



# Discover, Relate, Model, and Integrate Data Assets with Rational Data Architect

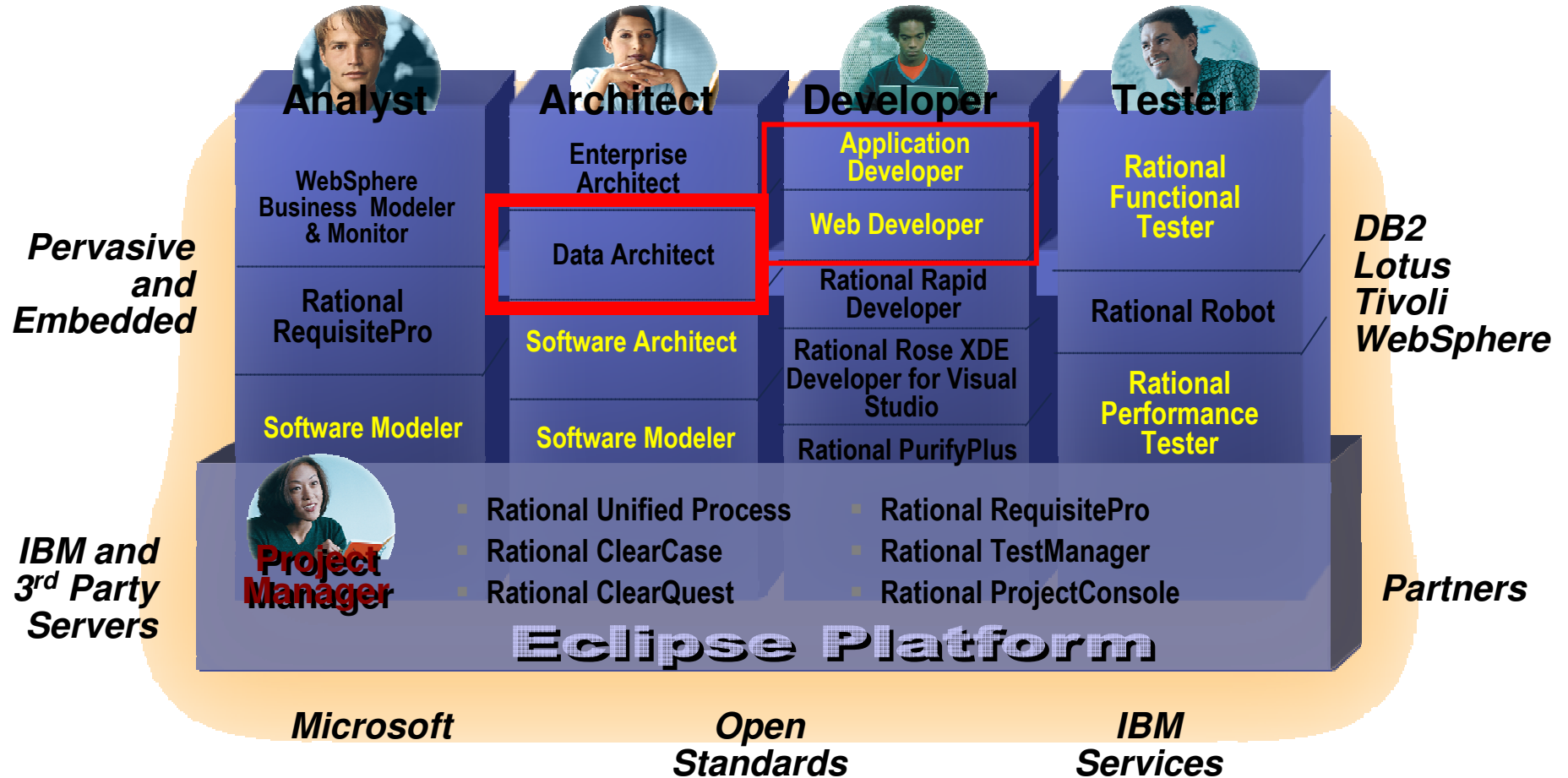
Niels C. Jacobsen (nielsj@dk.ibm.com)  
Associate IT Architect, IBM Software Group – Rational

IBM Software Group



*Move Faster*

# The IBM Software Development Platform: 2006 View



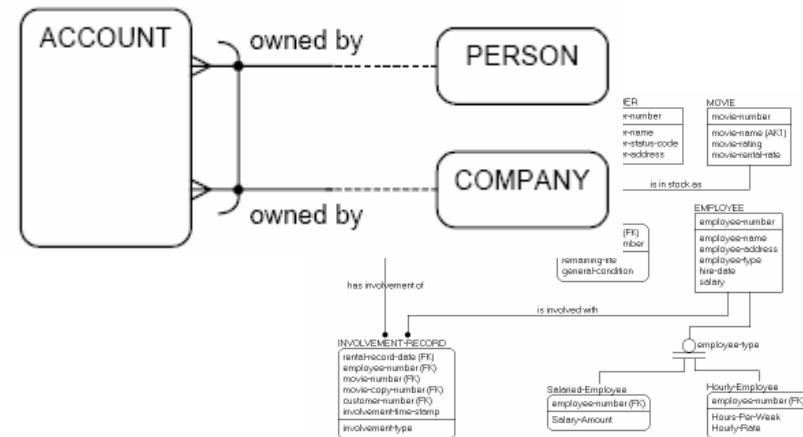
\* Yellow denotes Q4 2004 Offerings

## Agenda

- What is Rational Data Architect
- The benefits of design
- Discover your data sources
- Relate disparate data sources
- Integrate data assets
- Demonstration

# Use of modeling notations

- UML (10%)
- IDEF1X (20%)
- Barker (20%)
- IE (50%)



Notation	Information Engineering	Barker Notation	IDEF1X	UML
<b>Multiplicities:</b>				
- Zero or one				
- One only				
- Zero or more				
- One or more				
- Specific range	N/A	N/A	N/A	



