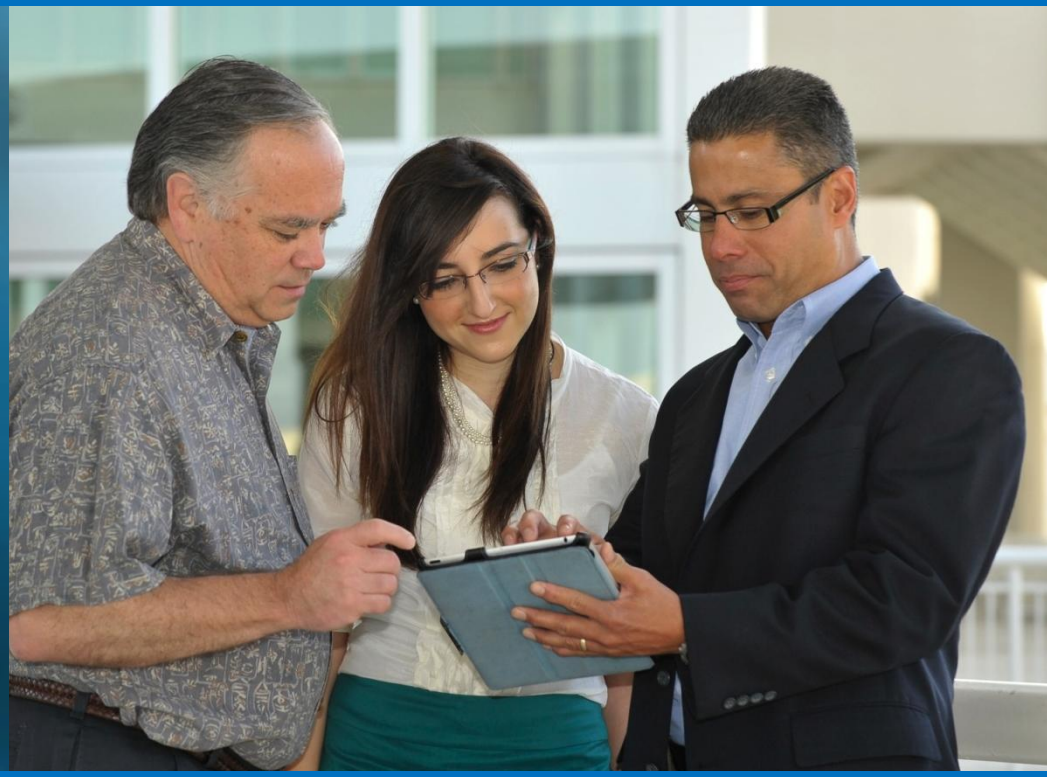
ASUG SAP BusinessObjects
USER CONFERENCE

September 9–11, 2013
Anaheim, California

Business Intelligence
for a PASSIONATE
COMMUNITY



Universe Building for Mere Mortals

Alan Mayer – Solid Ground Technologies
Session 0610

Agenda

- **Introduction**
- Getting started
- Making a connection
- Building the foundation
- Resolving inconsistencies
- Creating classes and objects
- Releasing the final version
- Conclusion

Dedicated to …

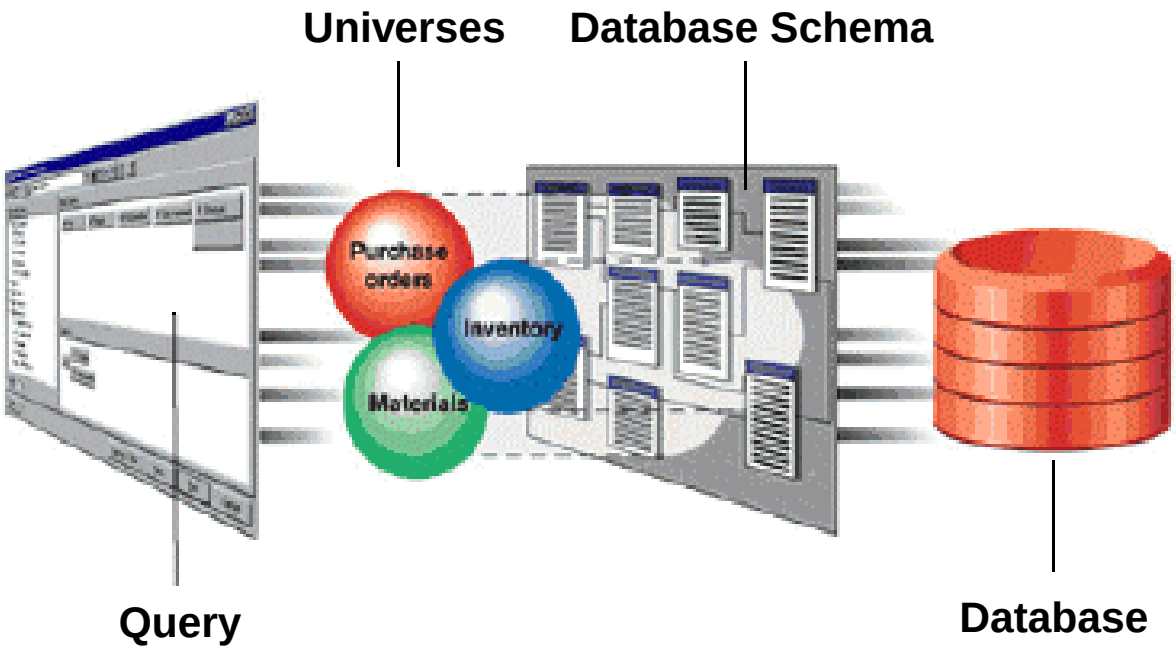
Introduction

Alan Mayer

- Co-founded Integra Solutions in 1993
  - Used BusinessObjects since 1992 (Version 2.2)
  - Wrote the first BusinessObjects training manuals
  - Over 75 Fortune 1000 customers before company was sold in 2007
- Presented at every national conference
- Founded Solid Ground Technologies in 2009
  - Different company – same principles
  - Specialize in BusinessObjects consulting and training

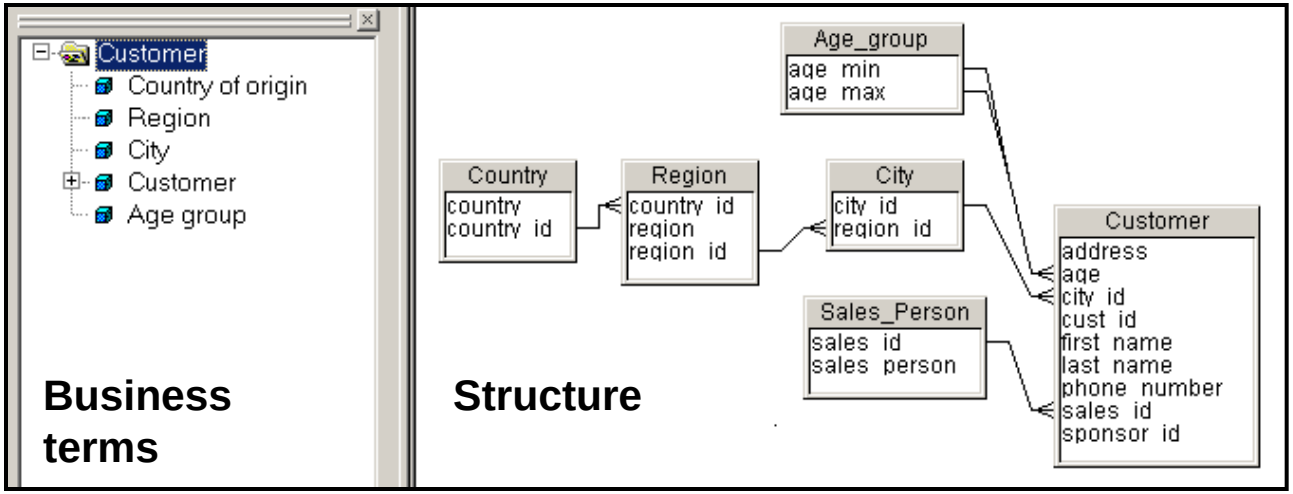
A Universe?

- Semantic layer that is created between data and the user
  - Expressed in business terms that users understand
  - Tables and joins are predefined



A Simple Definition

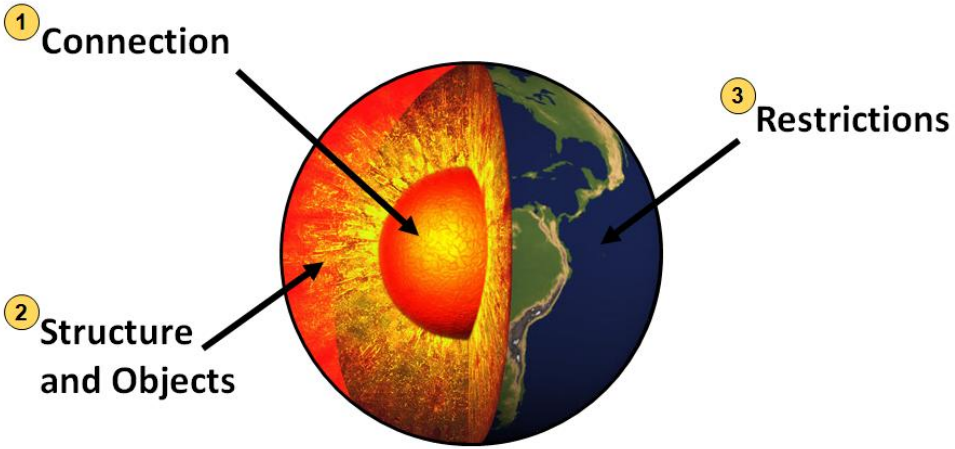
- Universes contain
  - A connection to the data
  - A structural representation of that data
  - Business terms based on that structure



Two Types of Universes

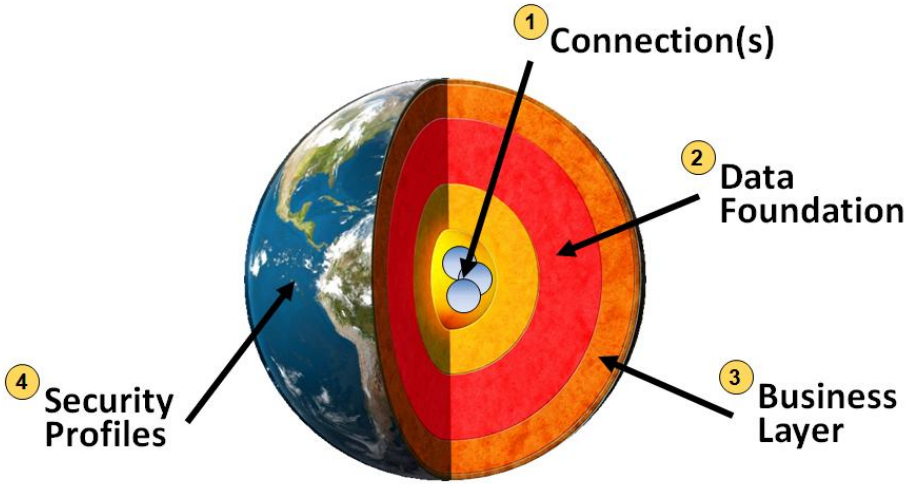
- Two types can be built based on version
  - .UNV
    - Legacy universes created in any current version (XI 3.1, BI 4.x)
    - To make things simpler, we'll restrict .UNV to just XI 3.1
  - .UNX
    - New for BI 4.x installations
- Which should you build?
  - Depends on your environment
  - Certain new features only available in .UNX
    - Multiple connections
    - More data sources

Another Way of Looking at it …



.unv

= .unv



.cnx or .cns

\+ .dfx

\+ .blx

= .unx

Our Mission

- Show how to build universes regardless of version
- Many basic concepts are the same
- Version-specific features will be pointed out
  - Look for these symbols

3.1 .unv

4.x .unx

- We'll develop **BOTH** types of universes in this presentation!

