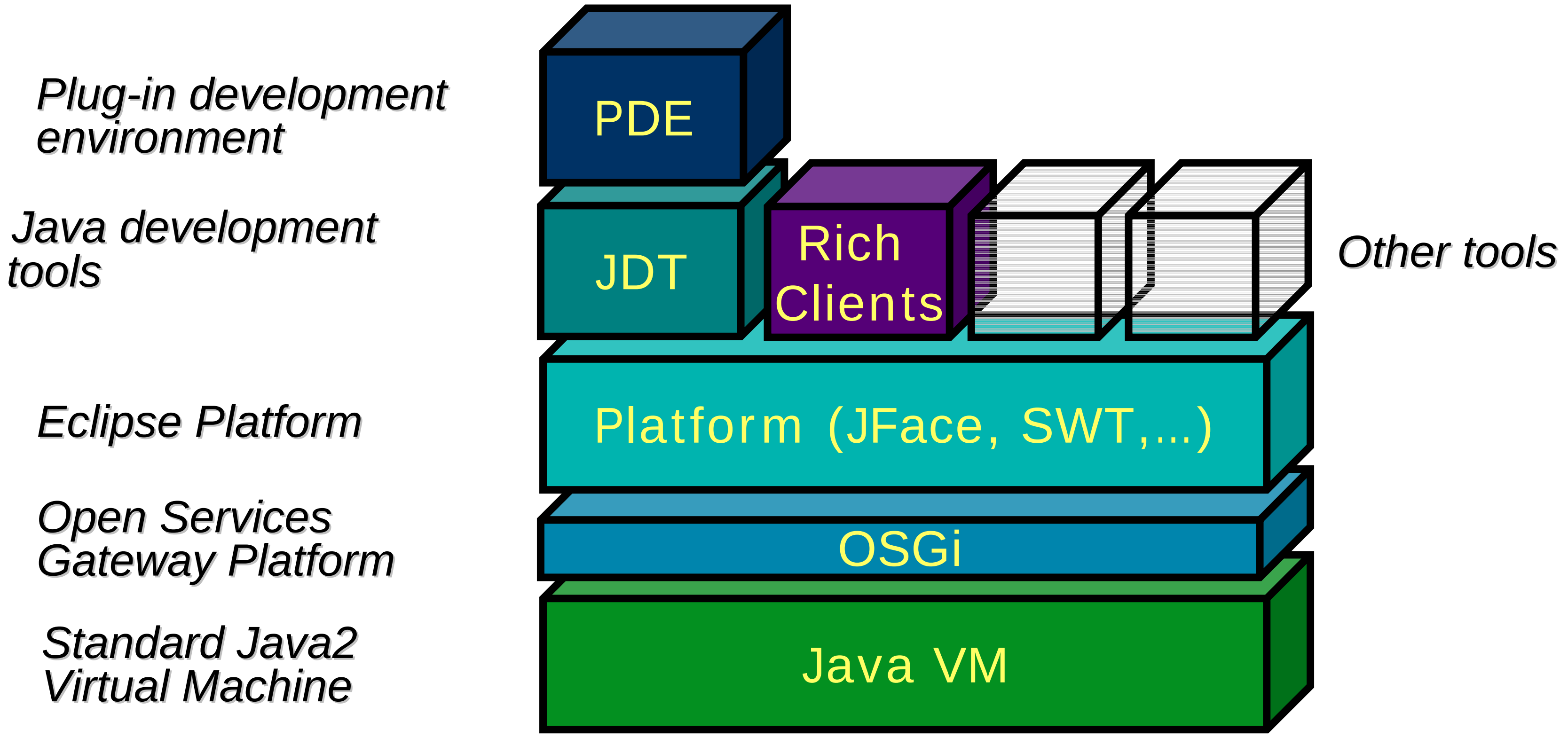
ObjectWeb Open Source Middleware

EclipseSource

Eclipse

What is it?