## What is Rational Data Architect?

- Tool
  - Helps you do work easier, faster, and more precisely
- For data architects and advanced DBAs
  - Contains functionality needed for those roles
    - Discovers data structures
    - Models new information schemas
    - Visualizes existing data sources
    - Relates models to each other
    - Documents plans and realization



## The benefits of design

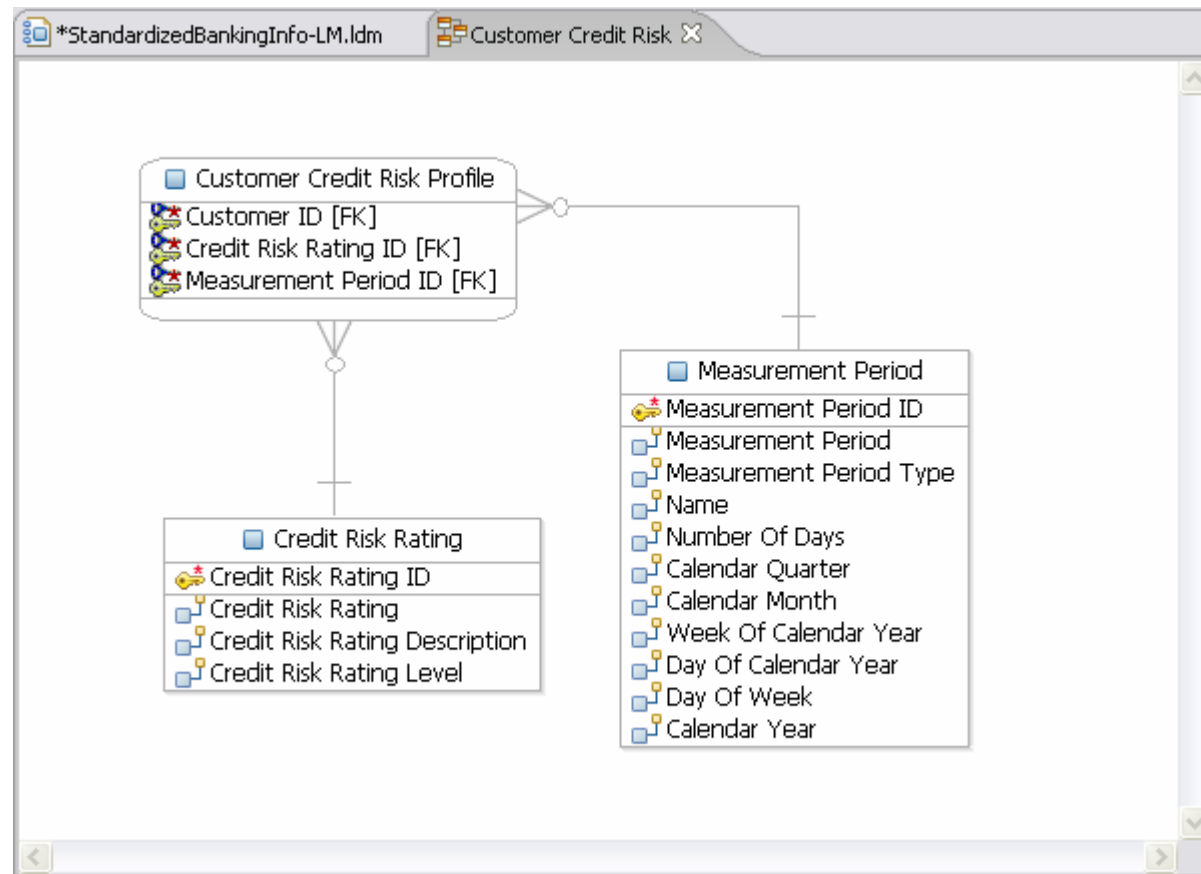
- Time to market
  - Clear communication of information requirements to team members
  - Explore different design approaches
  - Reuse existing design in new environments
  - Reliable and simplified change management
- Standardization
  - Implement corporate standards
  - Define standard tool platform



## Clear communication of information requirements

- 📄 Printed documentation is not synchronized with current requirements and current implementation
- 📄 Synchronized models represent the current status of requirements and implementation
- 📄 Models can be quickly shared throughout the team, allowing faster turn-around cycles
- 📄 Implementation itself is too detailed for broader audience
- 📄 Diagrams display just the right level of context-relevant information
- 📄 Reduces search time for the right information