Our Instruction Manual

- Create a project 4.x
- Add a data connection
- Define the structure by inserting tables and joins
- Resolve logical inconsistencies
- Create classes and objects
- Publish / export the final result

Beyond Our Scope

- Showing how universes are developed in 3.1 and 4.x is ambitious
  - Especially in less than an hour
- Not much time for these topics:
  - Detailed connection and parameter selections
  - Performance tuning
  - Federation
  - Complex object and join creation
  - Hierarchies / Navigation paths
  - Aggregate navigation
  - Security rules

Agenda

- Introduction
- **Getting started**
- Making a connection
- Building the foundation
- Resolving inconsistencies
- Creating classes and objects
- Releasing the final version
- Conclusion

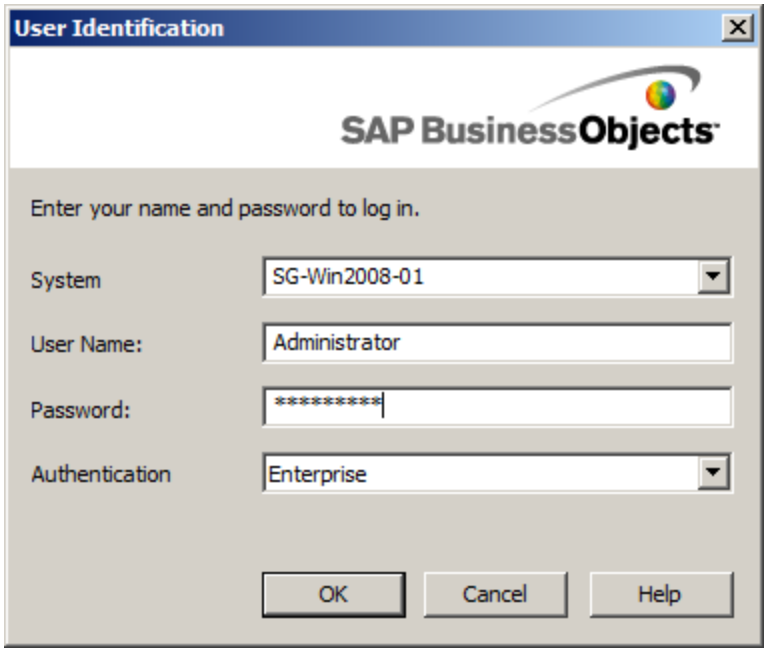
Use the Right Tool

- .UNV legacy universes can be created in either version
  - Universe Design tool in BI 4.x
  - Designer tool in XI 3.1
  - Very little difference between these two tools
- Use the Information Design Tool (IDT) for .UNX
- Best way to proceed:
  - Decide on which version (XI 3.1, BI 4.0)
  - Decide on which universe type to create (.UNV, .UNX)
  - Follow the slides for your choices

Logging in

3.1

- Must log into the Universe Designer as the first step
  - No login necessary for IDT (.UNX)

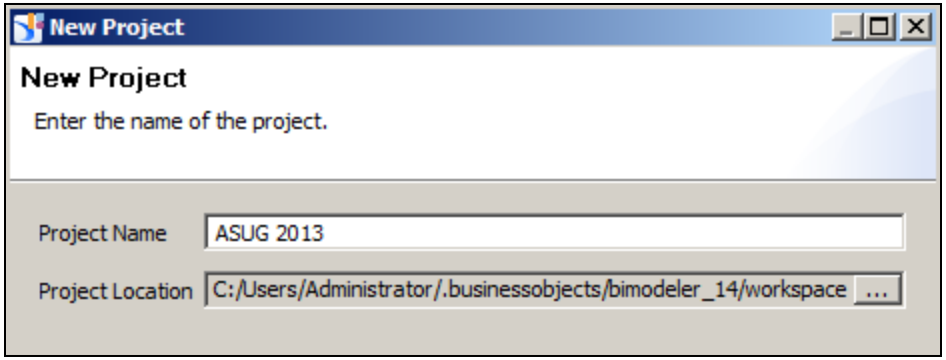


It is possible to bypass the login by setting Authentication to Standalone. You must have logged in at least once prior to trying.

Creating a Project

4.x

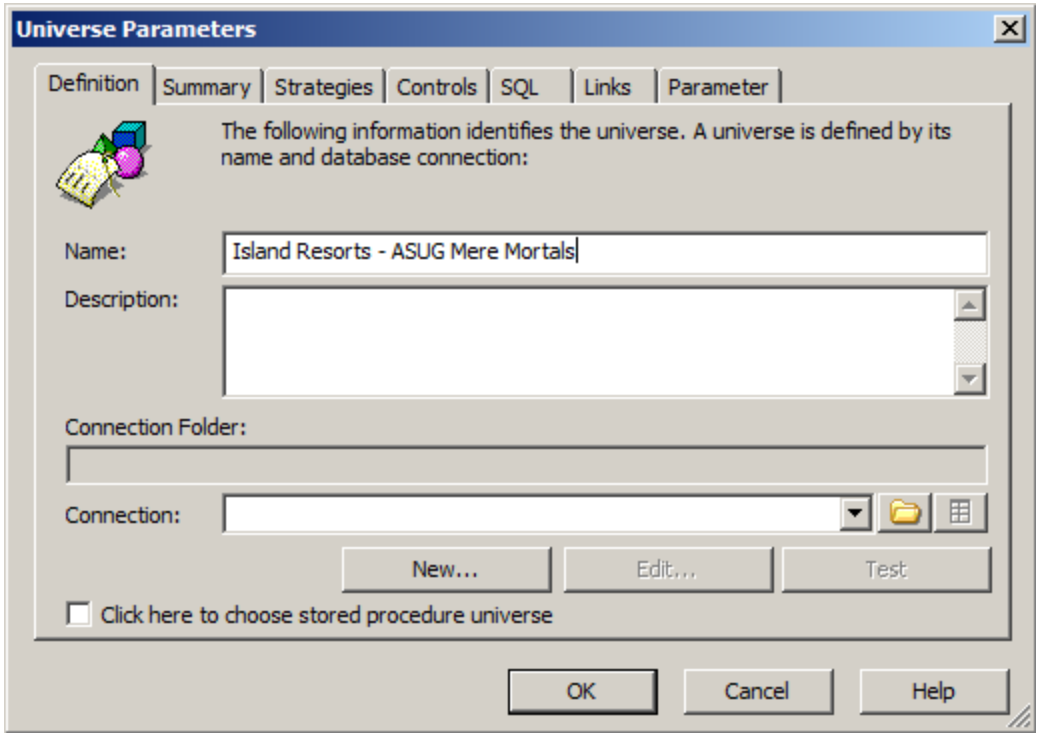
- Developers must create a project to get started
  - **File > New > Project**
- Projects contain:
  - Connections
  - Data Foundation layer (structure)
  - Business layer (business terms)



Creating a Universe

3.1

- Developers can create a new universe to get started
  - **File > New**



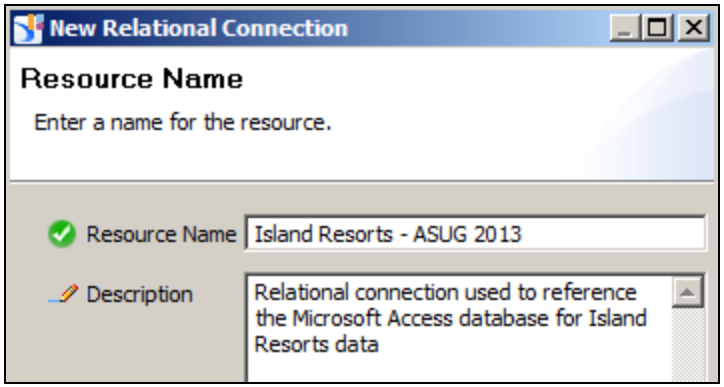
Agenda

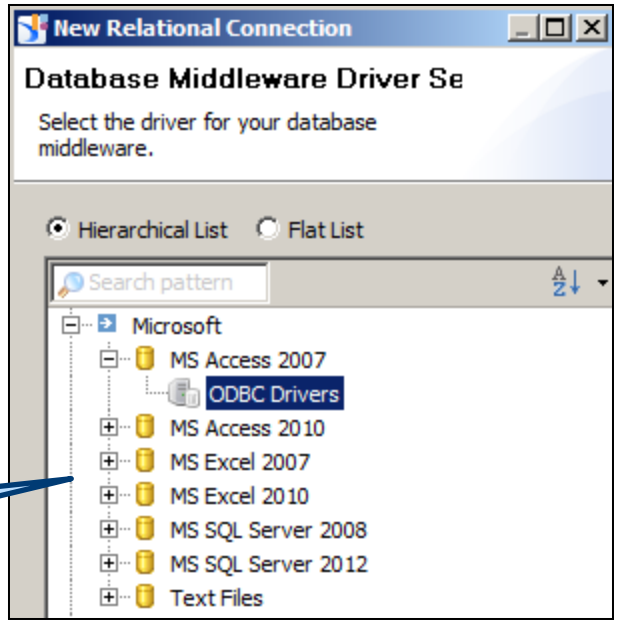
- Introduction
- Getting started
- **Making a connection**
- Building the foundation
- Resolving inconsistencies
- Creating classes and objects
- Releasing the final version
- Conclusion

Creating Project Connections

4.x

- While in a project, create a new connection
  - File > New > **Relational Connection** or **OLAP Connection**
  - **Relational Connection** chosen below





Connections to Excel and text files also available here.

Creating Project Connections, cont'd

4.x

- Additional connection choices

Relational

- Apache
- Generic
- GreenPlum
- Hewlett Packard
- HSQLDB
- IBM
- Ingres
- Microsoft
- Netezza
- Oracle
- PostgreSQL
- Salesforce.com
- SAP
- Sybase
- Teradata Corporation

- SAP
  - MaxDB 7.7
  - mySAP ERP 2004
  - SAP ERP 6
  - SAP HANA database 1.0
    - JDBC Drivers
    - ODBC Drivers

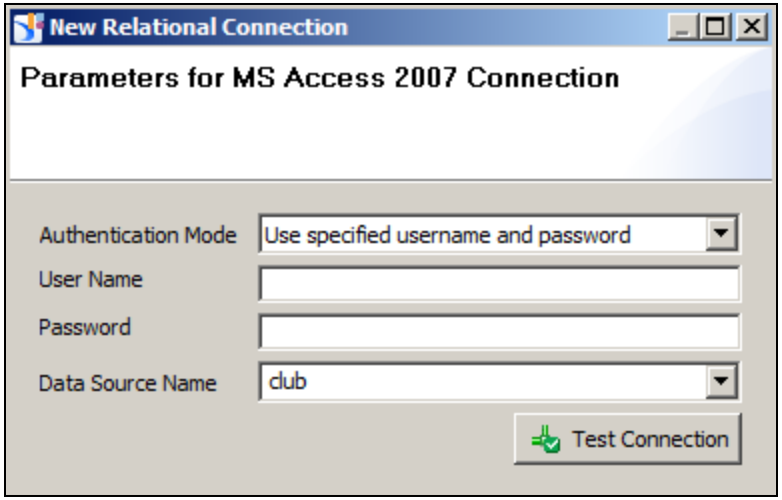
OLAP

- Microsoft
  - Microsoft Analysis Services 2005
    - XMLA
  - Microsoft Analysis Services 2008
    - XMLA
  - Microsoft Analysis Services 2012
    - XMLA
- Oracle
  - Hyperion Essbase 11.x
    - Essbase API for Analysis
- SAP
  - SAP Netweaver BI 7.x
    - SAP BICS Client
- SAP BusinessObjects
  - Financial Consolidation 10 for Microsoft Platform
    - XMLA
  - Planning and Consolidation 10 for Microsoft Platform
    - XMLA
  - Profitability and Cost Management 10
    - XMLA

Creating Project Connections, cont'd

4.x

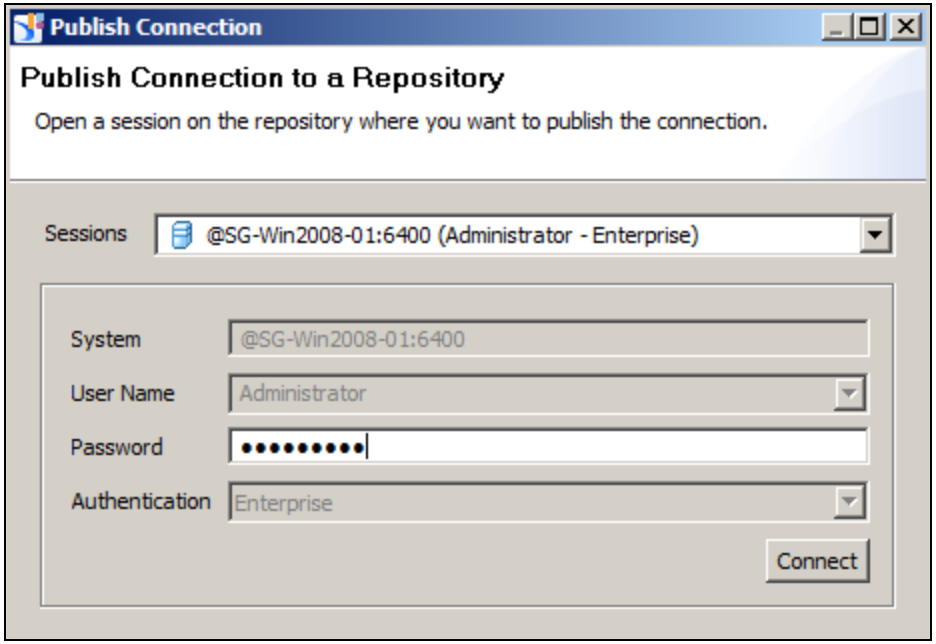
- Add login information to reach that data source
  - Relational example
  - User / password optional for MS Access, Excel, flat files
  - Additional details go beyond the scope of this talk



Publishing the Connection

4.x

- The initial connection is “local” (.cnx)
  - Cannot be access by anyone but yourself
  - Must be published for Webi-based universes
  - **Right click** on connection > **Publish Connection to Repository**



What if a connection isn’t published?

