LSID Specification Language for Relational Databases based on SQL Server 2000

Sapna Bafna and Daniel Miranker*
Department of Computer Sciences
University of Texas at Austin
{sapna, miranker}@cs.utexas.edu

1 Introduction

This document describes the specification of a domain specific language to introduce support for LSIDs in an existing archival database. The language constructs can be used to specify subsets of the existing relational schema that need to be exported via LSIDs.

A brief outline of the procedure is given below:

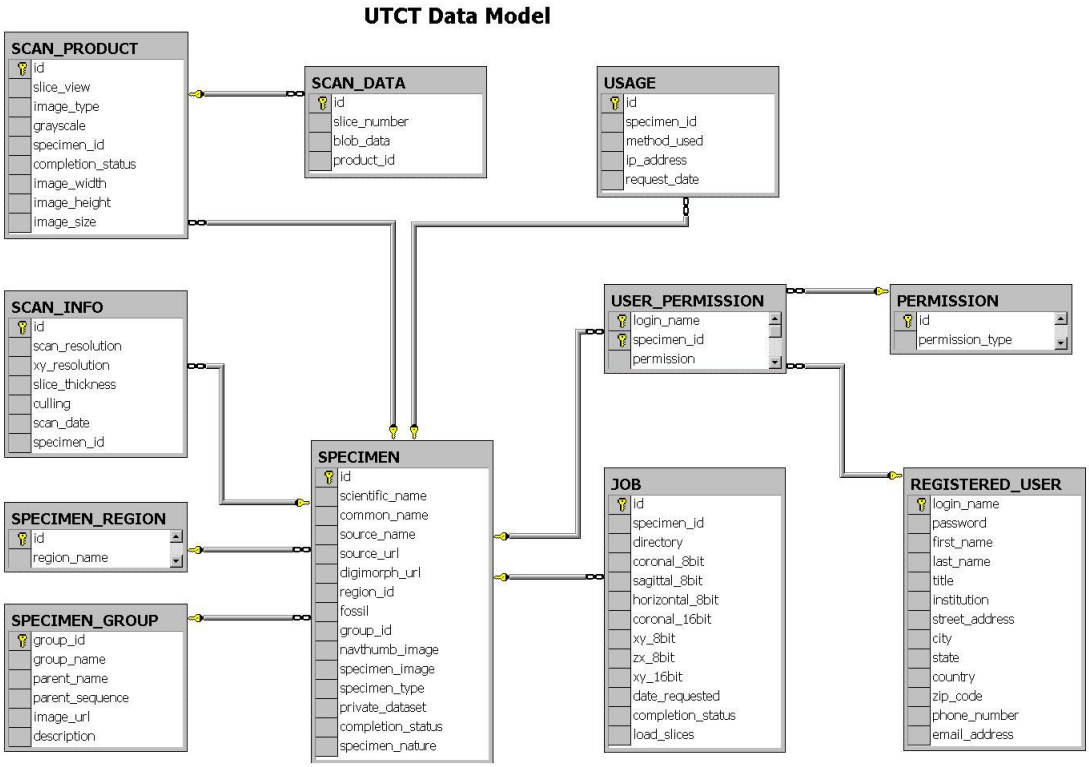
- Let OrigSchema represent the existing database schema
- The subset schema, ExportSchema is specified using the LSID language constructs
- A translator (compiler)
  1. Creates persistent tables that maps the OrigSchema to the ExportSchema
  2. Creates LSID runtime tables
  3. Installs triggers on OrigSchema to update the persistent and runtime tables on updates (inserts) to the original database

The following section gives the technical details and SQL queries to introduce LSIDs in an existing life sciences archival database instance. Some discussion points have been discussed in Section 3.

* This research is funded by the National Science Foundation grant IIS-0531767.