# Clear communication of information requirements with RDA



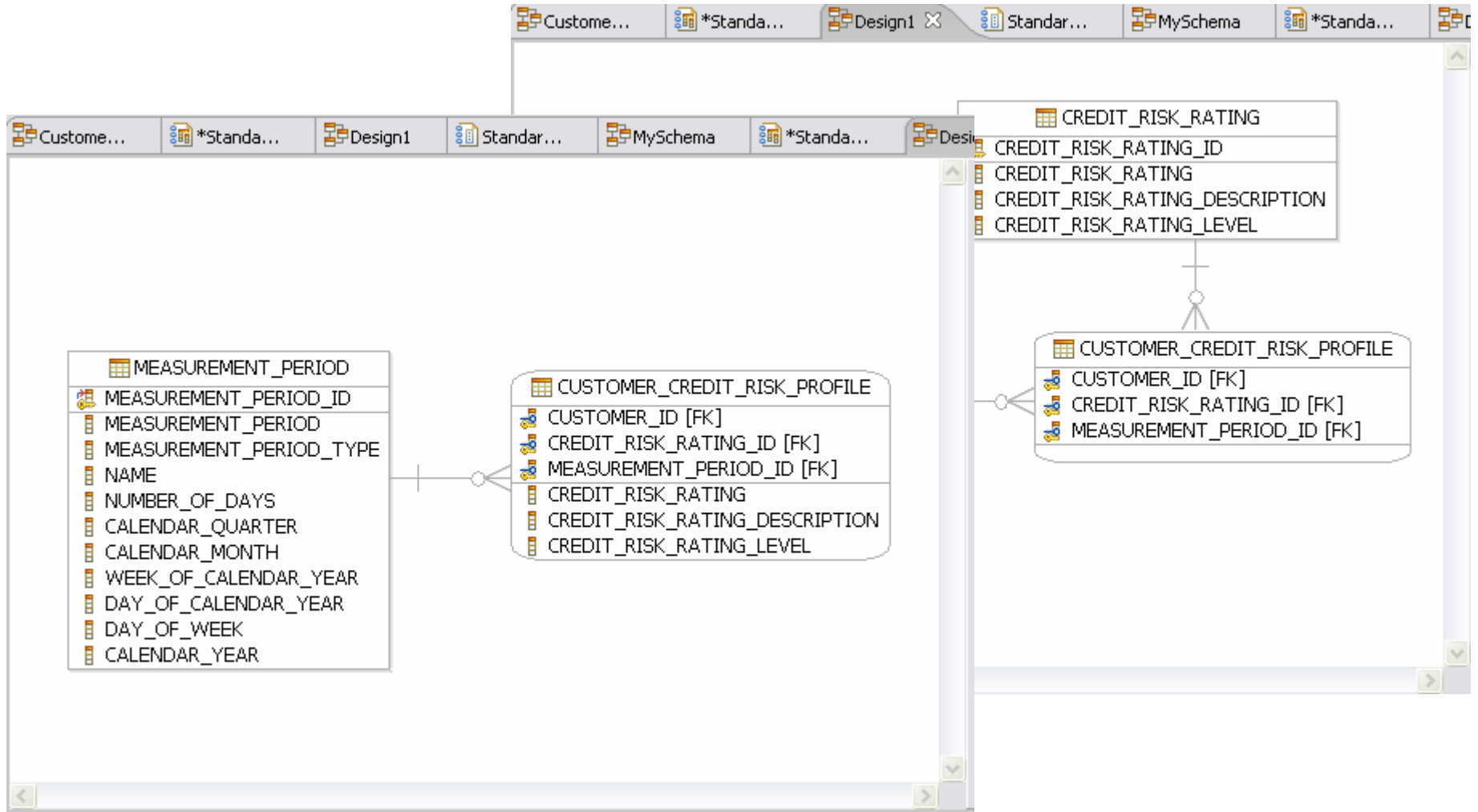


## Explore different design approaches

- 👤 Early decision on design does not satisfy all stakeholders
- 💡 Development of different design approaches
- 📊 Reduced need for costly implementation to proof the concept



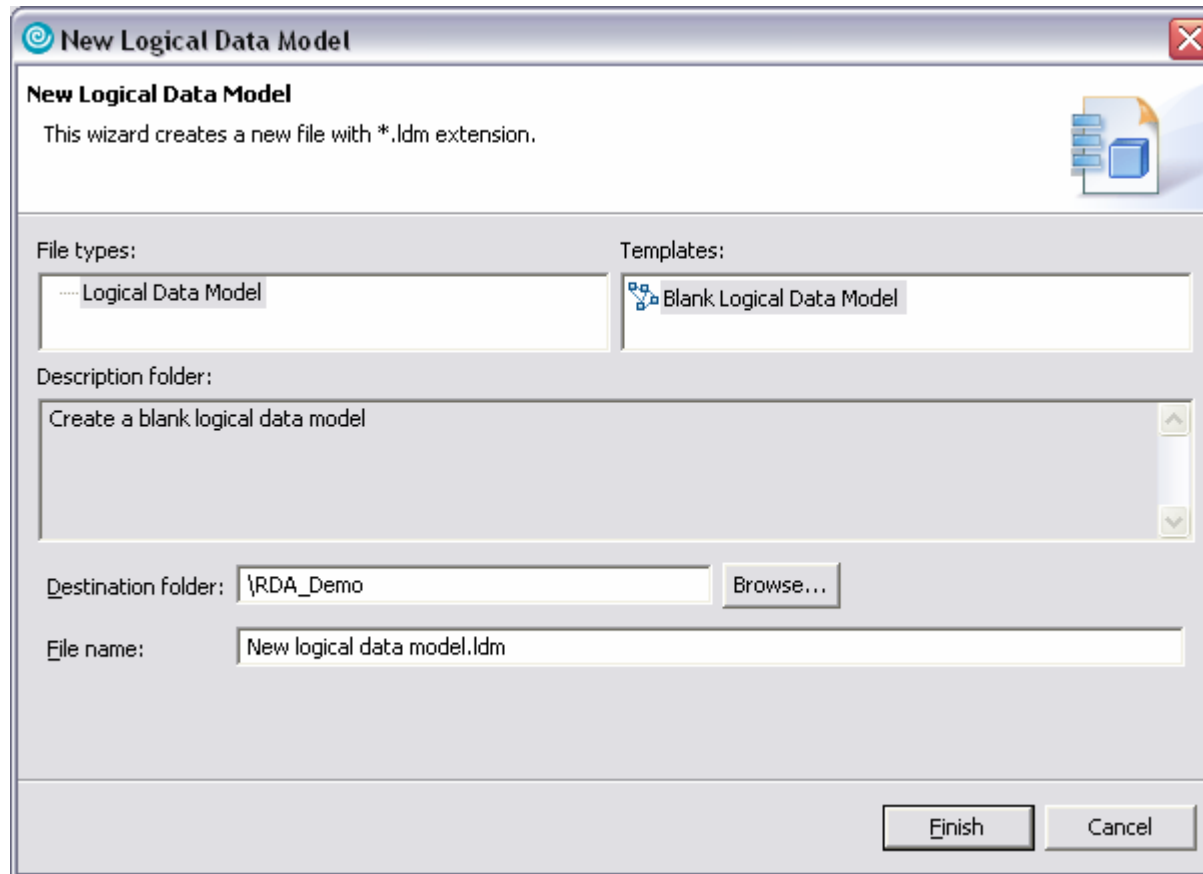
# Explore different design approaches with RDA



## Reuse existing design in new environments

- 🛠️ Reinventing the wheel over and over again
- 💡 Reuse design concepts from existing solutions
- 📊 Save time and increase quality with reuse
- 🛠️ New team members need intense introduction to new projects
- 💡 Defining standards for design and implementation allows team members to work more easily in a new team environment
- 📊 Dynamic resource allocation reduces stress on project management