- Universal platform for integrating tools
- Platform for functionally-rich applications (“rich client”)
- Architecture that is open, extensible, and based on plug-ins


Plug-in development environment
PDE
Java development tools
JDT
Rich Clients
Other tools
Eclipse Platform
Platform (JFace, SWT,...)
Open Services Gateway Platform
OSGi
Standard Java2 Virtual Machine
Java VM

Eclipse

A bit of history

- In 1990's, tools weren't integrated & didn't work together
  - Many types of resources (e.g., JSPs, XML, HTML, Java …)
  - Didn't understand each others data
- Meanwhile, IBM recognized the success function had changed:
  - No longer was: "Who can build the best IDE?"
  - Now is: "How do you provide a truly integrated set of tools?"
- So Eclipse was born – a project to create a universal tool platform
  - Started in 1999
  - Open sourced in 2001
  - 2.0 shipped June 2002
  - 3.0 shipped June 2004

What makes Eclipse Different?

Eclipse is “platform-centric”, not “tool-centric”

- Tool boundaries (as visible to user) disappear
- Platform has many standard and reusable interface components for performing common functionality
  - file management, repository integration, editors, view management, update manager, etc.
- Tools can be added at any time – and (with 3.0) they can be added even while running!
- Tool developers focus on their domain rather than “plumbing”, and rely upon other tools built by experts in other domains
- *This makes it an excellent platform for an integrated model development environment!*

Eclipse

Standard tools

- **Views** commonly needed by different tools
  - Workspace navigation and file resources
  - Properties, tasks and problems
- **Editors** for different types of files
  - Java, C#, C++, .properties, XML, XSLT, XSD, JSP, etc.
- **Wizards** to help create various files and associated projects
  - Creation of new files and artifacts
  - Importers and exporters
  - Refactoring and quick fixes
- **Builders** to “compile” artifacts
  - Java compiler, C++ compiler, JSP compiler, etc.
- **Preferences** for customizing behavior
- **Tools** for managing the installed components
  - Update manager to find, install, and manage tools