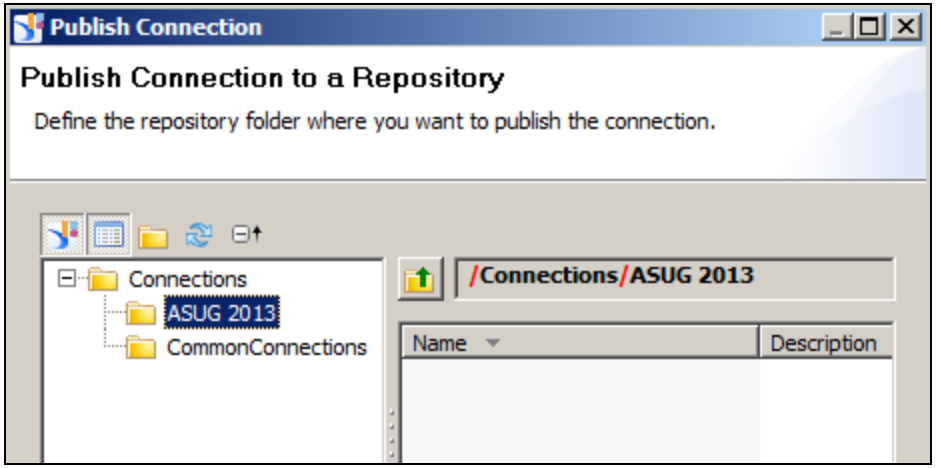
Data foundations could still use a local connection, **BUT** …

Universes could not be published based on that connection

Publishing the Connection, cont'd

4.x

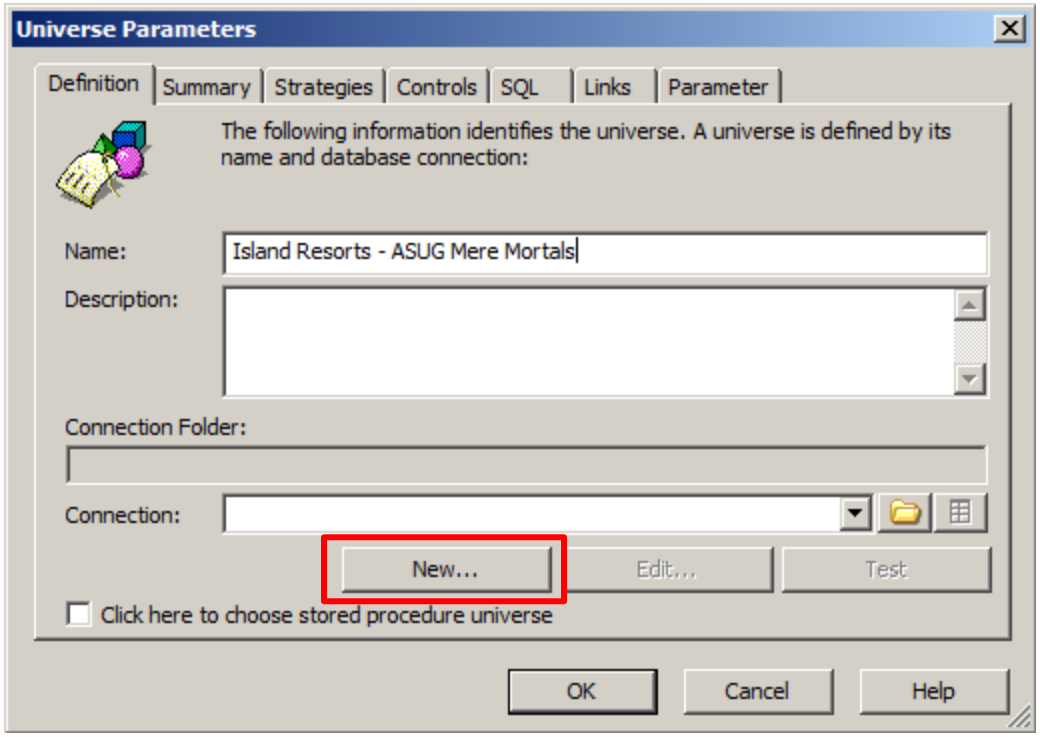
- The published connection can be stored in a folder
  - Select a folder and click Finish
  - Shortcut for the published connection is created (.cns)
  - This shortcut can be used in Data Foundations



Creating a Single .UNV Connection

3.1

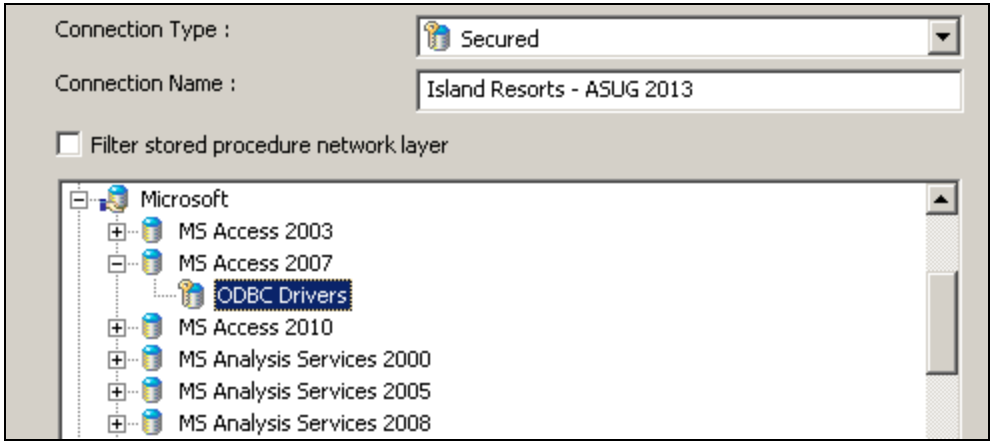
- Many connections can be created using the Universe Designer
- Only one of these may be used per universe
  - Options may vary based on version (3.1 vs. 4.x)



Creating a Single .UNV Connection, cont'd

3.1

- The connection should be secured for Enterprise use
  - Meaning … other people have access to the connection
- A few differences from BI 4.x connections
  - No connection folder
  - Can secure at creation time
  - Some of the data sources may not be available



Connections must be **secured** before publishing the universe. This allows other Enterprise users to use it

Demonstration

3.1 4.x

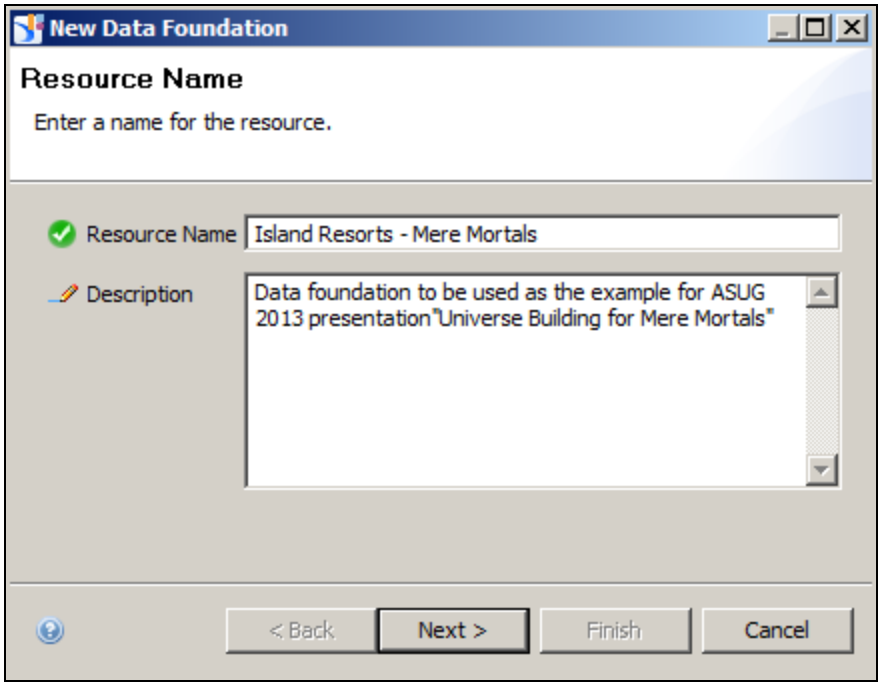
Agenda

- Introduction
- Getting started
- Making a connection
- **Building the foundation**
- Resolving inconsistencies
- Creating classes and objects
- Releasing the final version
- Conclusion

Creating a Data Foundation

4.x

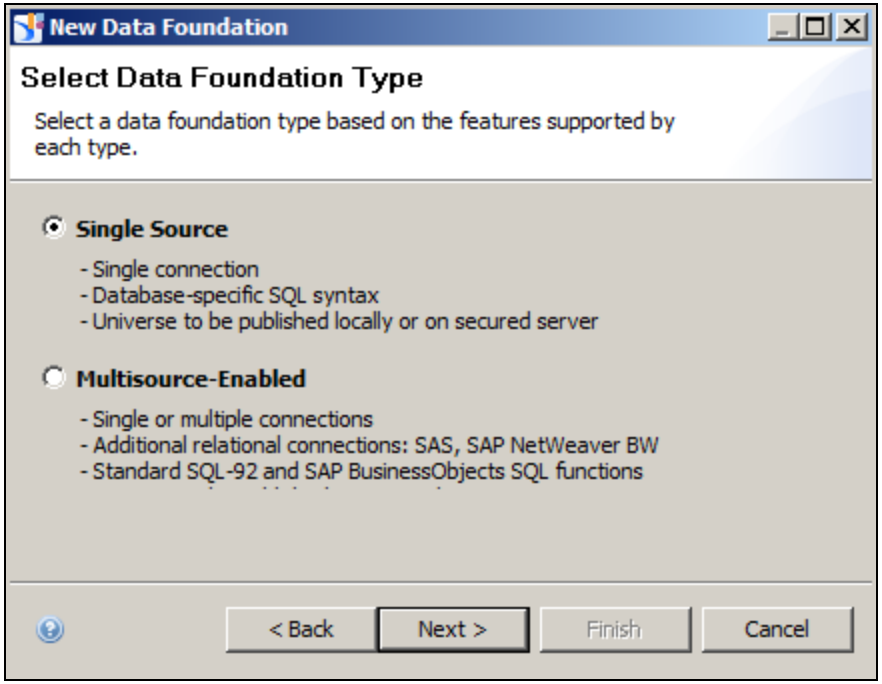
- Once a connection is created, structure can be defined
- In IDT, this is done by creating a Data Foundation layer (.dfx)
  - **File** > **New** > **Data foundation**



Creating a Data Foundation, cont'd

4.x

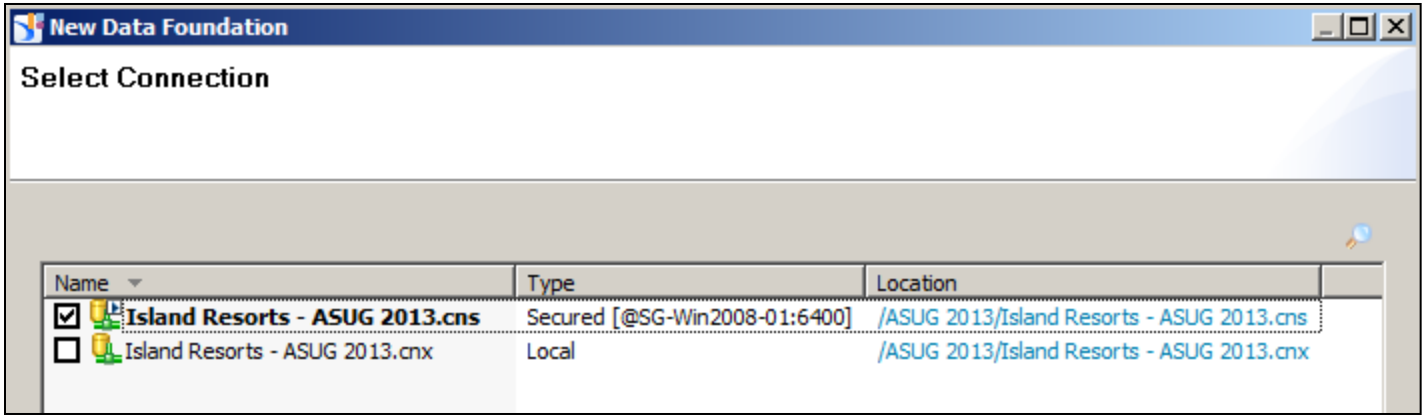
- Choose between single or multi-source
  - Some data sources require the multi-source option
  - This will involve federation techniques – beyond our scope