# Reuse existing design in new environments with RDA

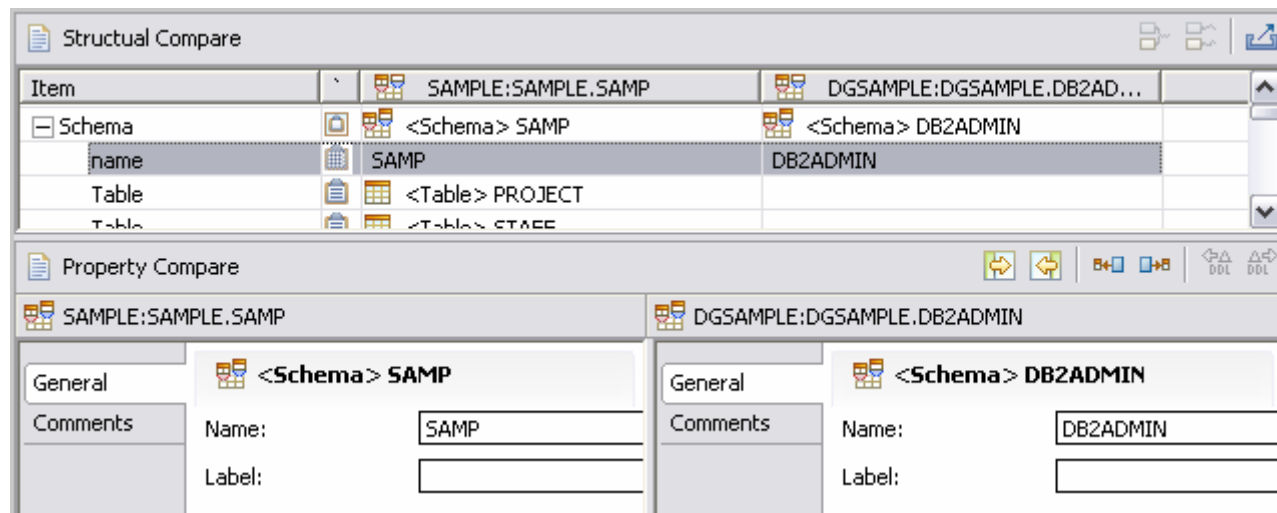


## Reliable and simplified change management

- 👤 Small changes cause big delays in projects
- 👤 Impact analysis lets you understand what changes in the project are required
- 📊 Better estimates of the real effort of changes allows reliable planning
- 👤 Implementation is not current with the design; design is not current with requirements, etc.
- 👤 Promote changes between models and implementation
- 📊 Synchronized requirements, design, and implementation reduce mistakes

## Change management

- Compare two models, model and database, or two databases
- Synchronize and generate DDL or update model

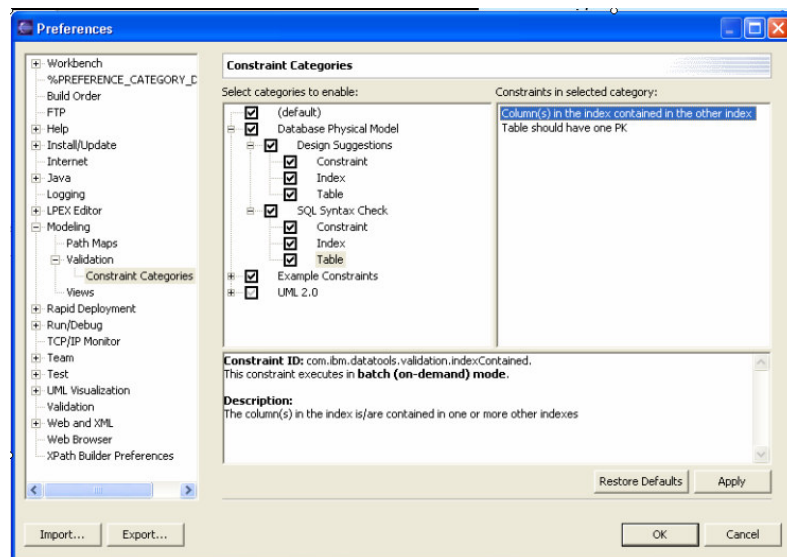


## Implement corporate standards

- 👤 It is impossible to understand alignment of data sources with corporate business rules
- 💡 Define standardization for design and implementation as executable business rules
- 📊 Quick check of conformance to the corporate rules will let you know how good you are
- 👤 It is impossible to understand used names in models and implementations
- 💡 Define glossaries and naming standards
- 📊 Standardized names allow you to understand design and implementation more quickly

## Implement corporate standards with RDA

- Rule-driven compliance checking
  - Operate on models or directly on database
  - User-extendable rule set (Java, OCL) in the future
- Design and normalization
  - Discover 1<sup>st</sup> 2<sup>nd</sup> 3<sup>rd</sup> normalization
- Index and storage
  - e.g., Check for excessive index
- Naming standards
- SQL syntax checks
- Model syntax checks





## Define standard tool platform

- 🛠️ Isolated tools require complex and costly integrations
- 👁️ Use tools based on widely adopted technology
- 📊 Reuse solutions developed by a big users community
- 🛠️ Users need intense training to use tools
- 👁️ Use tools that have similar user experience model than accepted practices
- 📊 Reduce the time new users need until they are productive with a tool

# Define standard tool platform with RDA

The screenshot displays the Eclipse IDE interface for a database schema. The main workspace shows a diagram with three tables: **MSRMT\_PERIOD**, **ACCT\_BAL**, and **CUS**. Relationships are indicated by lines connecting the tables. The **MSRMT\_PERIOD** table has columns: MSRMT\_PERIOD\_ID, MSRMT\_PERIOD, MSRMT\_PERIOD\_TYPE, NAME, NUM\_OF\_DAYS, QQ, MM, WK\_OF\_YY, DAY\_OF\_YY, DAY\_OF\_WK, and YY. The **ACCT\_BAL** table has columns: ACCT\_ID, ACCT\_TYPE, CUST\_ID [FK], MSRMT\_PERIOD\_ID [FK], and NET\_AMT. The **CUS** table has columns: CUST\_ID, NAME, CUST\_MKTG, CUST\_RLTN, EFF\_CUST\_D, END\_CUST\_I, and PROVISION\_I.