2 SQL Server 2000 Example

2.1 Original Schema



2.2 Export Schema

DarwinCoreRecord (GlobalUniqueIdentifier, DateLastModified, BasisOfRecord, InstitutionCode, CollectionCode, CatalogNumber, ScientificName, ImageURL, RelatedInformation)

SpecimenImage (ScientificName, SpecimenImage)

SpecimenSlice (ScientificName, SliceNumber, SliceImage)

2.3 LSID Specification

```
CREATE      LSIDVIEW DarwinCoreRecord AS
SELECT      SpecimenImage.LSID      AS    GlobalUniqueIdentifier
            scan_date               AS    DateLastModified,
            'ct dataset'            AS    BasisOfRecord,
            'utexas'                AS    InstitutionCode,
            'utct'                  AS    CollectionCode,
            specimen_id             AS    CatalogNumber, scientific_name
                        AS    ScientificName,
           'http://utct-test.tacc.utexas.edu/data.php?specimen_id=' +
            specimen_id             AS    ImageURL,
```

```
            ''                          AS    RelatedInformation
FROM        Specimen, Scan_Info
WHERE       Specimen.id = Scan_Info. specimen_id

CREATE      LSIDVIEW  SpecimenImage AS
SELECT      scientific_name, specimen_image
FROM        Specimen

CREATE      LSIDVIEW  SpecimenSlice AS
SELECT      scientific_name, slice_number, blob_data
FROM        Specimen, Scan_Data
WHERE       Specimen.id = Scan_Product.specimen_id
      AND   Scan_Data.product_id = Scan_Product.id
      AND   Scan_Product.image_type = 'JPG'
```

2.4 Permanent Tables created by the Translator

A copy of the exported data will be maintained in these tables, to support persistence of LSID data. The table definitions will be created per the LSIDVIEW specification.

DarwinCoreRecord	
LsObjID	Primary key
version#	
origPKSpecimen	Primary key of original data from Specimen, required for versioning.
origPKScan_Info	Primary key of original data from Scan_Info, required for versioning.
GlobalUniqueIdentifier	Attribute from original database
DateLastModified	Attribute from original database
BasisOfRecord	Attribute from original database
InstitutionCode	Attribute from original database
CollectionCode	Attribute from original database
CatalogNumber	Attribute from original database
ScientificName	Attribute from original database
ImageURL	Attribute from original database
RelatedInformation	Attribute from original database

SpecimenImage	
LsObjID	Primary key
version#	
OrigPK	Primary key of original data, required for versioning.
scientific_name	Attribute from original database
specimen_image	Attribute from original database

SpecimenSlice	
LsObjID	Primary key

version#	
origPKSpecimen	Primary key of original data from Specimen, required for versioning.
origPKScan_Data	Primary key of original data Scan_Data, required for versioning.
scientific_name	Attribute from original database
slice_number blob_data	Attribute from original database Attribute from original database

2.5 Runtime Tables created by the Translator

These tables will store the bookkeeping information for every implementation.

LSIDTable	
LsObjID	Primary key
TableName	DarwinCoreRecord / SpecimenImage / SpecimenSlice

2.6 Triggers created on SQL Server 2000 by the Translator

Triggers to populate the persistent tables when data is submitted to the archive.

