DDI and Relational Databases

Alerk Amin & Ingo Barkow

April 4, 2013

Welcome

Outline

- **Introduction to RDB / Access / SQL**
- **DDI Applications – XML DB or RDB?**
- **Modeling DDI in RDB**
- **Querying a DDI Database**
- **Advanced Topics**

Introduction to RDB / Access

- **We will use Access for modelling some examples during the workshop**
- **It is not an ideal solution for DDI as the database engine is quite limited as it is not fully ANSI-SQL compliant**
- **In reality you should use a 'real' SQL engine like e.g.**
  - **MySQL**
  - **PostgreSQL**
  - **Microsoft SQL Server**
  - **Oracle**

Introduction to RDB / Access

- **Creating an empty database in Access 2010**
- **Table Editor**
- **Query Editor**
- **SQL View**

Introduction to SQL

- **Structured Query Language**
- **Access and manipulate databases**
- **SELECT, INSERT, UPDATE, DELETE**

Table - Customers

Id	Name	Organization	City
1	Peter	Initech	Austin
2	Michael	Initech	Austin
3	Milton	Initech	Nassau
4	Joanna	Chotchkies	Austin
5	Bob	Consultant	Dallas

SELECT

- **SELECT * FROM Customers**

Id	Name	Organization	City
1	Peter	Initech	Austin
2	Michael	Initech	Austin
3	Milton	Initech	Nassau
4	Joanna	Chotchkies	Austin
5	Bob	Consultant	Dallas

SELECT

- **SELECT name, organization FROM customers WHERE city='Austin' ORDER BY name**

Name	Organization
Joanna	Chotchkies
Michael	Initech
Peter	Initech

INSERT

- **INSERT INTO customers (id, name, organization, city) VALUES (NULL, 'Samir', 'Initech', 'Austin')**

Id	Name	Organization	City
1	Peter	Initech	Austin
2	Michael	Initech	Austin
3	Milton	Initech	Nassau
4	Joanna	Chotchkies	Austin
5	Bob	Consultant	Dallas
6	Samir	Initech	Austin

UPDATE

- **UPDATE customers SET organization = 'Construction' WHERE id = 1**

Id	Name	Organization	City
1	Peter	Construction	Austin
2	Michael	Initech	Austin
3	Milton	Initech	Nassau
4	Joanna	Chotchkies	Austin
5	Bob	Consultant	Dallas
6	Samir	Initech	Austin

DELETE

- **DELETE FROM customers WHERE id = 2**

Id	Name	Organization	City
1	Peter	Construction	Austin
3	Milton	Initech	Nassau
4	Joanna	Chotchkies	Austin
5	Bob	Consultant	Dallas
6	Samir	Initech	Austin

Tables – Customers, Orders

id	name	organization	city
1	Peter	Construction	Austin
3	Milton	Initech	Nassau
4	Joanna	Chotchkies	Austin
5	Bob	Consultant	Dallas
6	Samir	Initech	Austin

id	customerId	item	amount
1	3	Stapler	1
2	4	Pins	8
3	4	Buttons	7
4	1	Stapler	2

JOIN

- **SELECT name, item,amount FROM customers JOIN orders ON customers.id = orders.customerid WHERE customers.id = 4**

name	item	amount
Joanna	Pins	8
Joanna	Buttons	7

Join Tables

- **For many-to-many relationships, a join table is used**

Id	vehicle
1	car
2	Boat
3	Plane

vehicleid	partid
1	1
1	2
2	2
2	4
3	1
3	2
3	3

Id	part
1	tires
2	frame
3	wings
4	anchor

Join Tables SQL

- **SELECT * FROM vehicles JOIN parts_vehicles ON vehicles.id = parts_vehicles.vehicleid JOIN parts ON parts_vehicles.partid = parts.id**

DDI Applications

- **What is XML?**
- **eXtensible Markup Language**
- **Used as a common import and export language between applications**
- **Superset to HTML/XHTML where own tags can be defined**