Callouts on the left side of the IDE identify the following components:

- Project Explorer**: Located at the top left, showing a tree view of the project structure.
- Outline View**: Located in the middle left, showing a hierarchical view of the current object.
- Server Explorer**: Located at the bottom left, showing a tree view of the database server and its schemas.

Callouts on the right side of the IDE identify the following components:

- Context-Specific Editor / Diagram**: Points to the main workspace area where the schema diagram is displayed.
- Properties Editor**: Located at the bottom right, showing the configuration for the selected diagram (e.g., Diagram name: Schema1, Diagram notation: IE (Information Engineering)).
- Other Information**: Points to the bottom status bar area.

## Discover your data sources

- Explore the structure of your data sources
- Sample the data of your data source
- Visualize data sources

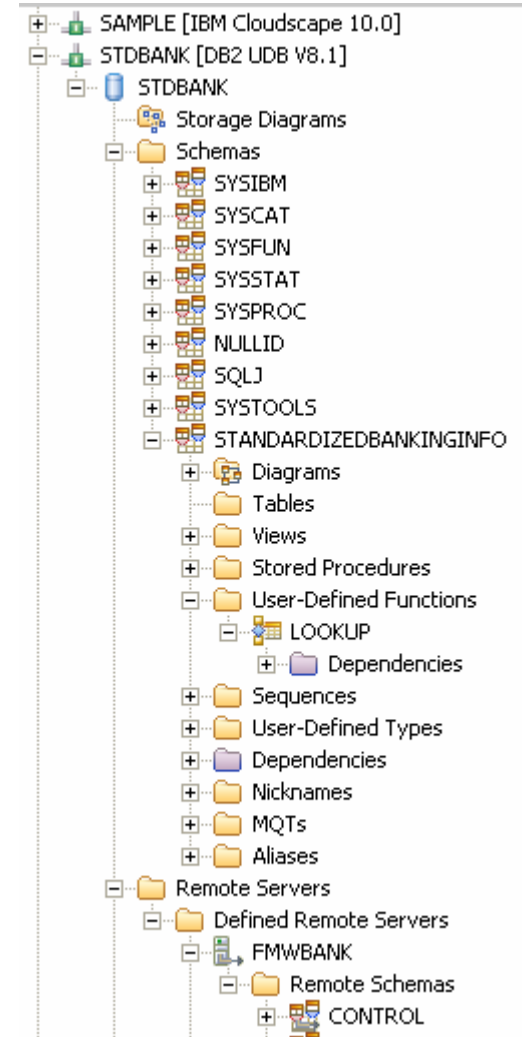


## Explore the structure of your data sources

- 🔍 Many data sources are not documented
- 🔍 Use the structure of the data source, and display it in understandable format
- 📊 By reducing dependency on the technology of the data source, increase the number of team members understanding it
- 🔍 Without RDA, each data source needs a separate tool
- 🔍 Use tools that can connect to many data sources at the same time
- 📊 Reduce the need for several tools and expensive maintenance for each of them

# Explore the structure of your data sources with RDA

- Server discovery
- JDBC connections
- Information request from the database on demand
- Database structure
- Display properties for selected elements



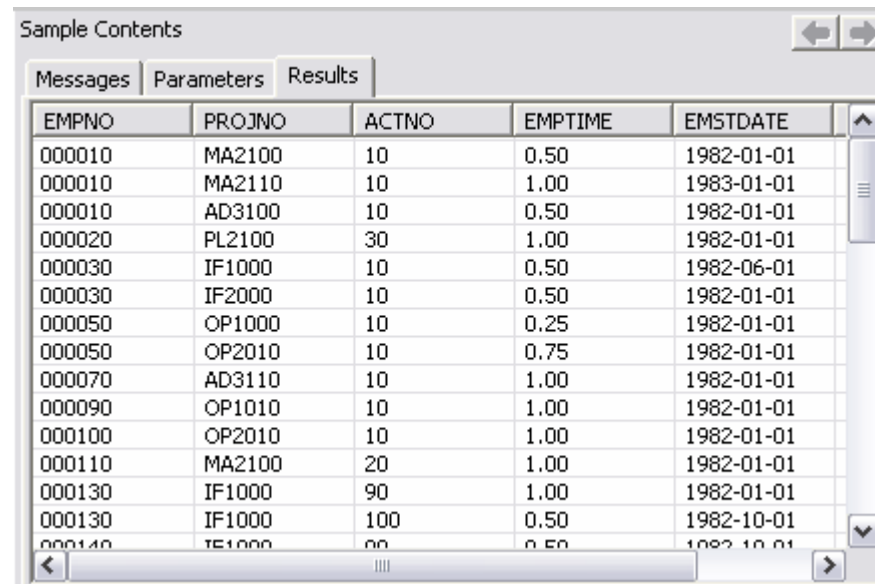
## Sample the data of your data source

- 👤 Documentation is not precise enough and allows freedom of interpretation
- 💡 Sample existing data from the data source to confirm your understanding
- 📊 Reduce costs of change by fewer misinterpretations of specifications



## Sample the data of your data source with RDA

- Sample data for increased understanding
- Edit data from the data source to create your test environment









