Druid Interchange Schema Format

Andrea Carboni, Mark D. Anderson, Antonio Gallardo, Bruno Vernay

Rev 3, 03-Jan-2004

Contents

Chapter 1

Introduction

1.1 Abstract

This document describes a XML document format that can be used to import / export database schemas using Druid. Even though this specification can be used to describe a generic database schema, the reader should be at least familiar with basic Druid concepts. See the DRUID ESSENTIALS manual for a reference.

1.2 The need

Even though a Druid project is stored in XML format, this format cannot be easily used to import / export database schemas. The problems are:

1. To use the project file format a DTD should be available. Because this file format follows every Druid's enhancement it is very dynamic so keeping the DTD updated can be difficult and error prone.

2. The format contains internal Druid stuff that should be avoided.

3. The format itself is not suited for an easy import / export. There are attributes, like serials, that are difficult to handle and don't need to be present in a general import / export format.

Given these problems, what we need is a XML file format that:

1. Is easy to generate / import,

2. Is independent from the project file format,

3. Has a simple DTD.

1.3 Assumptions

The name of the following objects are assumed to be identifiers and *must be unique*:

- tables, fields

The name of the following objects *can be any*:

- folders, notes

The name of the following objects *should be unique* in order to avoid sql errors:

- sequences, triggers

The name of the following objects *should be unique* in order to avoid sql errors (here the word name refers to the name used inside the sql code):

- views, procedures, functions

The name of the following objects *should be unique* inside the same table:

- rules

Chapter 2

The proposed format

2.1 Structure

The document is encoded in UFT-8 and represents a single database schema. The root element is named database so a generic file looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<database ...>

    ...

</database>
```

2.2 The root element

2.2.1 The database element

This is the XML root element and represents a database schema.

2.2.1.1 Attributes

Name	Value	Required	Description
name	Text	-	Name of the database
author	Text	-	Author of the document
date	Text	-	Creation date
version	Text	-	Document version
tool	Text	-	Creation tool

2.2.1.2 Subelements

Name	Cardinality	Description
description	0-1	General database description
extraSql	0-n	Extra sql at database level
folder	0-n	A container for other objects
table	0-n	A generic table
view	0-n	A generic view
procedure	0-n	A generic procedure
function	0-n	A generic function
sequence	0-n	A generic sequence
notes	0-n	Generic notes. Maps to Druid's notes object

2.2.1.3 Example

```
<database name=”TestDB”>

    <extraSql ...> ... </extraSql>
    <table ...> ... </table>

</database>
```

2.2.2 The extraSql element

Contains extra sql commands at database level. This sql is intended to be added (as is) to the generated sql script before (pre) or after (post) it. Each statement should not end with a semi-colon (;). Multiple statements can be added using several extraSql elements.

2.2.2.1 Attributes

Name	Value	Required	Description
type	pre | post	-	Specifies pre-sql or post-sql commands. Default = *pre*

2.2.2.2 Example

```
<extraSql type=”post”>


    INSERT INTO Patients(name, surname) VALUES(’John’,’Doe’)

<extraSql>
```

2.3 Elements for tables and fields

2.3.1 The table element

Specifies a generic table.

2.3.1.1 Attributes

Name	Value	Required	Description
name	Text	yes	Table's name
comment	Text	-	Comments are created using the COMMENT statement

2.3.1.2 Subelements

Name	Cardinality	Description
description	0-1	Table's description
field	0-n	A field of this table
extraSql	0-n	Extra sql at table level (for example INSERT statements).
foreignKey	0-n	Specifies integrity constraints.
index	0-n	A generic table index (unique or not).
trigger	0-n	A generic table trigger
rule	0-n	A rule (generic constraint created with the CHECK clause).
var	0-n	A table variable
attrib	0-n	Define a value for an attrib at table level.

2.3.1.3 Example

```
<table name=”Patients”>
   <field name=”code” type=”int” />
   <field name=”surname” type=”varchar” size=”32” >

      <attrib name=”not null” />
   </field>
   <index name=”patient_surname” unique=”no”>
      <field name=”surname” />
   </index>
   <attrib name=”primary key”>
      <field name=”code” />
   </attrib>
</table>
```

2.3.2 The field element

Describes a generic table field. All field's attribs (primary key, not null, unique) are defined using the attrib subelement.

2.3.2.1 Attributes

Name	Value	Required	Description
name	Text	yes	Field's name.
comment	Text	-	Comments are created using the COMMENT statement.
type	Text	*	Field's datatype, for example VARCHAR.
size	Text	*	Datatype's size, for example 32.

The type and size attribs are not used, and therefore can be omitted, only if the field is a reference to another one (in which case the datatype of the referenced field will be used). If the field is not a reference, the type attribute *must* be supplied. In this case, if the size attribute is omitted the type is considered of constant type. Otherwise, the type is considered of variable size. A decimal size can be specified as well (example '10,4').

2.3.2.2 Subelements

Name	Cardinality	Description
description	0-1	Field's description.
attrib	0-n	Specifies a value for a field's attrib at field level.

The attrib subelement has a required name and an optional value attributes. If the value attribute is missing, the attrib is supposed to be of boolean type. Otherwise it is of string type.

2.3.2.3 Example