DDI Applications

- **XML Example**

```
<Book>
    <Title>  The Hitchhiker's Guide to the Galaxy  </Title>
    <Author> Douglas Adams </Author>
    <Year>  1979  </Year>
</Book>
```

DDI Applications

- **XML schema**
- **To formalize XML a XML schema can be used to define the tags and content of an XML file**
- **This allows applications to validate against the schema to see if the contained information is correct**

DDI Applications

- **Example of a XML schema – German postcodes (format is e.g. “D-60486” for Frankfurt am Main)**

```
<xs:simpleType name="postcodesGER">
  <xs:restriction base="xs:string">
    <xs:pattern value="(D )?[0-9]{5}"/>
  </xs:restriction>
</xs:simpleType>
```

DDI Applications

- **DDI and XML**
- **DDI Lifecycle is represented as a XML schema (XSD) making this setting its native form**
- **Many organizations store DDI metadata either as XML instances in files or XML databases (e.g. eXist-DB)**
- **Here an example of an empty DDI instance**

DDI Applications

```
<DDIInstance>
      <StudyUnit> ... </StudyUnit>

      <ResourcePackage>
            <QuestionScheme />
            <VariableScheme />
            <ConceptScheme />
            <PhysicalInstance />
      </ResourcePackage>

</DDIInstance>
```

DDI Applications

- **Nevertheless some organizations have already existing relational database systems and want to combine metadata and data therefore XML is not ideal**
- **Challenges in this respect are working with different versions of DDI (e.g. DDI-L 3.0, 3.1, 3.2) and ensuring application compatibility with the original standard**

DDI Applications

- **Pros of relational databases in regards to DDI**
  - **Structure is very good for rectangular files (e.g. SPSS or Stata)**
  - **Easier combination between metadata and microdata by using the same storage structure (e.g. by referential integrity)**
  - **Very common structure with high degree of optimization (e.g. indexes, file groups, stored procedures)**
  - **Capability to store multiple studies in one database system (more opportunity for harmonization between studies)**
  - **Internal independence of DDI version (can be adapted in the import and export processes on each individual version)**

DDI Applications

- **Pros of XML structures in regards to DDI**
  - **XML is native to DDI therefore no compatibility issues (e.g. unknown nodes do not have necessarily to be processed)**
  - **Hierarchical structure is difficult to model in relational databases**
  - **Full set of DDI leads to a very complex relational database with heavy response times due to complex joins (nevertheless most DDI-XML implementations only use a subset)**
  - **DDI-XML can easier be verified against the DDI schema**
- **An interesting approach is to use a hybrid relational database with XML acceleration or processing (e.g. enterprise databases like SQL Server or Oracle)**

Modeling DDI in RDB

- **Takes time and effort**
- **DDI has many different relationships between its elements**
- **Each relationship requires attention when creating a RDB model**
- **There is no one “correct” solution – every application has different requirements**

DDI Element - Citation

XML Representation Summary

```
<Citation>
   Content: Title+, SubTitle*, AlternateTitle*, Creator*, Publisher*, Contributor*, PublicationDate?,
            Language?, InternationalIdentifier*, Copyright?, dc:DCElements?
</Citation>
```

Citation - Title

Title

Type: InternationalStringType, simple content

Full authoritative title. Field may be repeated to document multiple languages.

Simple Content

```
xs:string
```

XML Source *(w/o annotations (1); see within schema source)*

```
<xs:element maxOccurs="unbounded" ref="Title"/>
```

Citation Table

Id	Title	Subtitle	Alternate Title	…

DDI Attributes - TextDomain

XML Representation Summary

```
<TextDomain
   blankIsMissingValue = xs:boolean
   classificationLevel = ("Nominal" | "Ordinal" | "Interval" | "Ratio" | "Continuous")
   maxLength           = xs:integer
   minLength           = xs:integer
   missingValue        = xs:NMTOKENS
   regExp              = xs:string
   >
   Content: RecommendedDataType?, GenericOutputFormat?, Label*, Description*
</TextDomain>
```

TextDomain - SubElements

GenericOutputFormat

Type: CodeValueType, simple content

This field provides a recommended generic treatment of the data for display by an application. The value should come from a controlled vocabulary.