Sample Contents

Messages Parameters Results

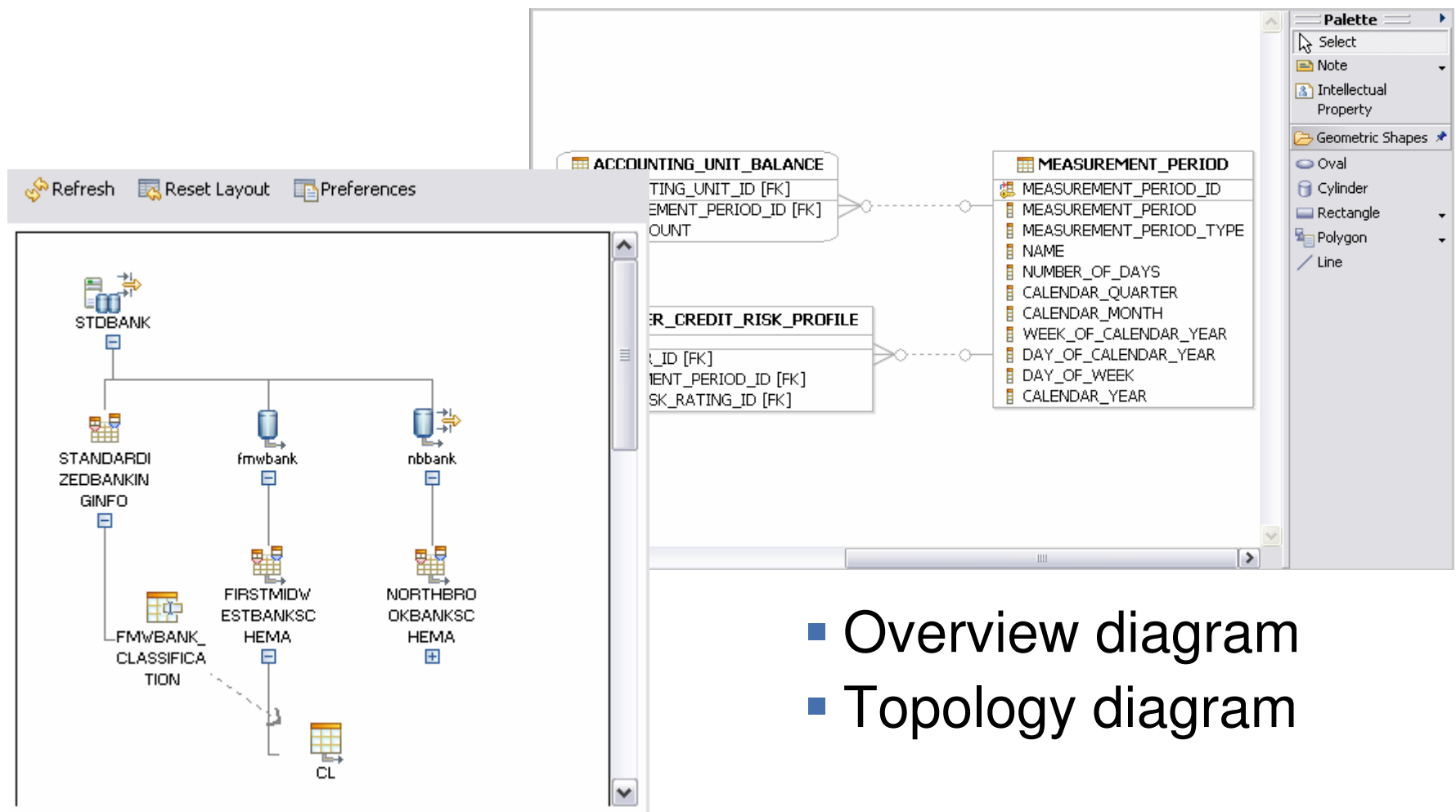
EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE
000010	MA2100	10	0.50	1982-01-01
000010	MA2110	10	1.00	1983-01-01
000010	AD3100	10	0.50	1982-01-01
000020	PL2100	30	1.00	1982-01-01
000030	IF1000	10	0.50	1982-06-01
000030	IF2000	10	0.50	1982-01-01
000050	OP1000	10	0.25	1982-01-01
000050	OP2010	10	0.75	1982-01-01
000070	AD3110	10	1.00	1982-01-01
000090	OP1010	10	1.00	1982-01-01
000100	OP2010	10	1.00	1982-01-01
000110	MA2100	20	1.00	1982-01-01
000130	IF1000	90	1.00	1982-01-01
000130	IF1000	100	0.50	1982-10-01
000140	IF1000	00	0.50	1982-10-01

## Visualize data sources

-  No one understands the complete structure of complex data sources
-  Create topological views of the data source
-  Find problems faster with understanding of the complete data source including storage
-  Data source complexity requires long search times to find relevant structural information
-  Display relational diagrams for quick overview of the structure
-  Reduce time needed to find context-related information



# Visualize data sources with RDA



- Overview diagram
- Topology diagram

## Relate disparate data sources

- Compare the structure of two data sources
- Discover similarities between data sources
- Relate data sources to each other



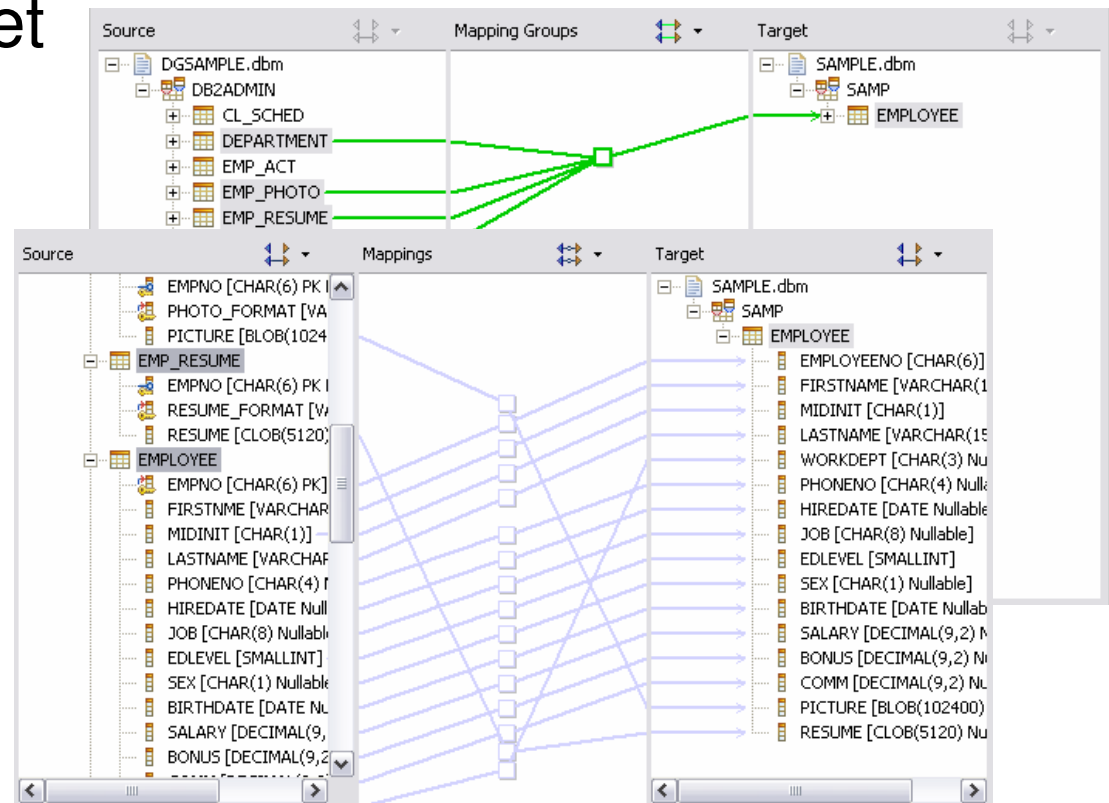
## Compare the structure of two data sources