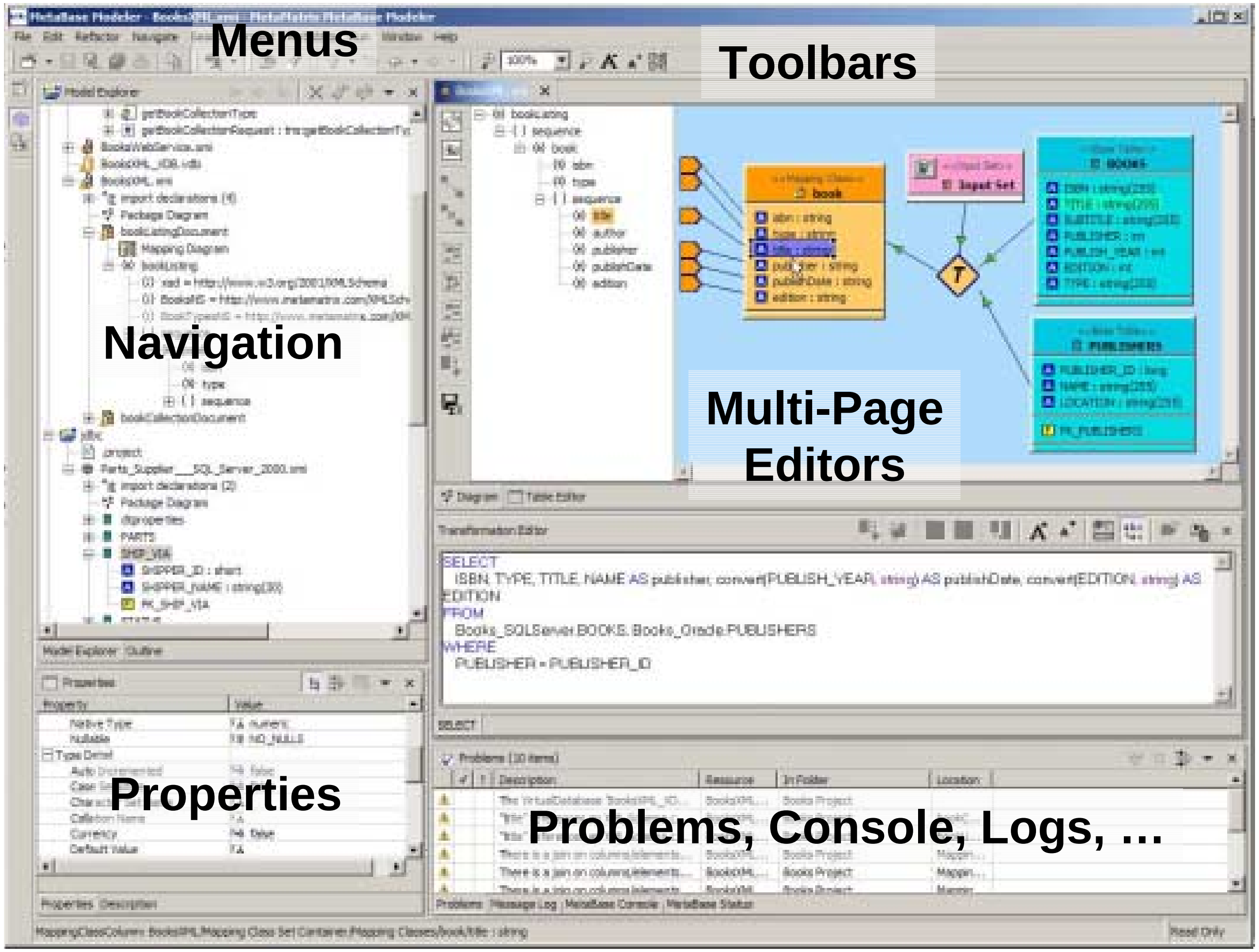
Model Development Environment

Domain-specific and domain-independent functionality

- General modeling functionality
  - Standard tree, table, and property views
  - Create, edit, delete, clone, copy/paste, move, undo/redo
  - Types of objects in model dictated entirely by metamodel
  - Standard model persistence (defaults to XMI, but is customizable)
  - Unified and common places for description and custom properties
  - Rule-based validation of model content identifies areas of models that are incorrect or problematic
  - Compare two models (or model versions) and view differences
  - Find and search for objects meeting criteria
- Extensible importers, exporters and wizards
  - Standard frameworks for automated loading, exporting and creating models
- Integrated diagramming
  - UML class diagrams; requires adapting metamodel into UML-like structures
  - Ability to add other diagram types via metamodel and UI code

Model Development Environment

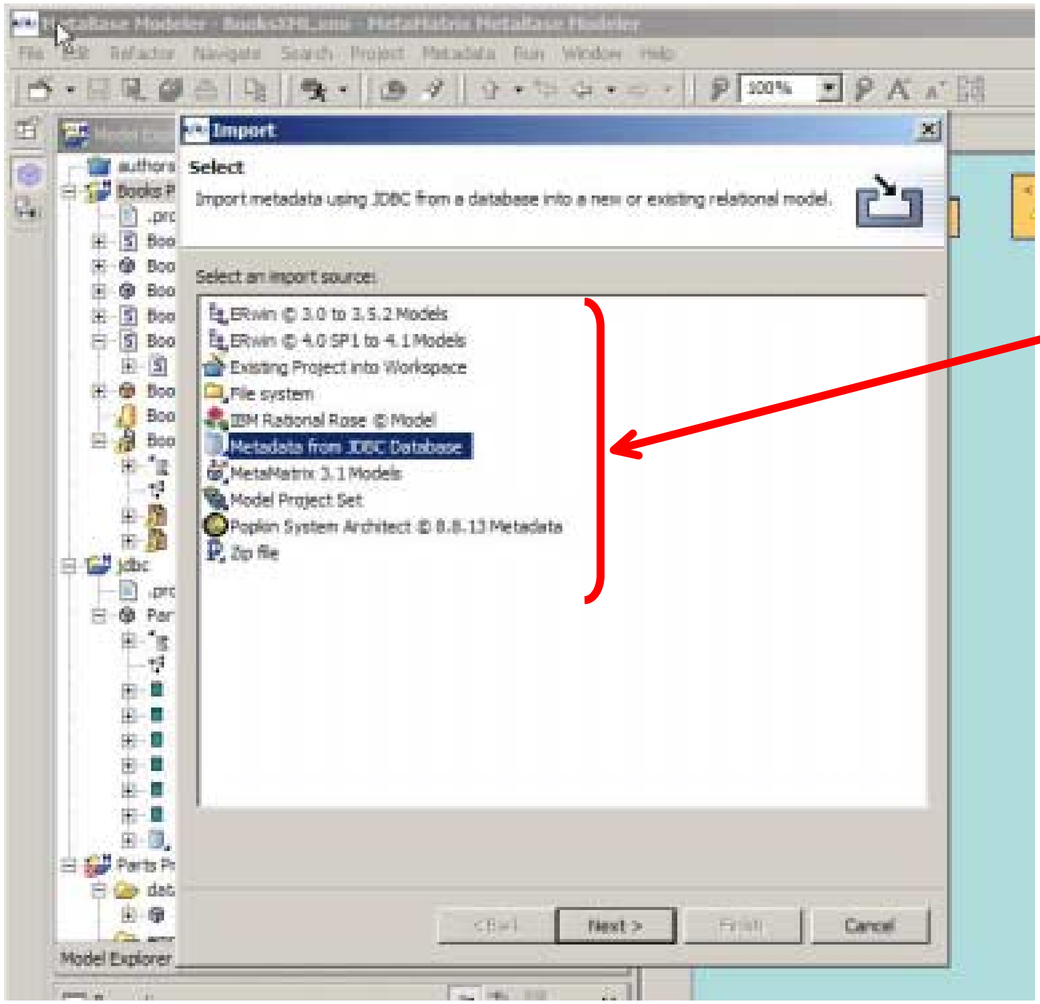
Eclipse-based integrated toolset



```
SELECT
  ISBN, TYPE, TITLE, NAME AS publisher, convert(PUBLISH_YEAR, string) AS publishDate, convert(EDITION, string) AS
EDITION
FROM
  Books_SQLServer.BOOKS, Books_Oracle.PUBLISHERS
WHERE
  PUBLISHER = PUBLISHER_ID
```