```
<field name="code">
   <attrib name="not null" />
</field>
```

2.3.3 The extraSql element

This element can be used to provide sql statements that init the database (like INSERT statements that fill the table). Each statement should not end with a semi-colon (;). Multiple statements can be added using several extraSql elements.

2.3.3.1 Example

```
<extraSql type="command">
```

```
    INSERT INTO Patients(surname, name) VALUES('doe', 'john')
</extraSql>
```

2.3.4 The foreignKey element

Define a generic foreign key constraint.

2.3.4.1 Attributes

Name	**Value**	**Required**	**Description**
name	Text	-	Name of the foreign key.
refTable	Text	yes	Name of the referenced table.
onUpdate	noaction | default | null | cascade	-	On update action. Default = *noaction*
onDelete	noaction | default | null | cascade	-	On delete action. Default = *noaction*

2.3.4.2 Subelements

Name	**Cardinality**	**Description**
binding	0-n	Binding between a field of this table and a foreign field.

binding is a subelement that has two (required) text attributes, locField and refField.

2.3.4.3 Example

```
<foreignKey refTable="Languages" onDelete="cascade">

    <binding locField="langCode" refField name="code" />
</foreignKey>
```

2.3.5 The index element

Specifies a generic index that can be unique or not.

2.3.5.1 Attributes

Name	**Value**	**Required**	**Description**
name	Text	-	Index's name.
unique	yes | no	-	Indicates if the index is unique. Default = *no*

2.3.5.2 Subelements

Name	Cardinality	Description
field	0-n	Indicate all fields that build the index

The field subelement contains only the (required) name attribute that indicates the table's field to use to build the index. The field must exists in the table's structure.

2.3.5.3 Example

```
<index name="patients_surname_name" unique="no">
   <field name="surname" />
   <field name="name" />
</index>
```

2.3.6 The trigger element

A generic table trigger.

2.3.6.1 Attributes

Name	Value	Required	Description
name	Text	yes	Trigger's name.
activation	before | after | instead	-	When the trigger should be activated. Default = *before*
onInsert	yes | no	-	Trigger activated on INSERT. Default = *no*
onUpdate	yes | no	-	Trigger activated on UPDATE. Default = *no*
onDelete	yes | no	-	Trigger activated on DELETE. Default = *no*
forEach	row | statement	-	Data affected by trigger code. Default = *row*
when	Text	-	Generic activation condition.

2.3.6.2 Subelements

Name	Cardinality	Description
description	0-1	Trigger's description.
code	1-1	Trigger's code goes here.

2.3.6.3 Example

```
<trigger name="patient_insert" activation="before" onInsert="yes" onU

   <code>... trigger's code goes here ...</code>
</trigger>
```

2.3.7 The rule element

A generic table constraint created using the CHECK statement. Must not end with a semi-colon (;).

2.3.7.1 Attributes

Name	Value	Required	Description
name	Text	-	Rule's name.
use	yes | no	-	Tells if the rule should be used during sql generation. Default = *yes*

2.3.7.2 Subelements

Name	Cardinality	Description
description	0-1	Rule's description.
code	1-1	Rule's code goes here.

2.3.7.3 Example

```
<rule name="age_check">
   age BETWEEN 1 AND 100
</rule>
```

2.3.8 The var element

Indicate a table's var. Variables are programming aids used to define constants. See the DRUID ESSENTIALS manual for more information.

2.3.8.1 Attributes

Name	Value	Required	Description
name	Text	yes	Variable's name.
type	bool | string | int | long | char | float | double	-	Default = *string*
value	Text	yes	Variable's value.
descr	Text	-	

2.3.8.2 Subelements

Name	Cardinality	Description
description	0-1	Var's description.

2.3.8.3 Example

```
<var name="STATUS_OK" type="int" value="0" />
```

2.3.9 The attrib element

Specifies field attribs at table level. Common attribs are “primary key” and “unique”.

2.3.9.1 Attributes

Name	Value	Required	Description
name	Text	yes	Name of the field attrib

2.3.9.2 Subelements

Name	Cardinality	Description
field	0-n	Indicate all fields that belong to the atttrib

There is only a field subelement which has a (required) name attribute.

2.3.9.3 Example