- 👤 Replicated information is distributed in many data sources using different formats
- 💡 Compare two data sources on the attribute level, and define how they map to each other
- 📊 Understanding which information is where and which data source is the steward for it increases precision of information



# Compare structure of two data sources with RDA

- Visualizes and defines mappings between source and target
  - Column based
  - Table based



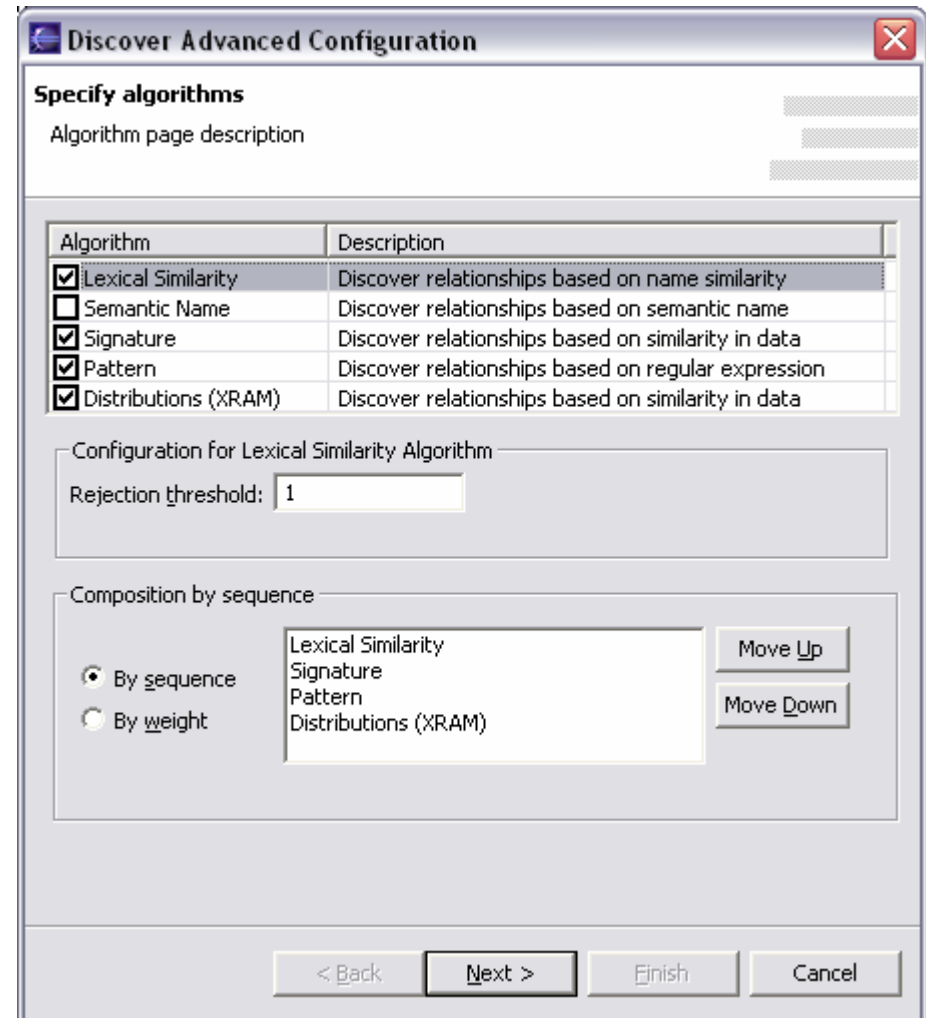
## Discover similarities between data sources

- 🤖 Understanding how two independent data sources relate to each other is a tedious manual process
- 💡 Use discovery tools to find suggestions for mappings
- 📊 Automation of mapping discovery makes mapping specification for bigger models easier and executable



## Discover similarities between data sources with RDA

- Identifies possible mappings
  - Different algorithms
  - Combination of algorithms
  - Suggests mappings
  - User can confirm or modify and annotate suggestion