2.6.1 Insert Triggers

```
CREATE      TRIGGER PopulatePersistentSpecimenTables
ON          Specimen
INSTEAD OF  INSERT
AS

BEGIN

DECLARE     @lsObjID          INT
DECLARE     @newTuplePK       INT
DECLARE     @newTupleFK       INT
DECLARE     @scientific_name  VARCHAR(256)
DECLARE     @slice_number           INT

/* Get primary key of the tuple in newTuplePK */
SELECT      @newTuplePK=id
FROM        INSERTED

/* Get maximum lsObjID from LSID table */
SELECT      @lsObjID=MAX(lsObjID)
FROM        LSIDTable
IF          @lsObjID IS NULL
SET         @lsObjID = 0
```

```
/* Insert new tuple in DarwinCoreLSIDTable in trigger on Scan_Info */

/* Insert a new tuple in SpecimenImageLSIDTable with incremented
lsObjID, version number 1,

original PK and new attribute values) */
SET         @lsObjID = @lsObjID + 1

SELECT      @scientific_name=scientific_name
FROM        INSERTED

INSERT      INTO SpecimenImageLSIDTable
SELECT      @lsObjID, 1, @newTuplePK, @scientific_name, specimen_image
FROM        INSERTED

INSERT      INTO LSIDTable
VALUES      (@lsObjId, 'SpecimenImageLSIDTable')

/* Queries for Specimen Join Scan_Product Join Scan_Data */
DECLARE     joinCursor  CURSOR LOCAL READ_ONLY
FOR
SELECT      INSERTED.scientific_name, INSERTED.id, Scan_Data.id,
            Scan_Data.slice_number
FROM        INSERTED, Scan_Product, Scan_Data
WHERE       INSERTED.id = Scan_Product.specimen_id
      AND   Scan_Data.product_id = Scan_Product.id
      AND   Scan_Product.image_type = 'JPG'

OPEN        joinCursor
FETCH       NEXT
FROM        joinCursor
INTO        @scientific_name, @newTuplePK, @newTupleFK, @slice_number

WHILE       @@FETCH_STATUS=0
BEGIN

SET         @lsObjID = @lsObjID + 1

INSERT      INTO SpecimenSliceLSIDTable
SELECT      @lsObjID, 1, @newTuplePK, @newTupleFK, @scientific_name,
            @slice_number, blob_data
FROM        Scan_Data
WHERE       id=@newTupleFK

INSERT      INTO LSIDTable
VALUES      (@lsObjId, 'SpecimenSliceLSIDTable')

FETCH       NEXT
FROM        joinCursor
INTO        @scientific_name, @newTuplePK, @newTupleFK, @slice_number
```

```
END

CLOSE       joinCursor
DEALLOCATE  joinCursor

/* Insert new tuple in the Specimen table. Required because of use of
instead of trgiger, in order to support images */
INSERT      INTO Specimen
SELECT      *
FROM        INSERTED

END

CREATE      TRIGGER PopulatePersistentDarwinCoreTables
ON          SCAN_INFO
INSTEAD OF  INSERT
AS

BEGIN

DECLARE     @lsObjID          INT
DECLARE     @newTuplePK       INT
DECLARE     @scientific_name  VARCHAR(256)
DECLARE     @globalUniqueIdentifier VARCHAR(256)
DECLARE     @dateLastModified VARCHAR(256)
DECLARE     @basisOfRecord    VARCHAR(256)
DECLARE     @institutionCode  VARCHAR(256)
DECLARE     @collectionCode   VARCHAR(256)
DECLARE     @catalogNumber    VARCHAR(256)
DECLARE     @imageURL         VARCHAR(256)
DECLARE     @relatedInformation     VARCHAR(256)

/* Get primary key of the tuple in newTuplePK */
SELECT      @newTuplePK=specimen_id
FROM        INSERTED

/* Get maximum lsObjID from LSID table */
SELECT      @lsObjID=MAX(lsObjID)
FROM        LSIDTable