```
<attrib name=”primary key”>
   <field name=”code” />
</attrib>
```

2.4 Other database objects

2.4.1 The folder element

This element is a container for other objects.

2.4.1.1 Attributes

Name	Value	Required	Description
name	Text	yes	Folder’s name

2.4.1.2 Subelements

Name	Cardinality	Description
description	0-1	Folder’s description
folder	0-n	A container for other objects
table	0-n	A generic table
view	0-n	A generic view
procedure	0-n	A generic procedure
function	0-n	A generic function
sequence	0-n	A generic sequence
notes	0-n	Generic notes. Maps to Druid’s notes object

2.4.1.3 Example

```
<folder name="Customer tables">

   <table ...> ... </table>
   <folder name="Old customers">
      <table ...> ... </table>
   </folder>

</folder>
```

2.4.2 The view element

Represents a view. Definition should not end with a semi-colon (;). The view's name is that found inside the sql code.

2.4.2.1 Subelements

Name	Cardinality	Description
description	0-1	View's description.
code	1-1	View's code goes here.

2.4.2.2 Example

```
<view>

   <code>

      CREATE VIEW PatientNames(name) AS SELECT name FROM Patients
   </code>

</view>
```

2.4.3 The procedure element

Represents a procedure. Definition should not end with a semi-colon (;). The procedure's name is that found inside the sql code.

2.4.3.1 Subelements

Name	Cardinality	Description
description	0-1	Procedure's description.
code	1-1	Procedure's code goes here.

2.4.3.2 Example

```
<procedure name=”Test”>
   <code>CREATE PROCEDURE ...</code>
</procedure>
```

2.4.4 The function element

Represents a function. Definition should not end with a semi-colon (;). The function’s name is that found inside the sql code.

2.4.4.1 Subelements

Name	Cardinality	Description
description	0-1	Function’s description.
code	1-1	Function’s code goes here.

2.4.4.2 Example

```
<function name=”Test”>
   <code>CREATE FUNCTION ...</code>
</function>
```

2.4.5 The sequence element

A generic sequence.

2.4.5.1 Attributes

Name	Value	Required	Description
name	Text	yes	Sequence’s name
increment	Text	-	Increment step
minValue	Text	-	Minimum allowed value
maxValue	Text	-	Maximum allowed value
start	Text	-	Initial sequence value
cache	Text	-	Values to keep in memory for faster access
cycle	yes | no	yes	Restart after the min/max value has been reached
order	yes | no	yes	Sequence’s order

2.4.5.2 Subelements

Name	Cardinality	Description
description	0-1	Sequence’s description.

2.4.5.3 Example

```
<sequence name="patients_seq" start="1" increment="1" />
```

2.4.6 The notes element

This element represents generic notes and maps to the druid's notes object.

2.4.6.1 Attributes

Name	Value	Required	Description
name	Text	yes	Notes' name
type	info | alert | danger	yes	Notes' severity (see DRUID ESSENTIALS)

2.4.6.2 Subelements

Name	Cardinality	Description
description	0-1	Notes' description.

2.4.6.3 Example

```
<notes name="Generic notes" type="info">
   <description>

      Generic database notes that don't fit anywhere.
   </description>
</notes>
```

2.5 The description element

This element can be used to supply an object's description. It doesn't have attributes and the description should be provided as the element's body. Example:

```
<description>
   This is a generic description.
</description>
```

Future revisions of this specification may allow a structured element.

Chapter 3

A working example

Chapter 4

Frequently asked questions

Question *How do description and comment relate to each other ?*

The description is a generic text used during docs generation while the comment is a string used to create the COMMENT sql statement.

Question *What about ’check’ constraints ?*

They are called rules here.

Question *What are notes ?*

Notes are druid specific objects that hold important information that doesn’t fit into a table or a field.

Question *How to indicate compound primary keys ?*

Use the attrib element of a table. Example:

```
<table>
   <field name=”userID” type=”int” />
   <field name=”langID” type=”int” />
   ...
   <attrib name=”primary key”>
      <field name=”userID” />
      <field name=”langID” />
   </attrib>
</table>
```

Chapter 5

Under evaluation

- Autoincrement attribute for fields.
- Templates substitution (probably added after Druid 4.0).
- Default values for triggers and other objects.
- Sql domains (probably added after Druid 4.0).
- Dialect attrib for extraSql elements.
- Table inheritance (probably added after Druid 4.0).