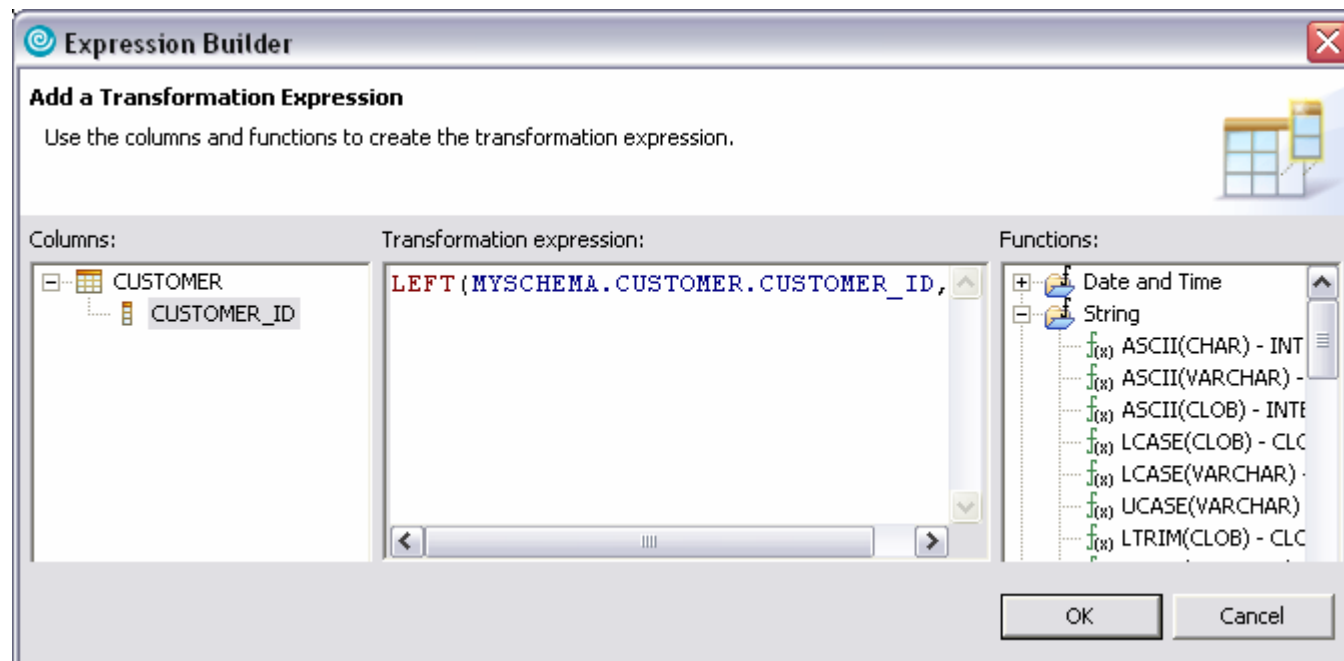
## Relate data sources to each other

- 🤖 Mappings between two data sources are very complex to specify
- 💡 Specify transformation expression between data sources for each attribute
- 📊 Exact transformation function makes mapping specifications uniquely understandable and allows less room for misinterpretations



## Relate data sources to each other with RDA

- Use any transformation expression valid in your environment





## Integrate data assets

- Integrate data assets to one federated database
- Create federated database elements



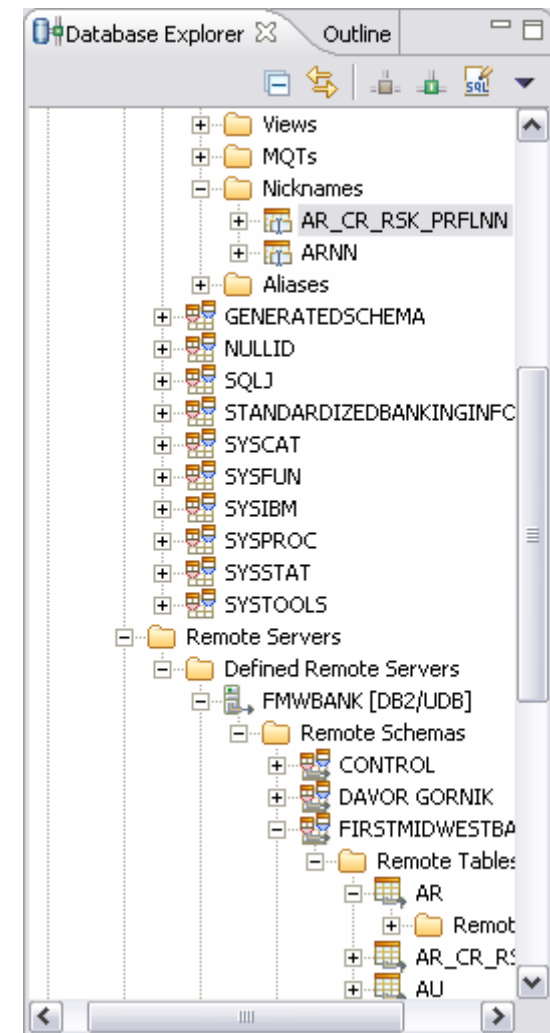
## Integrate data assets to one federated database

- 👤 Business-relevant information is distributed in many data sources
- 💡 Define the representation of remote databases and remote database objects in the federated database
- 📊 Increased understanding of remote databases and objects simplifies specification of federated schema









## Integrate data assets to one federated database with RDA

- Display remote database servers
  - Understand remote schemas
  - Research remote elements
- Explore dependent elements
  - Nicknames
- Find local elements based on remote elements
  - Views



## Create federated database elements

-  Manual, error-prone coding is required to define federated elements
-  Generate code out of the specified transformation between source and target schema
-  Speed the implementation
-  Testing requires work with many data sources at the same time
-  Validate design before deployment
-  Reduce hardware resources with design validation

## Create federated database objects with RDA

- Create references
  - Nicknames
- Create local elements based on remote elements
  - Views

**Generate Script**

**Generate Script Options**

Select the options to use to generate the script.

Deployment platform: /RDA\_Demo/StandardizedBankingInfo-Design2.dbm

Query language

SQL

SQL/XML

Query type

Create a Select statement

Create an Insert statement

Create a View statement

Create fully qualified names

Script name: Federated Schema.sql

< Back   Next >   Finish   Cancel

# Rational Data Architect in one slide



**Data Architect**



**Data Admin**



**Developer**

**Rational.**

**Data Architect**

Reverse engineering  
 Navigation  
 Discovery  
 Visualization  
 Code generation

Logical Design  
 Naming Standards  
 Integration Design  
 Physical Design  
 Rules and Model Validation

Lifecycle management  
 Compare and Sync  
 Impact Analysis  
 Reporting  
 Team Integration





Thank You

For more information, visit us at [www.ibm.com](http://www.ibm.com)

IBM Software Group



*Move Faster*

© 2005 IBM Corporation