Simple Content

```
xs:string
```

XML Source *(w/o annotations (1); see within schema source)*

```
<xs:element minOccurs="0" name="GenericOutputFormat" type="CodeValueType"/>
```

RecommendedDataType

Type: CodeValueType, simple content

This field provides the recommended treatment of the data within an application. The value should come from a controlled vocabulary - recommended values include the set found in W3C XML Schema Part 2, but excluding string sub-types, QNAME, and NOTATION.

Simple Content

```
xs:string
```

XML Source *(w/o annotations (1); see within schema source)*

```
<xs:element minOccurs="0" name="RecommendedDataType" type="CodeValueType"/>
```

TextDomain table

ID	Label	Description	Blank…	Classification-levelid	Min Length	Max length	…

XML Hierarchy

- StudyUnit
  - ConceptualComponent
    - ConceptScheme
    - GeographicLocationScheme
    - UniverseScheme
  - DataCollection
    - QuestionScheme
    - ControlConstructScheme
    - Instrument
  - LogicalProduct

References

- DDI make extensive use of references
- Key to “reusability” in DDI
- Refer to items in any DDI Instance
- Element can reference another element
  - In the database
  - Not in the database

References

- **DDI References are “one direction”**
  - **Variables reference QuestionItems**
  - **QuestionItems do not reference Variables**
- **But relationships are really in both directions**
- **Analyze the elements to model the relationship**

Variable

XML Representation Summary

```
<Variable
   action         = ("Add" | "Update" | "Delete")
   id             = xs:string
   isGeographic   = xs:boolean : "false"
   isTemporal     = xs:boolean : "false"
   isVersionable  = "true"
   isWeight       = xs:boolean : "false"
   objectSource   = xs:anyURI
   urn            = xs:anyURI
   version        = xs:string
   versionDate    = (xs:dateTime | xs:date | xs:gYearMonth | xs:gYear | xs:duration)
   >
   Content: UserID*, VersionResponsibility?, VersionRationale*, VariableName*, r:Label*, r:Description*,
            r:UniverseReference*, ConceptReference?, QuestionReference*, EmbargoReference?, ResponseUnit?,
            r:AnalysisUnit?, Representation?
</Variable>
```

References

- If “One-to-Many” (single or ?)
  - Concept?
  - Embargo?
  - Model with Foreign Key