Model Development Environment

Contributing tools, wizards and functionality


File Edit Refactor Navigate Search Project Metadata Run Window Help
100%
Model Explorer
< Back
Next >
Finish
Cancel

Import

Select

Import metadata using JDBC from a database into a new or existing relational model.

Select an import source:

- ERwin © 3.0 to 3.5.2 Models
- ERwin © 4.0 SP1 to 4.1 Models
- Existing Project into Workspace
- File system
- IBM Rational Rose © Model
- Metadata from JDBC Database
- MetaMatrix 3.1 Models
- Model Project Set
- Popkin System Architect © 8.8.13 Metadata
- Zip file

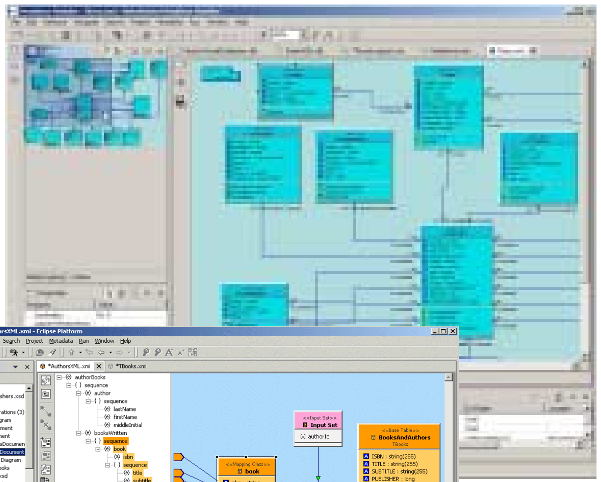
Some of these are provided by Eclipse, other are added in by plug-ins

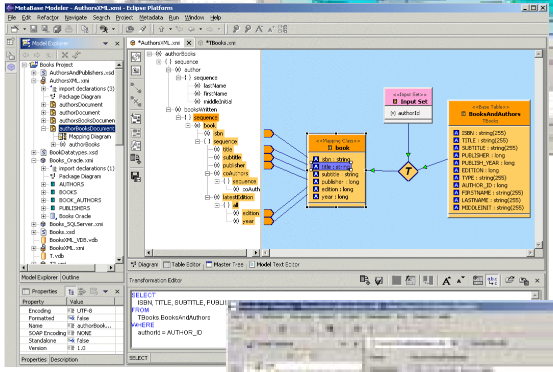
The "importer" functionality shows up in the right place