IF          @lsObjID IS NULL
SET         @lsObjID = 0

/* Insert a new tuple in DarwinCoreLSIDTable with incremented lsObjID,
version number 1, original PK and new attribute values, some of them
default) */
SET         @lsObjID = @lsObjID + 1
SET         @globalUniqueIdentifier = 'urn:lsid:utct-
            test.tacc.utexas.edu:ctdataset:' + cast
            ((@lsObjID - 1) as varchar(256))
SELECT      @dateLastModified = scan_date
FROM        INSERTED
```

```
SET         @basisOfRecord = 'ct dataset'
SET         @institutionCode = 'utexas'
SET         @collectionCode = 'utct'
SELECT      @catalogNumber=specimen_id
FROM        INSERTED
SELECT      @scientific_name=scientific_name
FROM        INSERTED, Specimen
WHERE       INSERTED.specimen_id = Specimen.id
SET         @imageURL = 'http://utct-
            test.tacc.utexas.edu/data.php?specimen_id=' + cast
            (@catalogNumber as varchar(256))

INSERT      INTO DarwinCoreLSIDTable
VALUES      (@lsObjID, 1, @newTuplePK, @globalUniqueIdentifier,
            @dateLastModified, @basisOfRecord,
            @institutionCode, @collectionCode, @catalogNumber,
            @scientific_name, @imageURL,
            @relatedInformation)

INSERT      INTO LSIDTable
VALUES      (@lsObjId, 'DarwinCoreLSIDTable')

/* Insert new tuple in the Scan_Info table. Required because of use of
instead of trgiger, in order to support images */
INSERT      INTO Scan_Info
SELECT      *
FROM        INSERTED

END

CREATE      TRIGGER PopulatePersistentSpecimenSliceTables
ON          Scan_Data
INSTEAD OF  INSERT
AS

BEGIN

DECLARE     @lsObjID          INT
DECLARE     @newTuplePK       INT
DECLARE     @newTupleFK       INT
DECLARE     @scientific_name  VARCHAR(256)
DECLARE     @slice_number           INT

/* Get maximum lsObjID from LSID table */
SELECT      @lsObjID=MAX(lsObjID)
FROM        LSIDTable
IF          @lsObjID IS NULL
SET         @lsObjID = 0

/* Queries for Specimen Join Scan_Product Join Scan_Data (INSERTED)*/
DECLARE     joinCursor  CURSOR LOCAL READ_ONLY
```

```
FOR
SELECT      Specimen.scientific_name, Specimen.id, INSERTED.id,
            INSERTED.slice_number
FROM        Specimen, Scan_Product, INSERTED
WHERE       Specimen.id = Scan_Product.specimen_id
      AND   INSERTED.product_id = Scan_Product.id
      AND   Scan_Product.image_type = 'JPG'

OPEN        joinCursor
FETCH       NEXT
FROM        joinCursor
INTO        @scientific_name, @newTuplePK, @newTupleFK, @slice_number

WHILE       @@FETCH_STATUS=0
BEGIN

SET         @lsObjID = @lsObjID + 1

INSERT      INTO SpecimenSliceLSIDTable
SELECT      @lsObjID, 1, @newTuplePK, @newTupleFK, @scientific_name,
            @slice_number, blob_data
FROM        INSERTED

INSERT      INTO LSIDTable
VALUES      (@lsObjId, 'SpecimenSliceLSIDTable')

FETCH       NEXT
FROM        joinCursor
INTO        @scientific_name, @newTuplePK, @newTupleFK, @slice_number

END

CLOSE       joinCursor
DEALLOCATE  joinCursor

/* Insert new tuple in the Specimen table. Required because of use of
instead of trgiger, in order to support images */
INSERT      INTO Scan_Data
SELECT      *
FROM        INSERTED

END
```