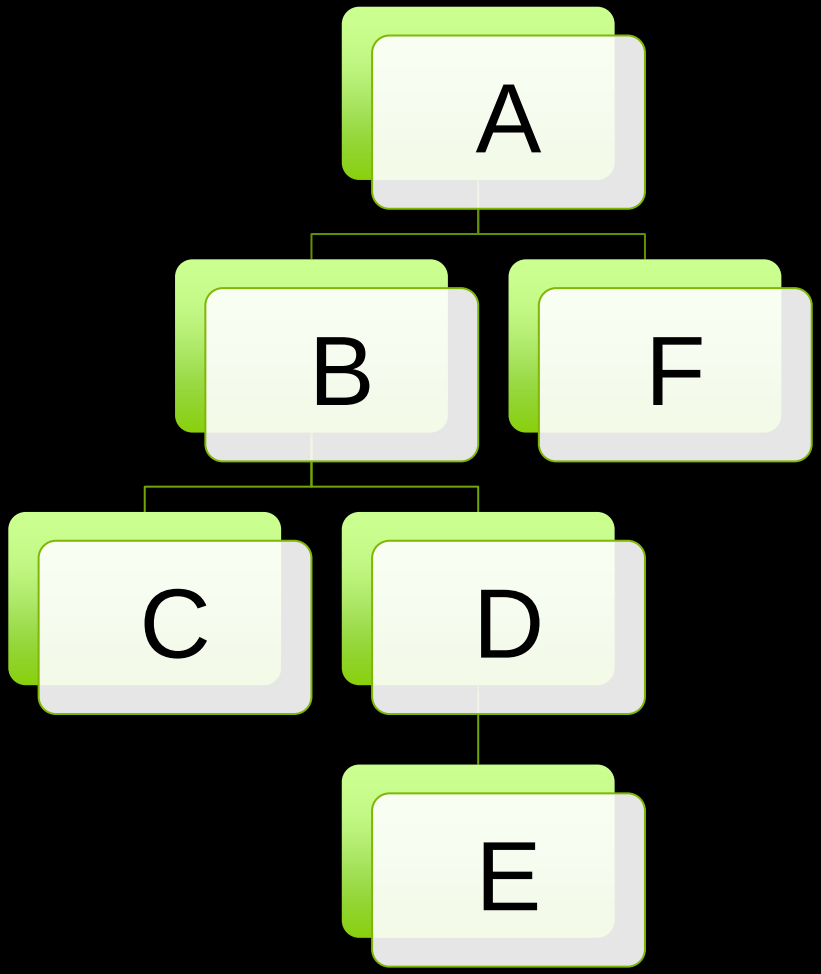
- If “Many-to-Many” (+ or *)
  - Universe*
  - QuestionItem*
  - Model with Join Table

Recursive Structures

- **DDI Elements that have sub-elements of the same type**
  - **Groups, ControlConstructs**
- **One-to-many relationships with themselves**

Recursive Structures

- **Simple method is a foreign key for parent**
  - **Points to the same table**
- **Better option is to use a Tree**
  - **Options include Path Enumeration, Nested Set**


A
B
F
C
D
E

Path Enumeration

Id	Name	Parent	Path
1	A	Null	A
2	B	A	A/B
3	C	B	A/B/C
4	D	B	A/B/D
5	E	D	A/B/D/E
6	F	A	A/F

Nested Set

Id	Name	Left	Right
1	A	1	12
2	B	2	9
3	C	3	4
4	D	5	8
5	E	6	7
6	F	10	11

Substitution Groups

- **Many DDI Elements serve as placeholders**
- **ResponseDomain, ControlConstruct, ValueRepresentation**

ResponseDomain

Namespace: ddi:datacollection:3_1
Type: *RepresentationType*
Content: complex, 3 attributes, 2 elements
Abstract: *(may not be used directly in instance XML documents)*
Subst.Gr: may be substituted with 6 elements
Defined: globally in datacollection.xsd, see XML source
Used: at 8 locations

XML Representation Summary

```
<ResponseDomain
   blankIsMissingValue = xs:boolean
   classificationLevel = ("Nominal" | "Ordinal" | "Interval" | "Ratio" | "Continuous")
   missingValue        = xs:NMTOKENS
   >
   Content: RecommendedDataType?, GenericOutputFormat?
</ResponseDomain>
```

Content model elements (2):

r:GenericOutputFormat (type CodeValueType), r:RecommendedDataType (type CodeValueType)

Included in content model of elements (2):

QuestionItem, StructuredMixedResponseDomain

May be substituted with elements (6):

CategoryDomain, CodeDomain, DateTimeDomain, GeographicDomain, NumericDomain, TextDomain

Substitution Groups

- **Model using Inheritance**
- **3 different possibilities**
  - **Multiple tables with “superclass” table**
  - **Multiple tables without “superclass” table**
  - **Single table**

Multiple Tables with "superclass" table

- **ResponseDomain – id, blankmissing**
  - **TextDomain – responsedomainid, minlen, maxlen**
  - **NumericDomain – responsedomainid, min, max**
  - **CodeDomain – responsedomainid, …**