Model Development Environment

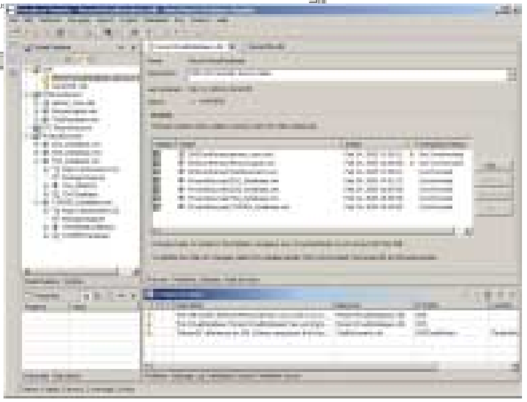
Modeling disparate information

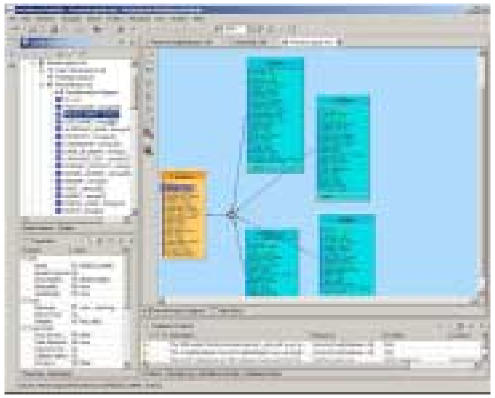
- Multiple, domain-specific modeling languages
  - Based upon MOF
  - Use the modeling language best suited for the system being described
  - Leverage native artifact structures (e.g., DDL, WSDL, XSD, OWL, etc.)
- Integrated but extensible environment
  - Eclipse plug-in platform
  - EMF, UML2, XSD and other Eclipse plug-ins

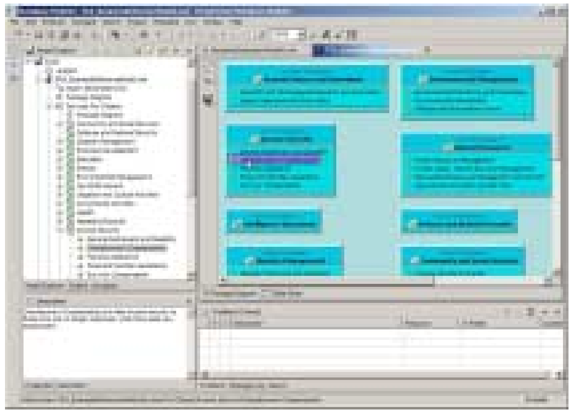




```
SELECT
  ISBN, TITLE, SUBTITLE, PUBLIS
FROM
  TBooks.BooksAndAuthors
WHERE
  authorId = AUTHOR_ID
```