2.6.2 Update Triggers

```
CREATE      TRIGGER PopulatePersistentSpecimenTablesNewVersion
ON          Specimen
INSTEAD OF  UPDATE
AS

BEGIN
```

```
DECLARE     @lsObjID          INT
DECLARE     @updatedTuplePK   INT
DECLARE     @updatedTupleFK   INT
DECLARE     @originalTupleFK  INT
DECLARE     @version#         INT
DECLARE     @scientific_name  VARCHAR(256)

/* Get primary key of the tuple in updatedTuplePK */
SELECT      @updatedTuplePK=id
FROM        DELETED

IF          UPDATE(scientific_name)
BEGIN

/* Get version number of the LSID data object for this tuple already
stored in DarwinCoreLSIDTable, using original PK */
SELECT      @version#=MAX(version#)
FROM        DarwinCoreLSIDTable
WHERE       origPK=@updatedTuplePK

IF          @version#   IS NOT NULL
BEGIN

/* Get the LSID */
SELECT      @lsObjID=lsObjID
FROM        DarwinCoreLSIDTable
WHERE       origPK=@updatedTuplePK
      AND   version#=@version#

/* Insert a new tuple in DarwinCoreLSIDTable for this updated data
object with incremented version number */
SET         @version# = @version# + 1

SELECT      @scientific_name=scientific_name
FROM        INSERTED

INSERT      INTO DarwinCoreLSIDTable
SELECT      @lsObjID, @version#, @updatedTuplePK,
            GlobalUniqueIdentifier, DateLastModified,
            BasisOfRecord, InstitutionCode, CollectionCode,
            CatalogNumber, @scientific_name, ImageURL,
            RelatedInformation
FROM        DarwinCoreLSIDTable
WHERE       origPK=@updatedTuplePK
      AND   version#=@version#-1
END

/* Get version number of the LSID data object for this tuple already
stored in

SpecimenImageLSIDTable, using original PK */
```

```
SELECT      @version#=MAX(version#)
FROM        SpecimenImageLSIDTable
WHERE       origPK=@updatedTuplePK

IF          @version#   IS NOT NULL
BEGIN

/* Get the LSID */
SELECT      @lsObjID=lsObjID
FROM        SpecimenImageLSIDTable
WHERE       origPK=@updatedTuplePK
      AND   version#=@version#

/* Insert a new tuple in SpecimenImageLSIDTable for this updated data
object with incremented

version number */
SET         @version# = @version# + 1

SELECT      @scientific_name=scientific_name
FROM        INSERTED

INSERT      INTO SpecimenImageLSIDTable
SELECT      @lsObjID, @version#, @updatedTuplePK, @scientific_name,
            INSERTED.specimen_image
            FROM  INSERTED
            JOIN  Specimen
            ON    INSERTED.id=Specimen.id
END

/* Iterate through tuples in SpecimenSliceLSIDTable with
origPK=@updatedTuplePK */
DECLARE     joinCursor  CURSOR LOCAL READ_ONLY
FOR
SELECT      Scan_Data.id
FROM        INSERTED, Scan_Product, Scan_Data
WHERE       INSERTED.id = Scan_Product.specimen_id
      AND   Scan_Data.product_id = Scan_Product.id
      AND   Scan_Product.image_type = 'JPG'

OPEN        joinCursor

FETCH       NEXT
FROM        joinCursor
INTO        @updatedTupleFK

WHILE       @@FETCH_STATUS=0
BEGIN

/* Get version number of the LSID data object for this tuple already
stored in SpecimenSliceLSIDTable, using original PK and FK */
```

```
SELECT      @version#=MAX(version#)
FROM        SpecimenSliceLSIDTable
WHERE       origPK=@updatedTuplePK
      AND   origFK=@updatedTupleFK

IF          @version#   IS NOT NULL
BEGIN

/* Get the LSID */
SELECT      @lsObjID=lsObjID
FROM        SpecimenSliceLSIDTable
WHERE       origPK=@updatedTuplePK
      AND   origFK=@updatedTupleFK
      AND   version#=@version#

/* Insert new tuple in SpecimenSliceLSIDTable for the updated data
object with incremented version number */
SET         @version# = @version# + 1

SELECT      @scientific_name=scientific_name
FROM        INSERTED

INSERT      INTO SpecimenSliceLSIDTable
SELECT      @lsObjID, @version#, @updatedTuplePK, @updatedTuplefK,
            @scientific_name, slice_number,
            blob_data
FROM        SpecimenSliceLSIDTable
WHERE       origPK=@updatedTuplePK
      AND   origFK=@updatedTupleFK
      AND   version#=@version#-1

FETCH       NEXT
FROM        joinCursor
INTO        @updatedTupleFK

END
END

CLOSE       joinCursor
DEALLOCATE  joinCursor

END

ELSE IF     UPDATE(specimen_image)
BEGIN

/* Get version number of the LSID data object for this tuple already
stored in SpecimenImageLSIDTable, using original PK */
SELECT      @version#=MAX(version#)
FROM        SpecimenImageLSIDTable
WHERE       origPK=@updatedTuplePK
```

```
IF          @version#   IS NOT NULL
BEGIN

/* Get the LSID */
SELECT      @lsObjID=lsObjID
FROM        SpecimenImageLSIDTable
WHERE       origPK=@updatedTuplePK
      AND   version#=@version#

/* Insert a new tuple in SpecimenImageLSIDTable for this updated data
object with incremented version number */
SET         @version# = @version# + 1

SELECT      @scientific_name=scientific_name
FROM        INSERTED

INSERT      INTO SpecimenImageLSIDTable
SELECT      @lsObjID, @version#, @updatedTuplePK, @scientific_name,
            INSERTED.specimen_image
            FROM  INSERTED
            JOIN  Specimen
            ON    INSERTED.id=Specimen.id
END
END

/* Update the tuple in the Specimen table. Required because of use of
instead of triggers to support images */
UPDATE      Specimen
SET         common_name=(SELECT common_name from INSERTED),
            scientific_name=(SELECT scientific_name from INSERTED),
            source_name=(SELECT source_name from INSERTED),
            source_url=(SELECT source_url from INSERTED),
            digimorph_url=(SELECT digimorph_url from INSERTED),
            region_id=(SELECT region_id from INSERTED),
            fossil=(SELECT fossil from INSERTED),
            group_id=(SELECT group_id from INSERTED),
            specimen_type=(SELECT specimen_type from INSERTED),
            private_dataset=(SELECT private_dataset from INSERTED),
            completion_status=(SELECT completion_status from INSERTED),
            specimen_nature=(SELECT specimen_nature from INSERTED)
WHERE       id=@updatedTuplePK

/* Update images separately, using joins */

UPDATE      Specimen
SET         specimen_image=inserted.specimen_image
FROM        INSERTED
      JOIN  Specimen
      ON    INSERTED.id=Specimen.id
```

```
UPDATE      Specimen
SET         navthumb_image=inserted.navthumb_image
FROM        INSERTED
      JOIN  Specimen
      ON    INSERTED.id=Specimen.id

END
```