Creating a Data Foundation, cont'd

4.x

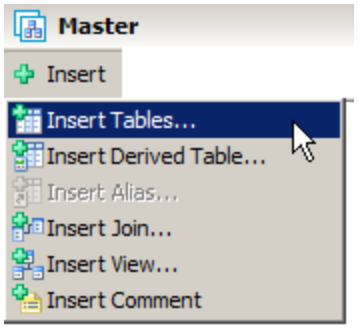
- Choose the secured connection

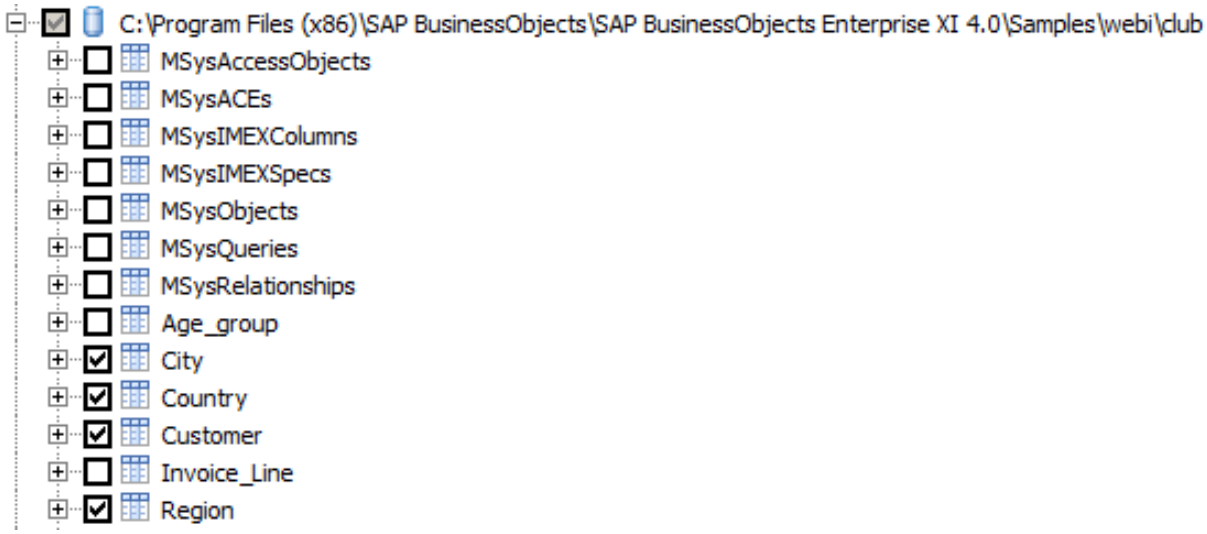


Adding Tables

4.x

- Select the **Insert** menu drop-down
- Select **Insert Tables …**
- Select the **club** datasource and choose one or more tables

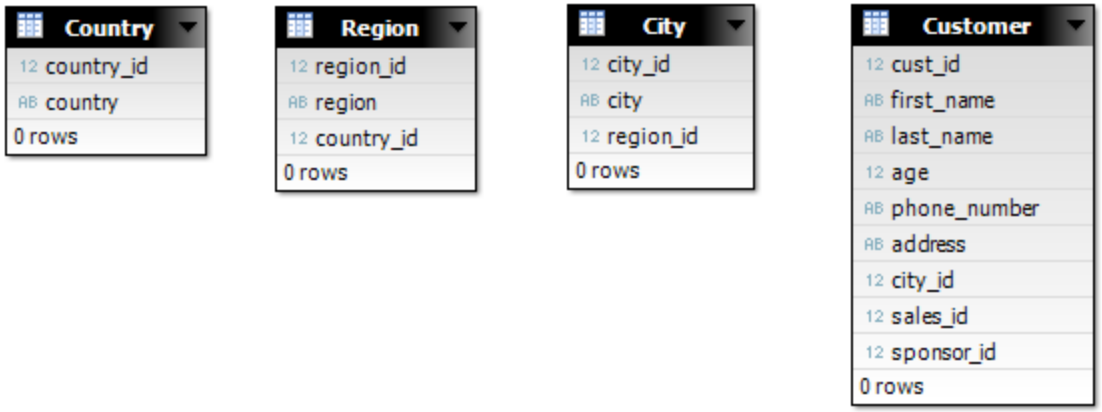




Adding Tables, cont'd

4.x

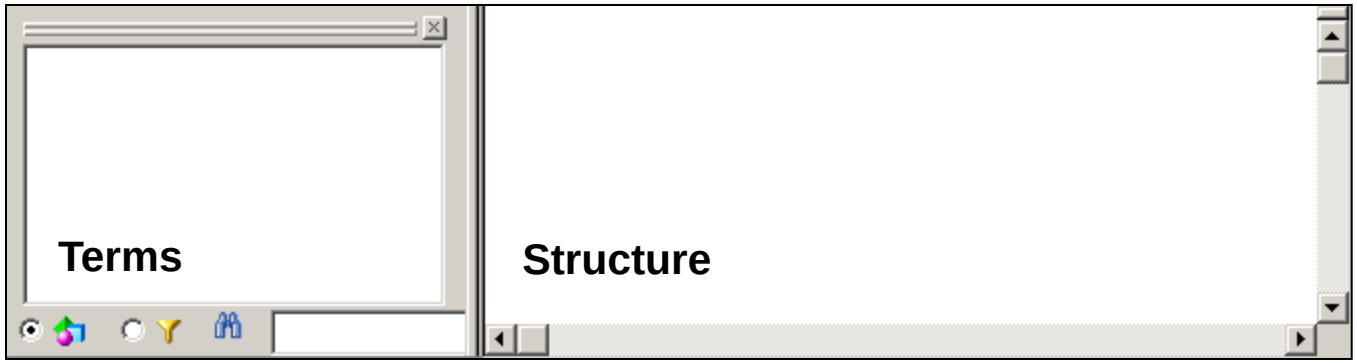
- Arrange the tables in the order to be joined



Adding Structure

3.1

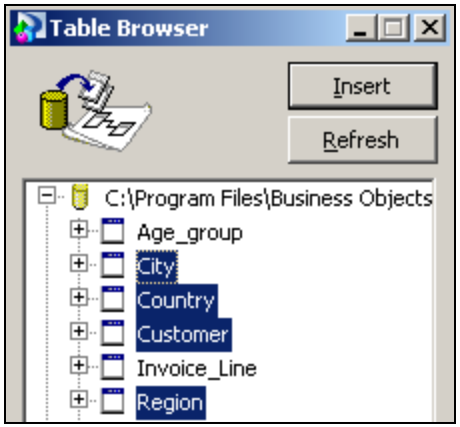
- No concept of a data foundation
- The structure is part of the universe once created
  - Initial Structure Pane window will be blank



Adding Tables

3.1

- Use the Table Browser 
- Open the **club** data source
- Select the tables needed then click **Insert**
- Arrange the tables in the order to be connected



Demonstration

3.1 4.x

Joins

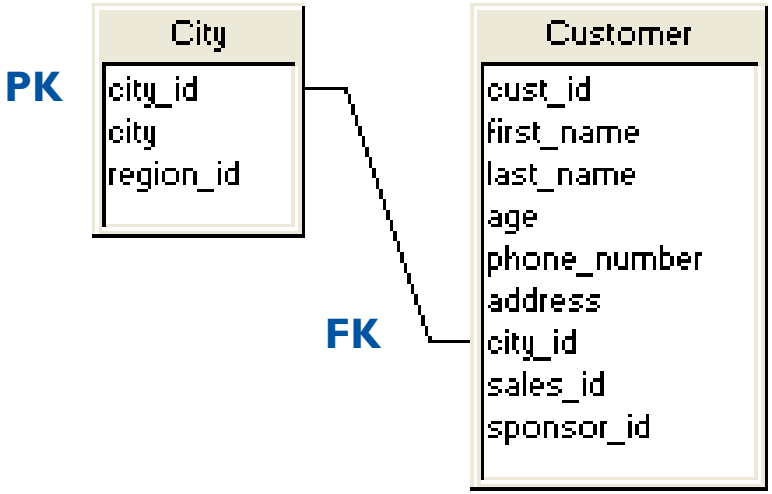
3.1 4.x

- Relationships between tables can now be defined
- Known as joins, these relationships can take many forms
  - Inner join
  - Outer join
  - Theta join
  - Recursive join
  - Self-restricting join
  - Shortcut join
- The next few slides will explain each join type

Inner Joins

3.1 4.x

- Also known as equi-joins or normal joins
- Usually take the the following form
  - Single join: Primary Key (PK) = Foreign Key (FK)
  - Compound Join: $PK_1 = FK_1$ and $PK_2 = FK_2$ and …

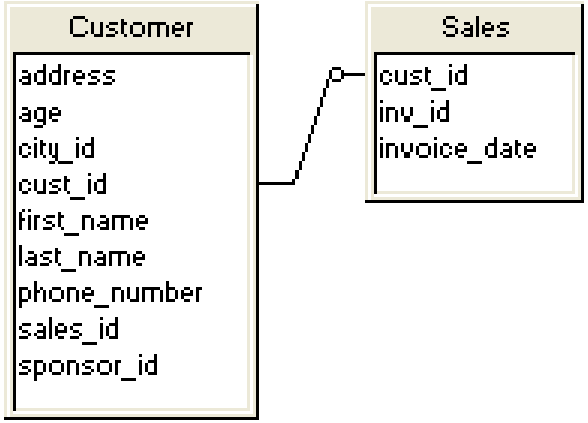


City.city_id=Customer.city_id

Outer Joins

3.1 4.x

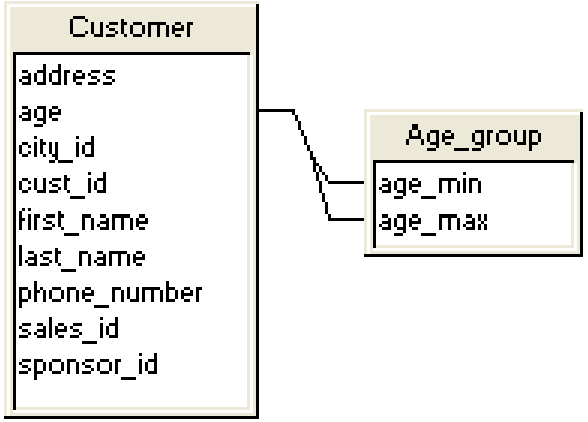
- Forces all rows from one table to be considered even if no matching row exists in second table
  - For example: “Return all customers and orders if they exist”
  - Syntax varies based on database
  - Outer joins CASCADE!



Theta Joins

3.1 4.x

- Relates two tables using relationships other than equality

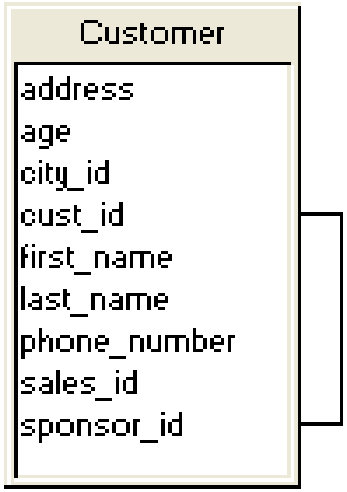


Customer.age
BETWEEN
Age_group.age_min and
Age_group.age_max

Recursive Joins

3.1 4.x

- A row is related to other row(s) within the same table
  - Example: A sponsor may be stored in the same table as their referrals

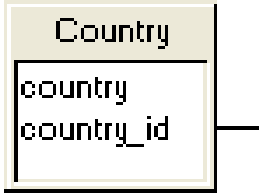


Customer.sponsor_id = Customer.cust_id

Self-Restricting Joins

3.1 4.x

- A condition that should ALWAYS be applied against a table
  - A universal condition rather than a join
  - One way to force BusinessObjects to always add the condition to any SQL statement that references that table

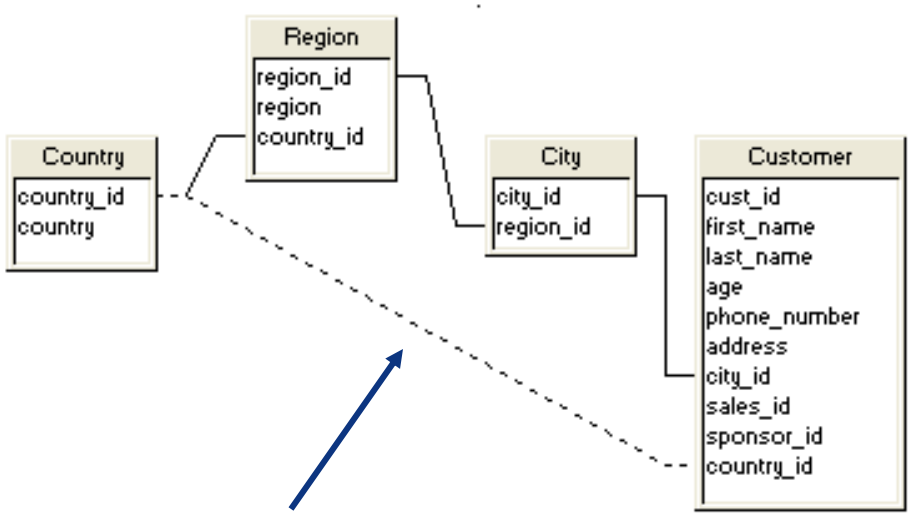


Country.country_id = 1

Shortcut Joins

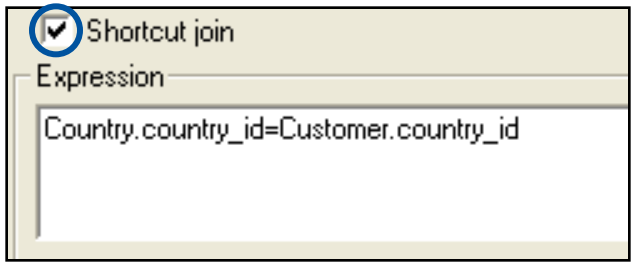
3.1 4.x

- Provides a shortcut or alternative path between tables
  - Example: The Customer table may contain an extra column that allows a direct join to Country