- **Space efficient, but queries require joins**

Multiple Tables without "superclass" table

- TextDomain – id, blankmissing, minlen, maxlen
- NumericDomain – id, blankmissing, min, max
- CodeDomain – id, blankmissing, …

- Space efficient, but queries require many joins, references are difficult to model, ids are difficult to manage

Single Table

- One table contains all fields for all possible substitute elements

- ResponseDomain
  - id, blankmissing, type, minlen, maxlen, min, max…
  - The “type” field indicates which subtitute element is being used for this row

- Fast queries but inefficient space

Substitution Groups

- **Model using**
  - **Multiple tables with “superclass” table**
  - **Single table**

Controlled Vocabularies

- **Some DDI fields have values that come from a Controlled Vocabulary**
- **Managed by the DDI Alliance Controlled Vocabulary Working Group**
- **AnalysisUnit, LifecycleEventType, ResponseUnit, …**

Response Unit

XML Representation Summary

```
<ResponseUnit
   translatable = xs:boolean : "true"
   translated   = xs:boolean : "false"
   xml:lang     = xs:language
   >
   Content: { xs:string }
</ResponseUnit>
```

Response Unit Controlled Vocabulary

Value of the Code	Descriptive Term of the Code	Definition of the Code
Self	Self	Unit of response is same as unit of observation/analysis.
Informant	Informant	Unit of response is different than the unit of observation as part of the study design, and the identity of the response unit is also pre-determined by the study design. Such studies are actually called "informant surveys" and they involve, for instance, obtaining information about a community from certain select members who hold specific information, like church leaders, business owners or leaders, physicians, school teachers, civic leaders, etc. Or, if a business or organization is the unit of observation, information may be sought at different administrative/professional levels, etc.
Proxy	Proxy	Unit of response is different than the unit of analysis because the sampled unit is unavailable/unable to participate. Proxies may be allowed by the study design, but their identity is not specifically designated (for example, anyone in the household can provide information about another household member, or any available student can provide information about another student, etc.).
Interviewer	Interviewer	Data are entered directly by the interviewer, as a result of his/her own observations, and not by eliciting answers to a question.
Other	Other	Use if the response unit is known, but not found in the list.

Controlled Vocabulary

- **Model as a table for ResponseUnit**
  - **Rows in table correspond to published Controlled Vocabulary**
- **Use a foreign key for elements that include a ResponseUnit**
  - **QuestionConstruct, Ncube, Variable**

Database IDs

- Fast database performance is dependent on fast joins
- Speed of joins depends on the type chosen for foreign keys
- For ID columns in the database, choose the fastest type that your database supports
- Most common are
  - (unsigned) ints with auto_increment
  - UUID

DDI IDs

DDI 3.1 uses three forms of identification. The basic level is an *AbstractIdentifiable* which provides a urn, id, action, objectSource and UserID. An *AbstractVersionable* adds a version, version date, Version Responsibility, and Version Rationale. An *AbstractMaintainable* adds an agency, externalReferenceDefaultURI, xml:lang, and a Boolean isPublished.

DDI IDs / URNs

- **Require many database fields to implement the exact standard**
- **But can be simplified for most applications**

DDI IDs Simplified

- With the id & version number, the rest of the URN can be generated
- Many applications do not need to implement the other ID-related attributes
- For the DDI ID
  - Use the database id?
    - If using auto_increment, then no
    - If using UUID, then maybe

DDI IDs Simplified Even More (Abused?)

- **Some DDI applications have multiple versions of an element with the same version number, and use the versionDate to differentiate between them**

The attribute 'action' is used in for inheritance situations where the identified element is being added (Add) to the inherited content, updates or overrides (Update) the inherited element, or indicates that an inherited element is not being used (Delete). Elements that 'Update' or 'Delete' an inherited element will have the SAME id as the inherited element. The attribute 'objectSource' allows the user to enter the DDI URN of an object that could be included by reference, but is being entered in-line in exact detail from its source. This feature supports distribution of non-published documentation with data extracts or archival versions where the problem of broken links or difficulties with resolution services must be avoided. It allows for the retention of the link for comparability purposes which providing the content in-line.