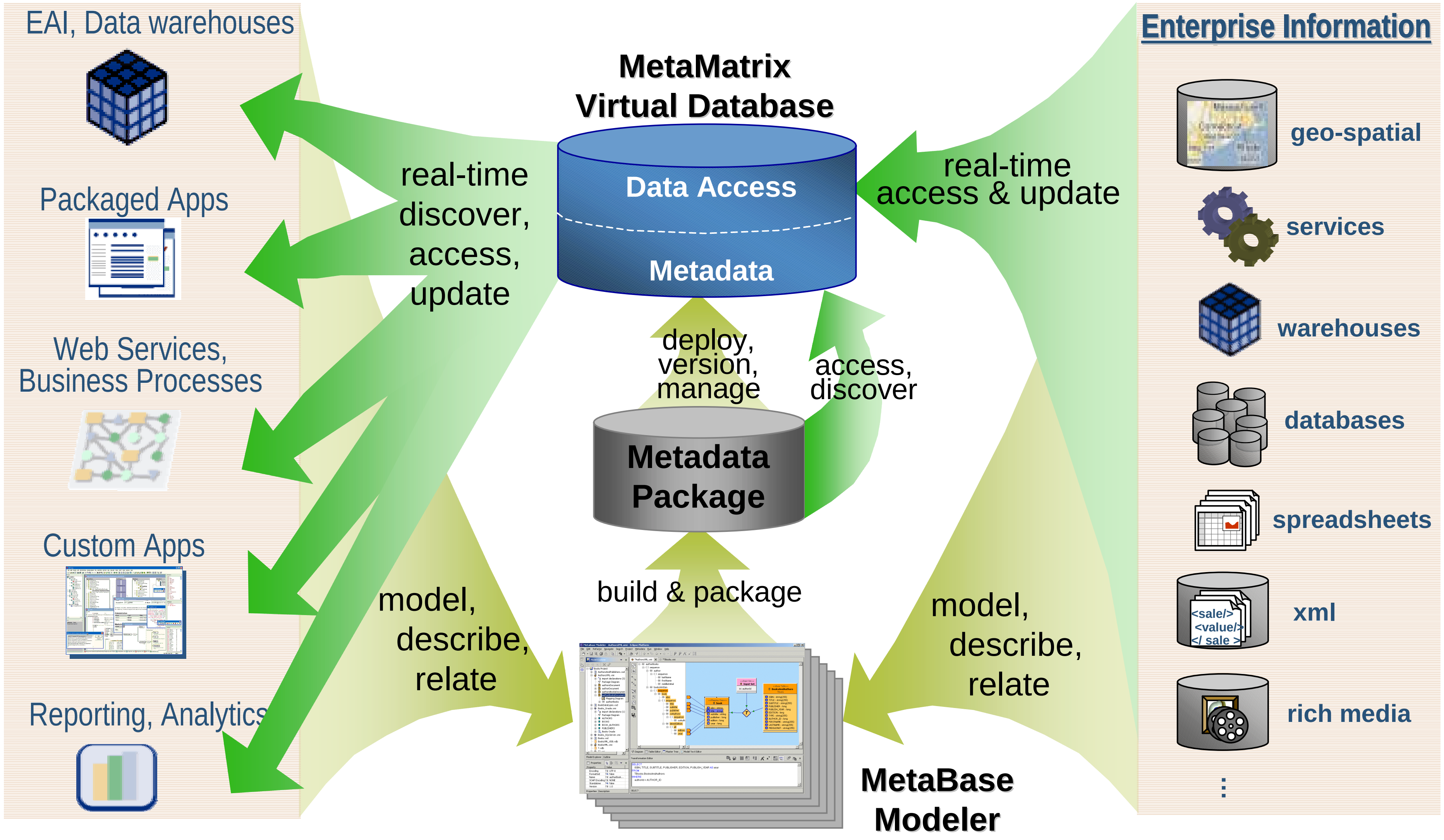






Use the Models to Drive Systems

Deploy models to enable real-time integration



Summary

Eclipse has changed the landscape

- Changed the open-source community
  - Major backing by commercial companies
  - Established a large community of organizations that are
  - Sub-communities to develop common and domain-specific plug-ins
- Changed the business model for tool providers
  - Companies that used to compete with each other now collaborate on the overlapping functionality and each can specialize in distinct areas where they add value
  - Lowers the bar; companies can build on top of already large set of functionality
- Changed the Integrated Development Environment (IDE)
  - Free and very powerful Java IDE is a best-in-breed
  - Allows very small tools to be added in, yet not a proprietary platform
- Changed the Java community
  - The SWT is a UI toolkit that leverages OS platform resources

Summary

MOF and MDA™ have changed the landscape

- One modeling language is no longer the ideal
  - Domain-specific languages are easily understood by users
  - Multiple domain-specific languages enables modeling multiple systems accurately, precisely and in homogeneous ways
- Models are not just for documentation anymore
  - Models are valuable artifacts and reusable components
  - MDA™ is not just for generating code
  - Existing systems drive their behavior at runtime from models
- Enterprise information includes corporate data and metadata
  - Information consumers need integrated enterprise information and transparent access to the enterprise metadata

Abstract

Information and data is represented many ways, even as the same information is passed from system to system. Tracking the technical and semantic details about how information is used and shared throughout an enterprise is challenging but more important than ever. Formally modeling the information provides significant benefits, but has been difficult in the past because of the different types of information systems. This talk will describe how a single modeling environment can use domain-specific metamodels and open-source components to make it easy for people to describe, visualize, modify, discover, track, relate, and make better use of their enterprise information.

About the Presenter

Randall Hauch is the Chief Architect at MetaMatrix, the leading provider of Enterprise Information Integration (EII) middleware to access and integrate in real time disparate information sources. He has been working with Java technologies for the past 8 years, and with application development for over 10 years. After receiving a BS and MS in Aerospace Engineering, he developed engineering models and engineering applications for aerospace systems. He has been at MetaMatrix for 5 years, and has helped define and build the company's modeling infrastructure and metadata management capabilities, including the Eclipse-based modeling tool. Randall is also the MetaMatrix representative to the Object Management Group (OMG), where he has participated on various task forces, including MOF, XMI, CWM, MOF2 and UML2 Infrastructure.