Shortcut Join

Join Editor

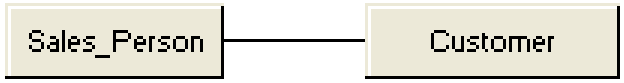


Join Cardinality

3.1 4.x

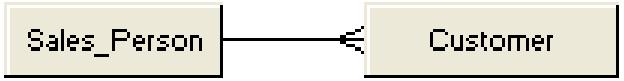
- Join cardinalities MUST be defined
  - Cardinality determines the number of rows related to a current row
  - They help resolve logical problems later

1:1



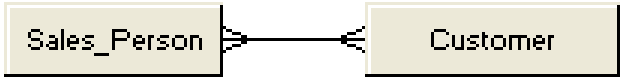
A salesperson has 1 customer;
A customer has 1 salesperson

1:Many



A salesperson has 1 or more customers;
A customer has one salesperson

Many: Many



A salesperson has 1 or more customers;
A customer has 1 or more salespersons

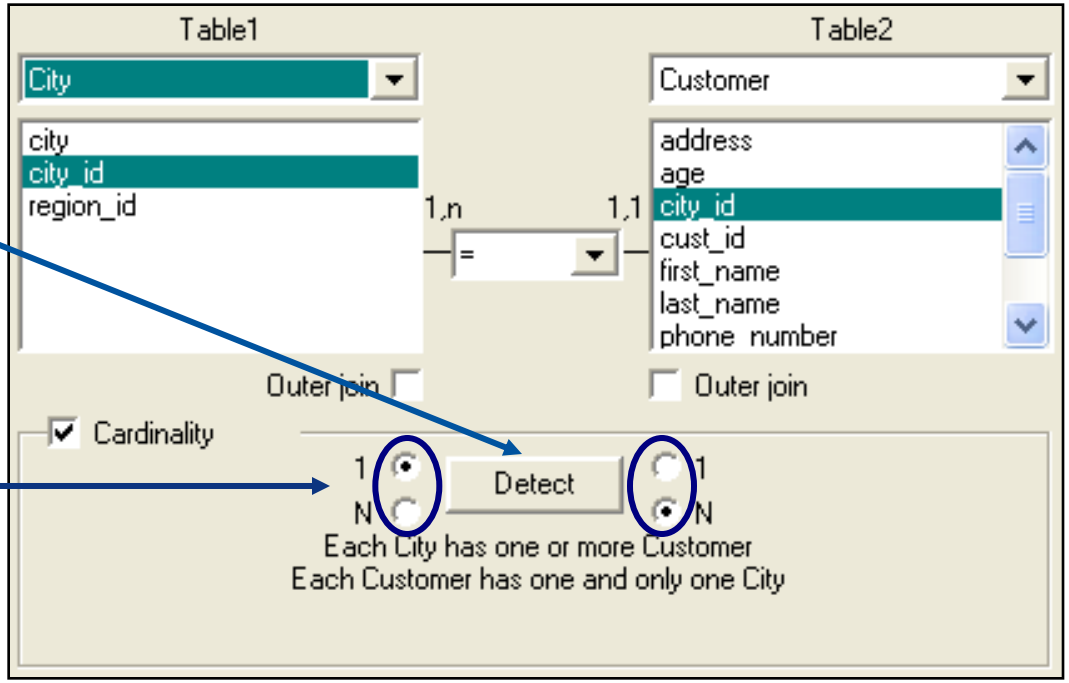
Setting Cardinalities

3.1 4.x

- Cardinalities can be established two different ways
  - Automatic Detection (not as good)
  - Manually via Join Editor (better)

Automatic

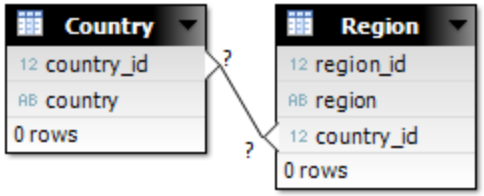
Manual



Adding Joins

3.1 4.x

- Several methods
  - Trace the join from one table to another
    - Click and drag from one column to another



  - Use the Join Editor
    - **Insert** Menu > **Insert Join**
  - Detect joins
    - 3.1 **Tools** > **Automated Detection** > **Detect Joins**
    - 4.x From Data Foundation: **Detect** > **Detect Joins**

Detecting joins is not a preferred strategy. Additional joins may be added that are technically possible but not realistic

Demonstration

3.1 4.x

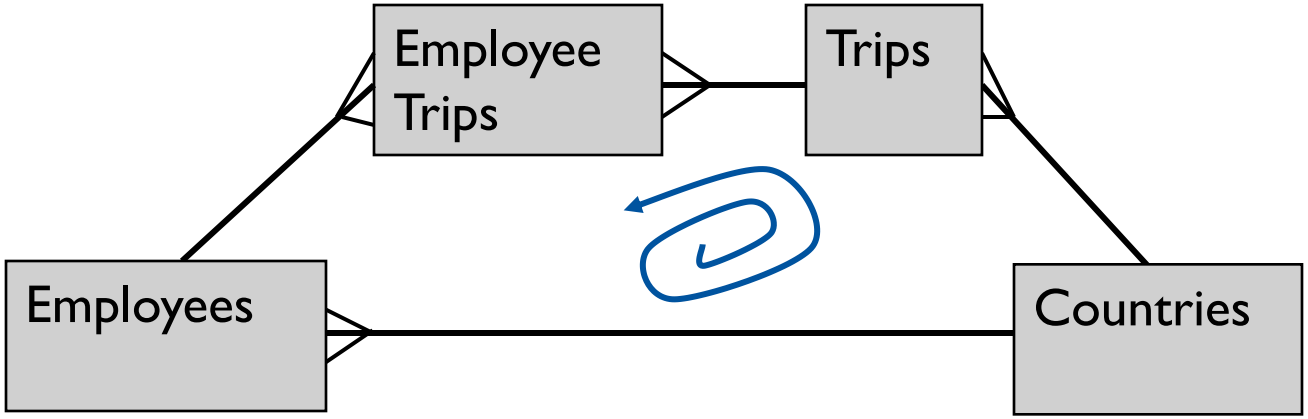
Agenda

- Introduction
- Getting started
- Making a connection
- Building the foundation
- **Resolving inconsistencies**
- Creating classes and objects
- Releasing the final version
- Conclusion

Loops

3.1 4.x

- A loop is created when two or more paths exist between tables
  - An employee can take a business trip to a country
  - An employee is born in a country

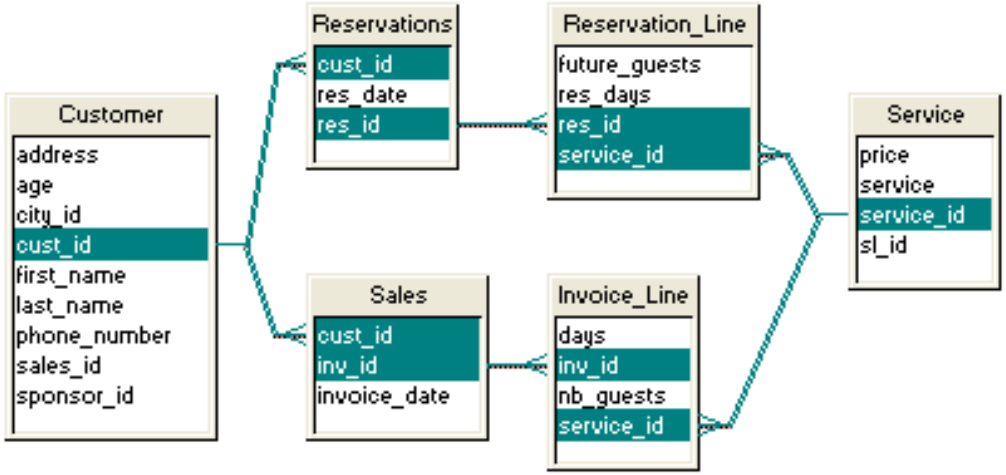


Another definition: Loops represent "pools of water" that cannot escape

Loops, cont'd

3.1 4.x

- Detecting Loops
  - 3.1 Tools > Automated Detection > Detect Loops 
  - 4.x Aliases and Contexts > Visualize Loops 



Why are loops bad?

SQL cannot be created because there is more than one path between tables

Loops, cont'd

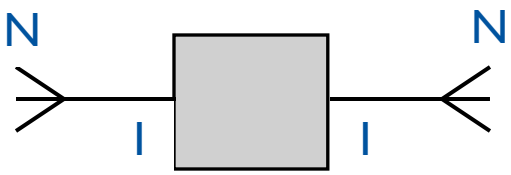
3.1 4.x

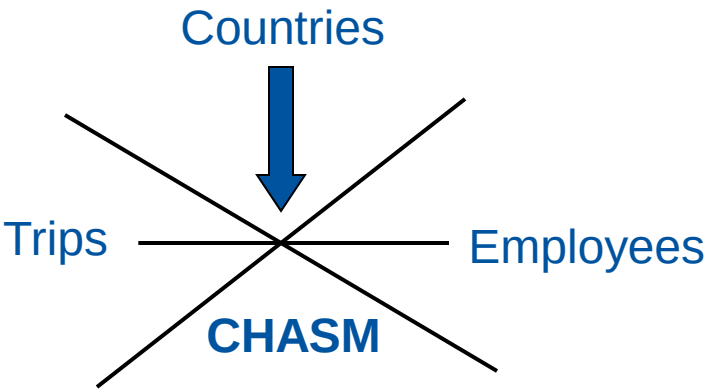
- Caution when using Detect tools ….
  - Join cardinalities must be set!
    - Else Detection may offer the wrong advice
  - Always review the solutions offered

Chasm Traps

3.1 4.x

- Look for logical traps
  - The chasm trap is a common one
  - Usually the result of a many to one, one to many relationship
- Chasms cannot be crossed
  - Took a trip to England …
  - … means you were born there?



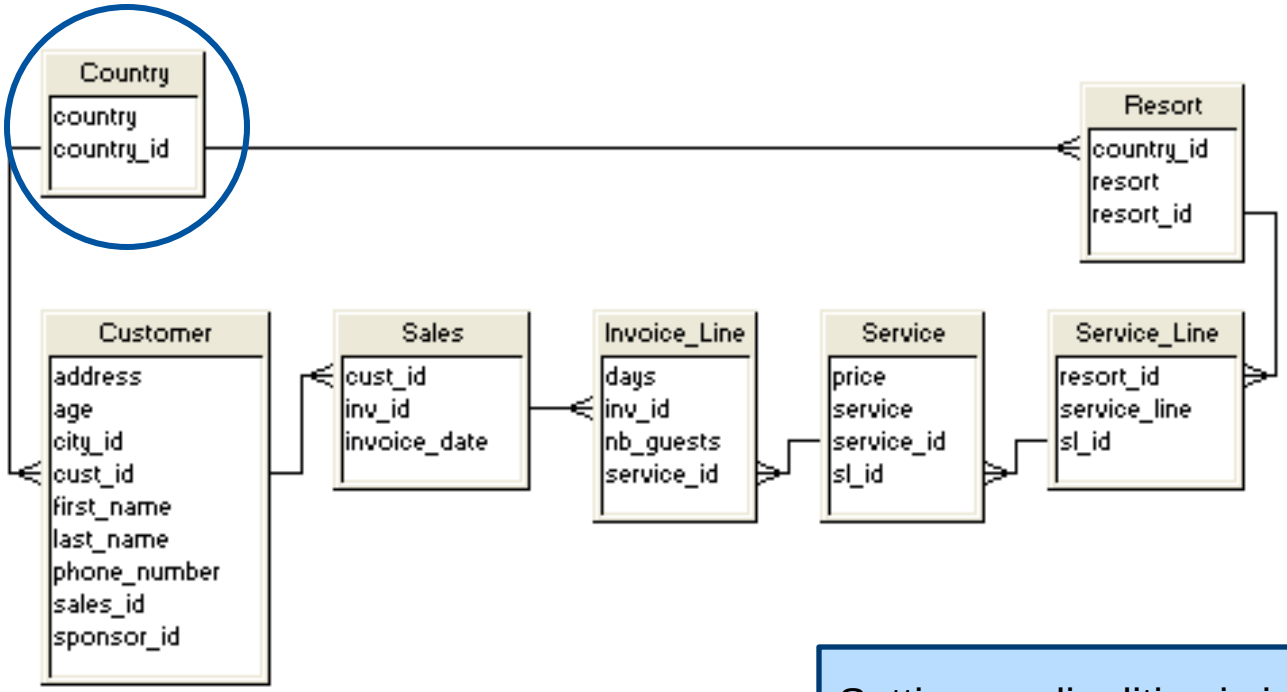


Chasms are often created when joining to lookup tables.