Hands on - Exercise 1

- **Please use Access for the following:**
  - **Create an empty database**
  - **Look for the QuestionItem element in the DDI Lifecycle documentation**
  - **Create a table (using table editor or SQL) which represents this element**

COFFEE BREAK

Querying a DDI Database

- **Once model is created, querying a DDI database is straightforward**
- **Use joins to combine tables**

Query for a DDI URN

- **QuestionSchemeA:QuestionItemB**
- **SELECT * FROM QuestionItem JOIN QuestionScheme ON QuestionItem.questionschemeid = QuestionScheme.id WHERE QuestionScheme.label='A' AND QuestionItem.label='B'**

Hands on - Exercise 2

- **Please use Access for the following:**
  - **Use the Rogatus database**
  - **Try to write a query using the Query Editor or SQL View which shows all QuestionItems with the fitting QuestionScheme**

Advanced Topics

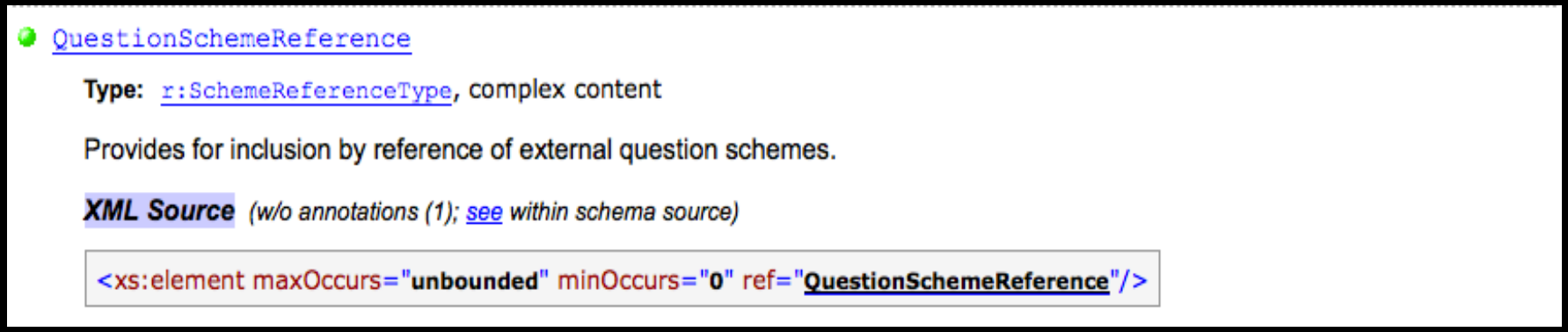
- **Versioning**
- **DDI Schemes – inclusion by reference**
- **Multi-language support**
- **Application compatibility**

Versioning

- **Versioning (including late bound references) can be established the following way in a relational database**
  - **Array of triggers on fitting tables**
  - **Managed code / external programming**
  - **Data warehouse technology (slowly changing dimensions)**

DDI Schemes

- **Many DDI Schemes can include other schemes of the same type by reference**



QuestionScheme

XML Representation Summary

```
<QuestionScheme
    action                    = ("Add" | "Update" | "Delete")
    agency                    = xs:NCName
    externalReferenceDefaultURI = xs:anyURI
    id                        = xs:string
    isMaintainable            = "true"
    isPublished               = xs:boolean : "false"
    objectSource              = xs:anyURI
    urn                       = xs:anyURI
    version                   = xs:string
    versionDate               = (xs:dateTime | xs:date | xs:gYearMonth | xs:gYear | xs:duration)
    xml:lang                  = xs:language
    >
    Content: UserID*, VersionResponsibility?, VersionRationale*, QuestionSchemeName*, r:Label*, r:Description*,
             QuestionSchemeReference*, (QuestionItem | MultipleQuestionItem)+
</QuestionScheme>
```