3 Discussion Points

3.1 How do we define triggers when the export schema involves joins?

An LSID data object may span across multiple tables. With each object then, we would need to store multiple primary keys. So, we need to store a list of origTableName.PK with each LSID data object.

We can then define on insert triggers on the (both) relations involved in the join. And corresponding update triggers as well.

3.2 Can we use triggers on Views?

INSTEAD OF Triggers can be defined on views. Initially, I thought we could define triggers on views so that the trigger definitions would be generic, irrespective of presence of joins in the export schema. However, we cannot define triggers on views because we need to capture insert/update events on the base (original) tables, which are fired from existing query dispatchers and cannot be updated to fire inserts/updates on views.

3.3 How do we deal with text, ntext and image attributes in inserted/deleted tables in triggers?

Attributes of type text, ntext and image cannot be accessed from the inserted and deleted tables in triggers. As a workaround, we can use INSTEAD OF triggers.

References

[1] Life sciences identifiers specification. http://www.omg.org/docs/dtc/04-05-01.pdf

[2] Garcia-Mollina, Ullman and Widom. Database Systems: The Complete Book. *Prentice Hall*

[3] Miranker, D., Bafna, S. and Humphries, J. Schema Driven Assignment and Implementation of Life Science Identifiers (LSIDs). The University of Texas at Austin, Department of Computer Sciences, Technical Report *TR-06-50*, October 16, 2006