Chasm Traps, cont'd

3.1 4.x

- Identfying chasms
  - In the following structure, Country is a chasm trap



Setting cardinalities is important!
It helps identify traps like this one

Aliases

3.1 4.x

- Aliases can resolve chasm traps
  - Known as table aliases when writing SQL statements
  - Used by BusinessObjects to logically separate the trap into pieces

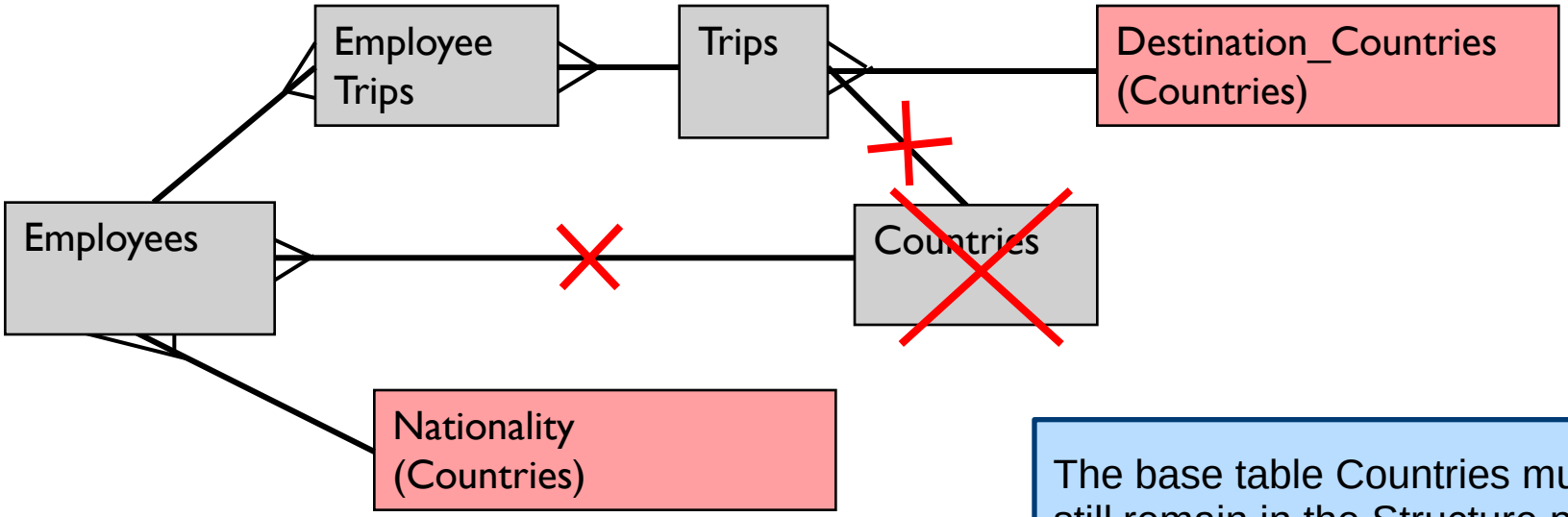
```
SELECT    a.country,
          b.country
FROM      country a,
          country b
WHERE …
```

Table aliases

Aliases, cont'd

3.1 4.x

- Countries would be replaced by one (or two) aliases
  - Create an alias for each path
  - One alias is sufficient
  - Two aliases makes the diagram more readable



The base table Countries must still remain in the Structure pane!

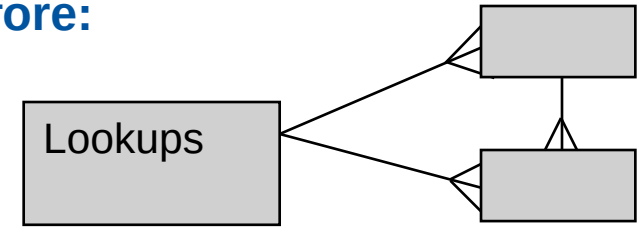
Aliases

- Generic lookup tables can be resolved using aliases

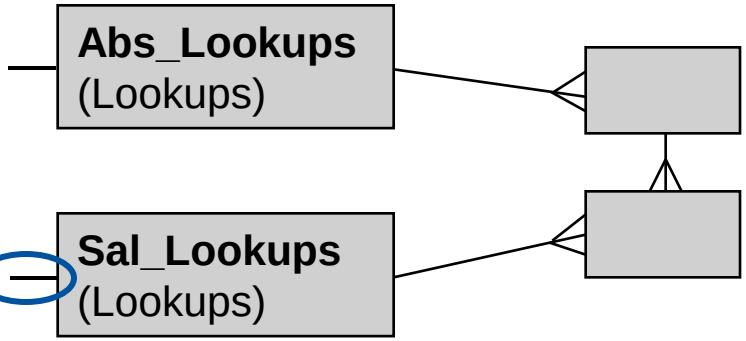
Lookups

Type	Code	Description
SAL	001	Base Salary
SAL	002	Overtime
SAL	003	Company Car
ABS	001	Holiday
ABS	002	Sick
ABS	003	Sick of Job

Before:



After:



Self-Restricting Join
Sal_Lookups.type = ‘SAL’

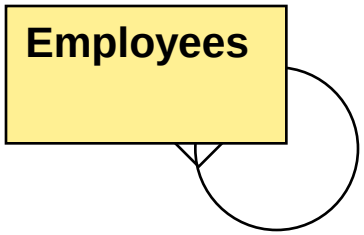
Aliases

- Recursive relationships can also be resolved
  - The depth of those relationships should be known

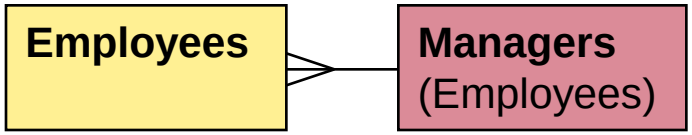
Employees

Emp_ID	Name	Manager_ID
1	Mayer	5
5	Smith	23
23	Betten	42
42	Byrd	

Before:



After:



Employees.manager_id = Managers.emp_id

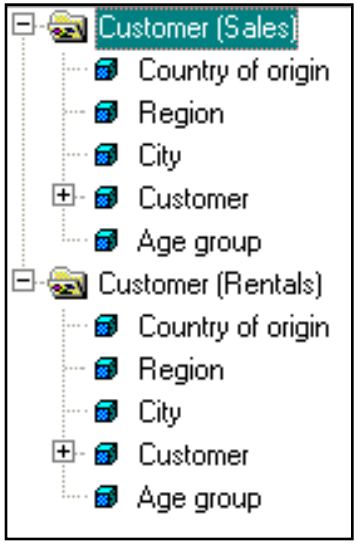
NOTE:
There are better ways of resolving recursive relationships using database techniques

Aliases, cont'd

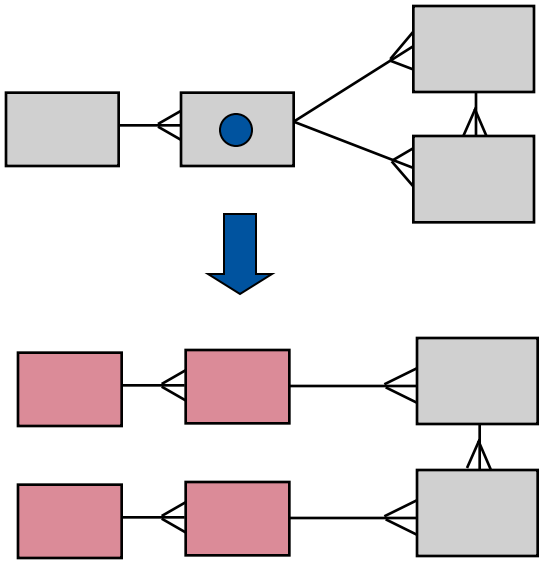
3.1 4.x

- Every loop can be resolved with aliases
  - There are drawbacks to using aliases
  - More business terms (objects) will be added
  - Those additional terms may confuse some users
  - Aliases also **CASCADE**

Problem #1



Problem #2



Aliases

3.1 4.x

- Adding aliases
  - 3.1 **Insert** menu > **Alias** 
  - 4.x Right-click on a table and choose **Alias** or **Insert Alias**
- Aliases can also be detected
  - **Tools** menu > **Automated Detection** > **Detect Aliases…** 
  - Data Foundation > **Aliases and Contexts** > **Detect Contexts…** 
  - Looks for possible chasm traps for you
  - May not be a good idea based on previous drawbacks

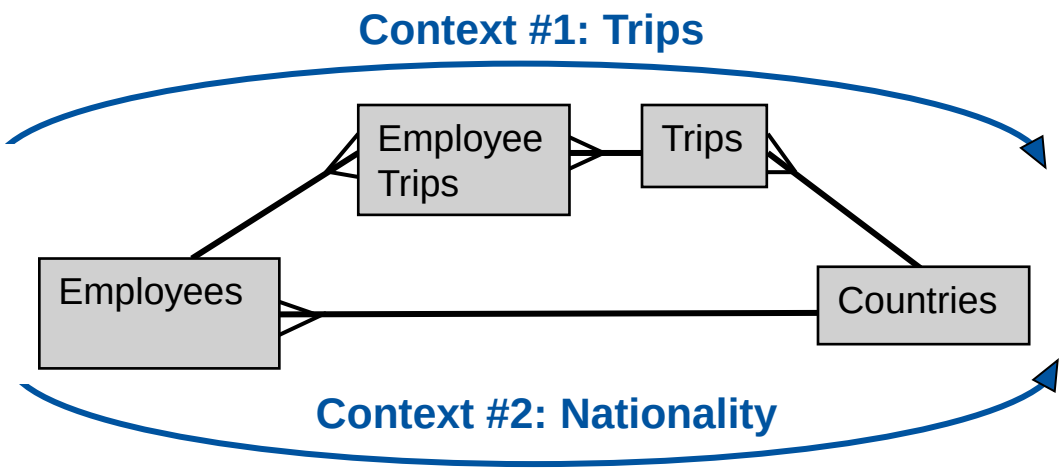
Demonstration

3.1 4.x

Contexts

3.1 4.x

- Contexts can also resolve loops
- A context represents one path or set of joins between tables



Trips	Employees.emp_id = Employee_Trips.emp_id
	Employee_Trips.trip_id = Trips.trip_id
	Trips.country_id = Countries.country_id
Nationality	Employees.country_id = Countries.country_id

Contexts, cont'd

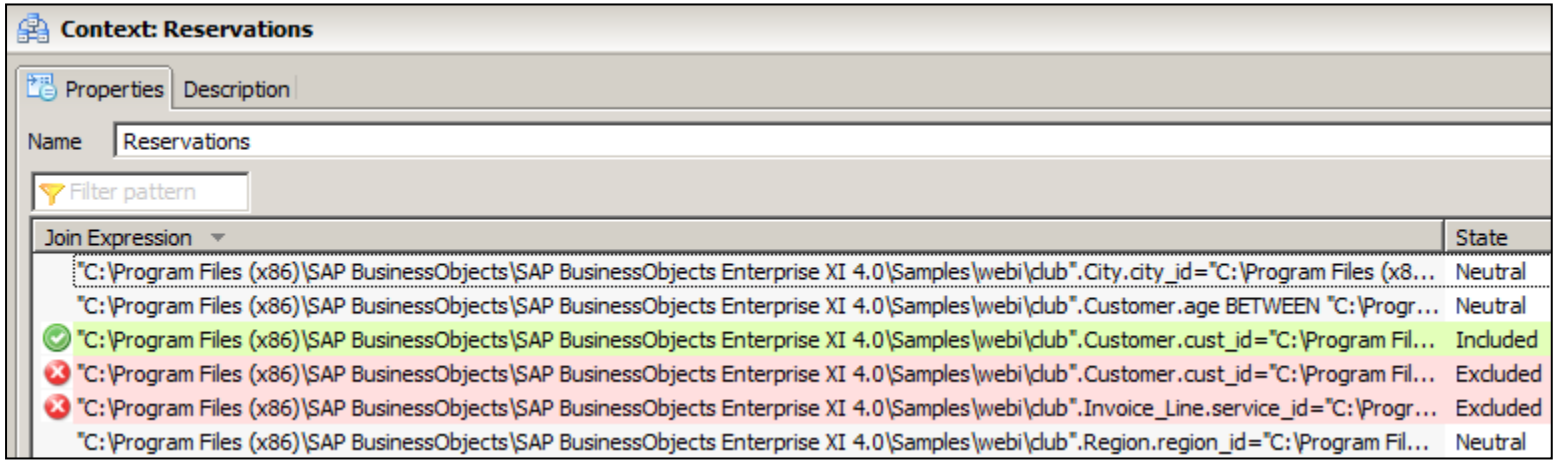
3.1 4.x

- Contexts resolve the loop at runtime rather than in the Designer
  - This means that a context-based solution still has loops!
- The user may be asked to choose between the contexts
  - BusinessObjects will try to infer which context to use
  - If it can't figure it out, the user usually chooses a context
- Once a context is chosen, all other joins "disappear" 3.1
  - Only joins listed in the context will be used to build the final SQL program
- Using contexts does not force additional objects to be created