Modeling DDI Schemes

- **2 possible ways to model scheme inclusion by reference**

DDI Schemes – Model XML Structure

- **Create a QuestionSchemeReference table**
- **Each row in the table has a foreign key of the “source” QuestionScheme that is trying to reference the “target” QuestionScheme**
- **Effectively, this becomes a many-to-many relationship from the QuestionScheme table to itself**

SchemeReferenceType

XML Representation Summary

```
<...
    URI             = xs:anyURI
    isExternal      = xs:boolean : "false"
    isReference     = "true"
    lateBound       = xs:boolean : "false"
    objectLanguage  = xs:language
    sourceContext   = xs:anyURI
    >
    Content: Module?, Scheme?, (URN | (ID, IdentifyingAgency?, Version?))[1..2], Exclude*
</...>
```

Exclude

- Create a QuestionExclude Table
- Foreign key points to QuestionSchemeReference
- QuestionExclude has a many-to-many relationship with QuestionItems
  - Create a join table to model this relationship

Scheme References with Excluded Items

- **Why?**
- **Most common reason**
  - **Creating a new version of a scheme**
    - **Some items have been removed**
    - **Some items have changed**
- **Changed items require Comparison to document the changes**

DDI Comparison

XML Representation Summary

```
<ItemMap
   alias = xs:NCName
   >
   Content: SourceItem, TargetItem, Correspondence
</ItemMap>
```

ItemMap

- **Create a QuestionItemMap table**
- **Foreign key points to QuestionSchemeReference**
- **Foreign keys point to source QuestionItem, target QuestionItem**
- **Text field(s) for Correspondence**

Modeling Scheme Inclusion

- QuestionScheme
- QuestionItem
- QuestionSchemeReference
- QuestionExclude
- QuestionExclude_join_QuestionItem
- QuestionItemMap

Resolving a DDI Scheme

1. **Does the scheme include other schemes by reference?**
   - **If yes, first resolve the referenced scheme**
   - **Then, process the Exclude list**
2. **Process all of the items in the current scheme**
   - **Add, Update, Delete**

DDI Schemes – Model XML Structure

- Advantage
  - Stays “closer” to DDI structure
- Disadvantage
  - Implementing this will explode your brain
  - Running code that does this will melt your server
- XML DB based applications store DDI structure, so they have to deal with this
  - “cache” the resolved schemes for performance

2nd Solution for Scheme Inclusion by Reference

- **Store the resolved schemes**
- **QuestionScheme**
  - **QuestionItem**
- **Many-to-Many relationship – use a join table**
  - **Store ItemMap, Correspondence info in join table**

Storing Resolved Scheme

- **Deviates from DDI Standard**
- **Model is much simpler to implement and maintain**
- **Read/Write operations become much faster**

3rd Solution for Scheme Inclusion by Reference

- Most common use case
  - Creating a new version of a scheme
    - Some items have been removed
    - Some items have changed
- Other possibilities are less likely
  - Including schemes from a different agency
  - “Publish” a scheme that just includes other schemes

Versioning of Schemes

- **3rd solution - Don't do it!**
- **Manage the versioning of items**
- **Organize the different item version into a scheme**
  - **One-to-many relationship**
- **Don't worry about versioning the scheme**

Multi-language Support

- **Two ways for multi language support**
  - **Exporting translations into XLIFF files (XML translation standard)**
  - **Direct injection from tables into DDI-XML files while exporting**

Application Compatibility

- **Database model supports application functionality**
- **DDI XML is used to exchange metadata with other applications/agencies**
- **Can my application import/export metadata from/to that application?**

Application Compatibility

- **RDB implementations of DDI will support a subset of DDI**
- **When moving metadata from application A to application B**
  - **Does application B accept all of the fields that application A exports?**

DDI Profile

- **Collection of XPaths that describe the DDI elements used by an application**
- **Best Practices paper at DDI Alliance website**

Conclusion / Questions


RAND
CORPORATION