Contexts, cont'd

4.x

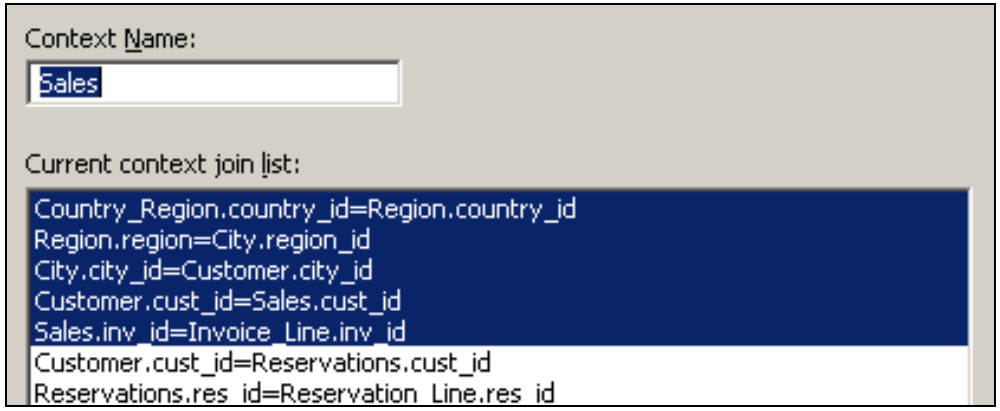
- Adding contexts
  - Data Foundation > **Aliases and Contexts** > **Add Context** 
  - Joins can be Included, Excluded, or Neutral
  - **Include** joins from one side of the loop
  - **Exclude** joins from the other side
  - All others will remain neutral (added by default)



Contexts, cont'd

3.1

- Adding contexts
  - **Insert** menu > **Context…** 
  - Name the context and add a description
  - Choose the joins that will belong
  - **WARNING!**
    - All joins must be added that make business sense



New joins that are added after the context is created must be added to at least one context ….

… else it will never be used!

Contexts, cont'd

3.1 4.x

- Contexts can be detected
  - 3.1 **Tools** menu > **Automated Detection** > **Detect Contexts** 
  - 4.x Data Foundation > **Aliases and Contexts** > **Detect Contexts** 
  - Use these options carefully
  - Don't accept the proposed contexts blindly
  - Use them as an "assist" to create your own contexts

Demonstration

3.1 4.x

Agenda

- Introduction
- Getting started
- Making a connection
- Building the foundation
- Resolving inconsistencies
- **Creating classes and objects**
- Releasing the final version
- Conclusion

Starting the User Interface

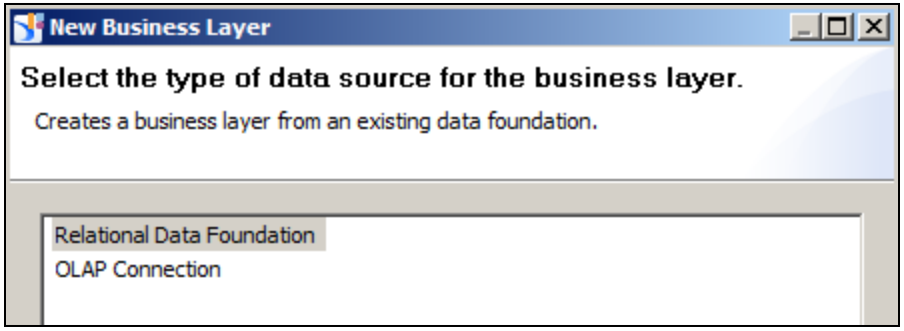
3.1 4.x

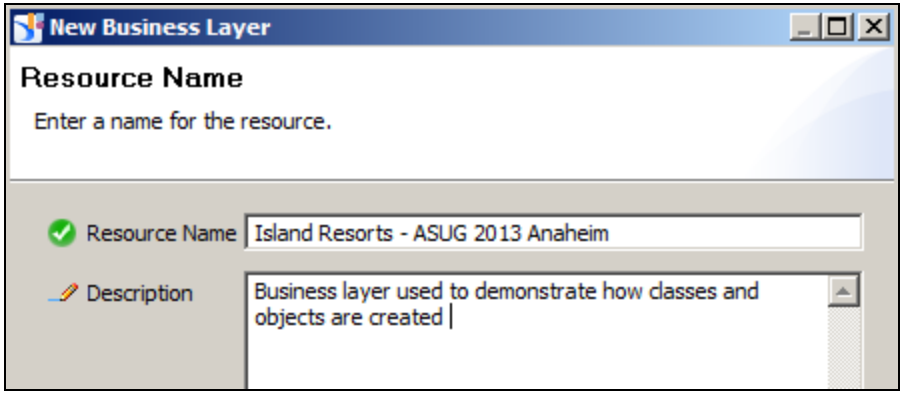
- Classes and objects can now be created
  - Objects reveal portions of the database schema to your users
    - Act as “business terms” used to build queries
    - Automatically created for multi-dimensional data sources
  - Classes organizes those business terms
    - Known as folders in BI 4.x
    - Should make sense to the ultimate users
    - Organizing objects by table rarely make sense

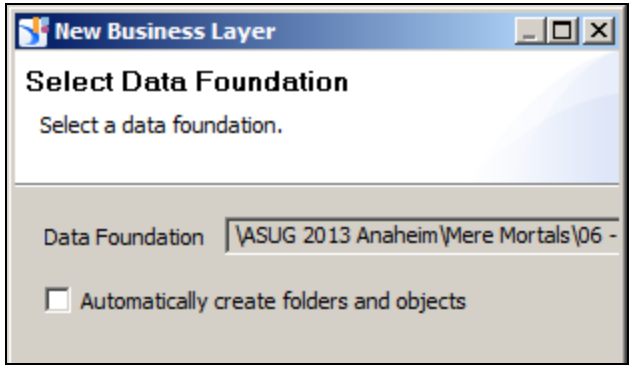
Starting the User Interface, cont'd

4.x

- In IDT, this is done by creating a Business layer (.blx)
  - **File** > **New** > **Business Layer**

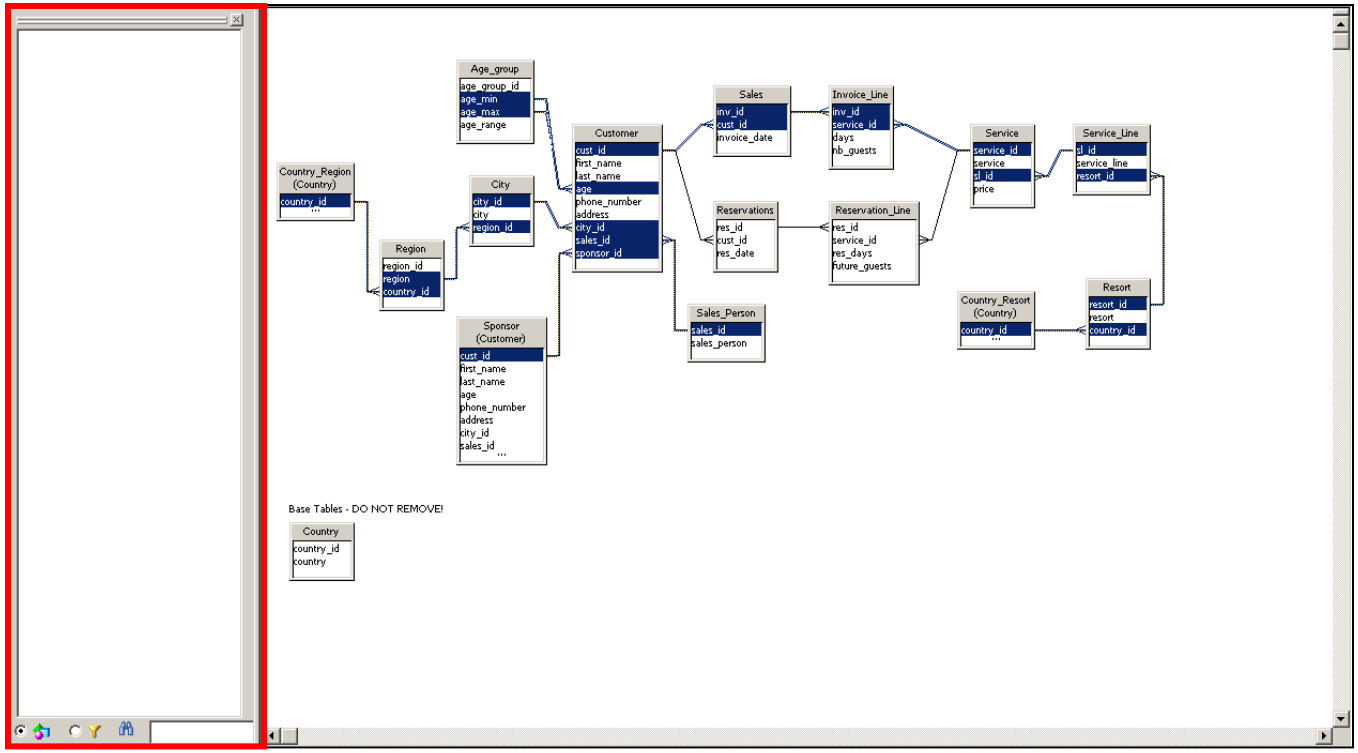






Starting the User Interface, cont'd

- In Universe Designer, use the Universe window 
  - Informally known as the Classes and Objects pane



Creating Classes / Folders

3.1 4.x

- Classes are like directories or folders for objects
- Can be nested (sub-classes are fine)
- Use any of these methods to create a class
  - 3.1 Right-click on the Universe window and choose Class
  - 3.1 **Insert** menu > **Class…** or **Subclass…** 
  - 4.x Business Layer > **Business Layer** pane > **New** > **Folder** 

Always add descriptions to all new classes. This will make the universe easier to navigate for new users.

Object Definition

3.1 4.x

- Objects are business terms used to create queries
- They are SQL expressions when building a universe
  - … except for OLAP / multi-dimensional sources
  - 50 – 75% of objects are usually just a table column
  - The remainder are calculations or expressions

```
SELECT  <SQL expression 1>,
        <SQL expression 2>
FROM …
WHERE …
```

Types of Objects

3.1 4.x

- Four types of objects that can be created
  - Dimensions
    - Base information (Example: Customer)
    - What you query by (Example: Revenue BY Customer …)
  - Details
    - Depend on a dimension (Example: Address)
  - Measures
    - Aggregated calculations (sum, count, min, max, average)
  - Conditions
    - WHERE clauses that are named

Creating Objects

3.1 4.x

- Create objects using any of these techniques
  - 3.1 Drag a table into the Universe window
    - (creates a class for table, object for each column)
  - 3.1 Drag a table column into an existing class
  - 4.x Automatically create folder and objects
    - Choice when business layer is created
    - **NOT** a good idea unless you need a quick demo universe
  - Manually create an object
    - 4.x **Business Layer** pane > **New** > **Dimension** or **Measure** or **Filter** 
    - 3.1 **Insert** menu > **Object** or **Condition**  

The Object Editor

3.1 4.x

- Create the SQL expression using the SELECT pane
- **DO NOT** add anything in the WHERE pane
  - For experienced developers
  - Use condition objects instead (just for WHERE clauses)
  - **DO ADD** descriptions for each object
    - At a minimum: Definition and example
- The editors look a little different in each version
- Major concepts are still the same
- We'll focus on the SELECT expression and List of Values

The Object Editor, cont'd

4.x

Dimension: Country

Name: Country — Active

Description: Resort's country

Data Type: String — Show Script... — Show Values...

SQL Definition | Keys | Advanced | Source Information | Custom Properties

SELECT: Country_Resort.country — SQL Assistant...

WHERE: — SQL Assistant...

Extra Tables: Country_Resort

SQL Expression

Validate

1 Country_Resort.country

Tables: Filter pattern; City, Country, Customer

Functions: Filter pattern; Operators, Database Functions, System Variables

Business Layer: Resort, Sales, Customer

List of Values: Filter pattern; List of Values

The Object Editor, cont'd

Definition | Properties | Advanced | Keys | Source Information

Name: Country

Type: Character

Description:

Select: Country_Region.country

Where:

Country_Region.country

Show object SQL

Parse

Tables and Columns:
- Age_group
- City
- Country
- Country_Regior
- Country_Resorl
- Customer

Classes and Objects:
- Test

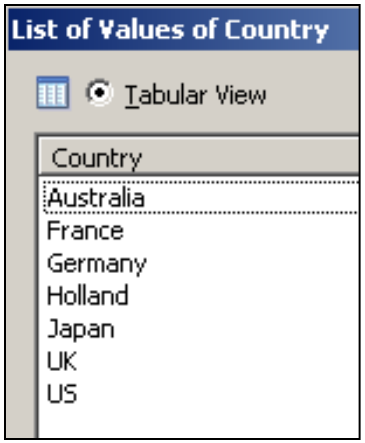
Operators:
- -
- *
- /
- +

Functions:
- Number
- Character
- Date
- @Functions

List of Values

3.1 4.x

- Gives the users a “cheat sheet” of object values
  - Used to complete query conditions
- Steps to create this list:
  - 4.x Business Layer > Parameters and List of Values
    List of values based on business layer objects
  - 3.1 Object Properties > Properties > Edit

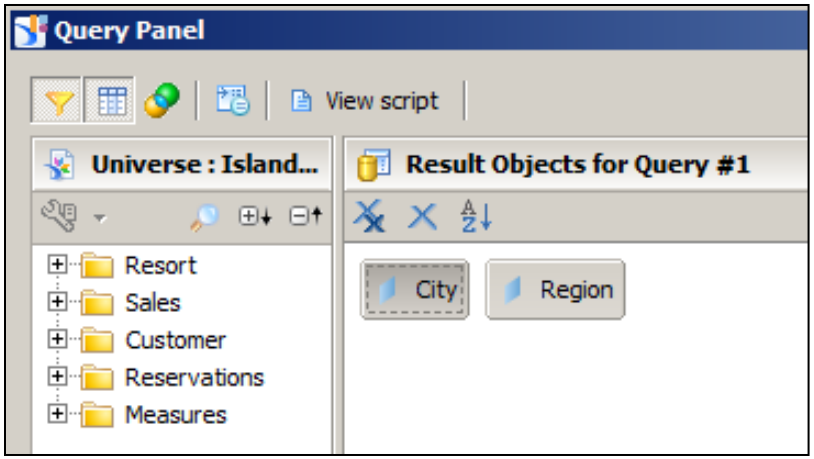


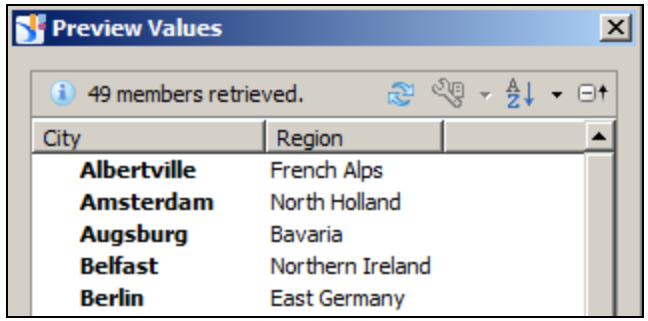
In BI 4.x, list of values can also be created from a static list or custom SQL statements as part of the Data Foundation layer

List of Values, cont'd

3.1 4.x

- The List of Values editor looks just like a Web query





The List of Values query can have more than one object, as long as the **leftmost** object represents the final value for the list,

Demonstration

3.1 4.x

Agenda

- Introduction
- Getting started
- Making a connection
- Building the foundation
- Resolving inconsistencies
- Creating classes and objects
- **Releasing the final version**
- Conclusion

A Word On Testing

3.1 4.x

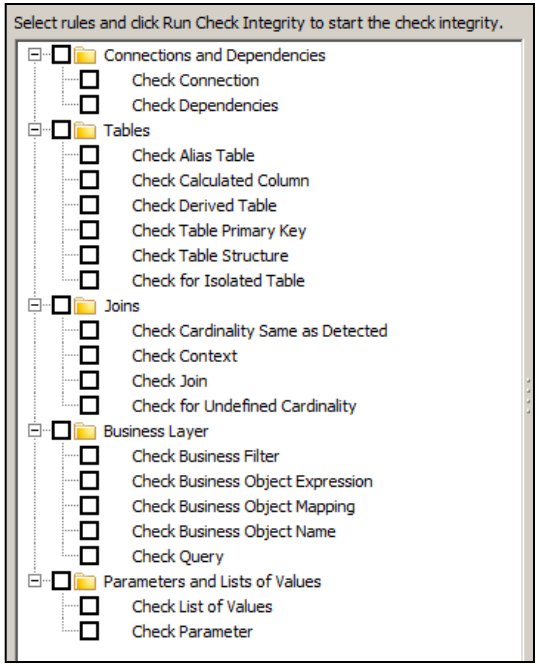
- Universe development runs in cycles
  - Add a few tables
  - Connect them with joins and resolve any problems
  - Create a few classes and objects
  - **TEST** using sample queries
    - Query editors are part of Universe Designer, IDT
    - Could also use Web Intelligence if the universe has been published
  - Repeat the process until universe is complete

Integrity Checking

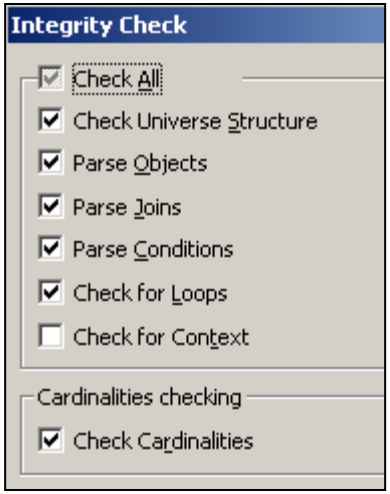
3.1 4.x

- A sanity check to make sure there are no universe problems
  - Not always 100% accurate
  - BUT … still very much worth the time to use
  - Use the **Check Integrity** button

4.x



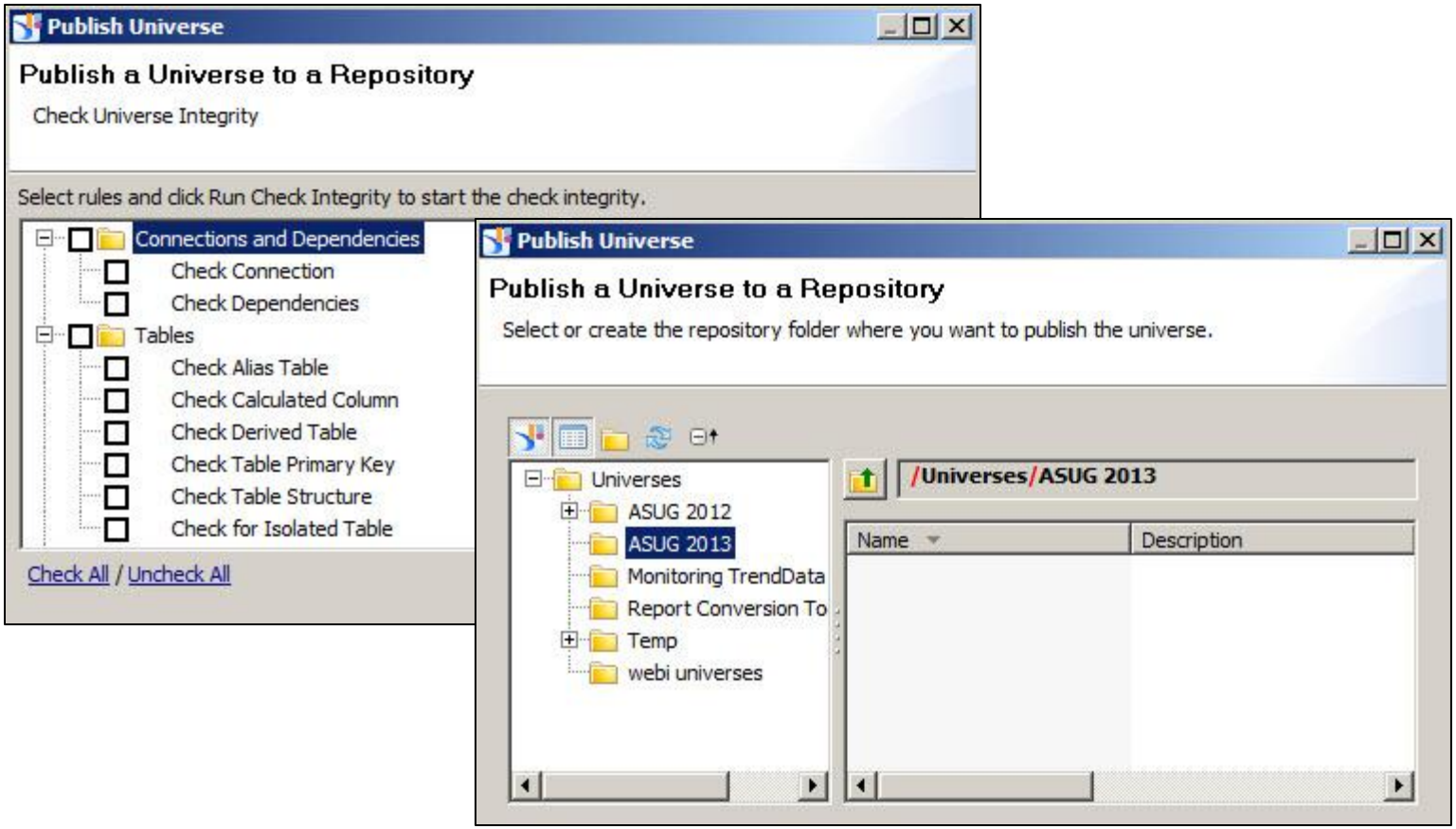
3.1



Publishing

4.x

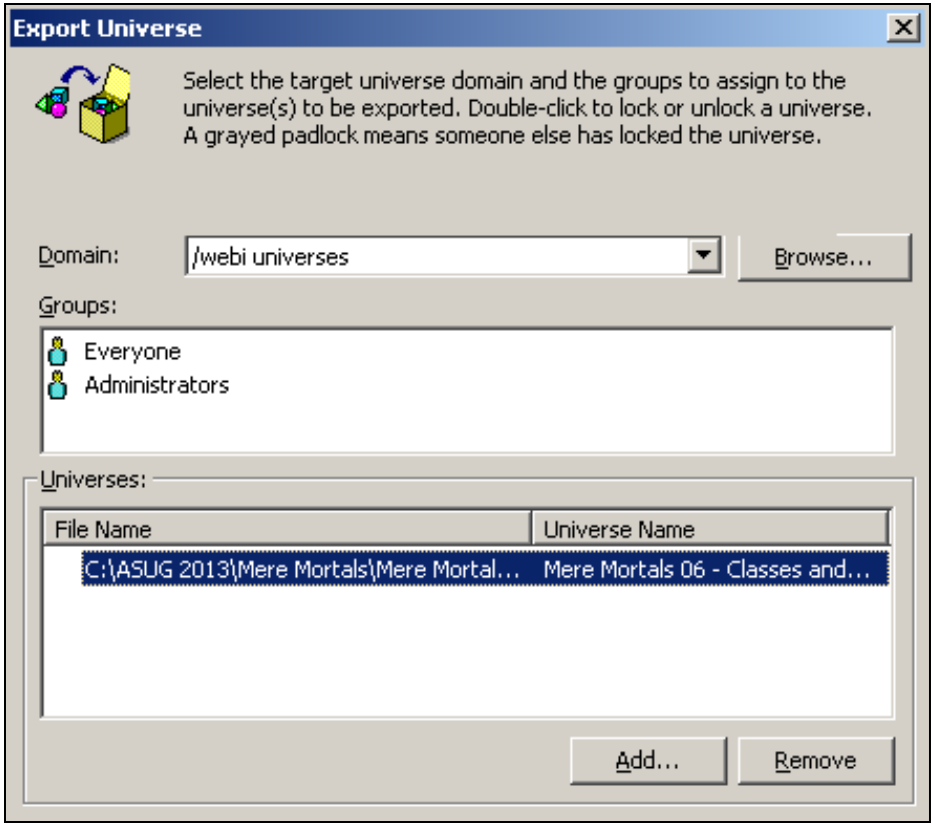
- Right-click on any business layer
  - **Publish > To a Repository…**



Exporting

3.1

- The way to publish universes in XI 3.1
  - **File** > **Export**



Domain here represents the folder that universe will be exported to. More than one universe can be exported at the same time.

Demonstration

3.1 4.x

Agenda

- Introduction
- Getting started
- Making a connection
- Building the foundation
- Resolving inconsistencies
- Creating classes and objects
- Releasing the final version
- **Conclusion**

Key Learnings

- Creating universes is easy once you know how
- This presentation showed the basics
- Download the examples to practice at home
- Are there more detailed topics?
  - Of course!
  - But this was geared for “Mere Mortals”
  - More advanced topics in future presentations

Questions?

Alan Mayer

Session 0610

Universe Building for Mere Mortals

alan.mayer@solidgrounded.com

214-295-6250 (office)

214-755-5771 (mobile)

214-206-9003 (fax)

Thank you for participating.

Please provide feedback on this session by completing a short survey via the event mobile application.

SESSION CODE: 0610

Learn more year-round at www.asug.com