## LexEVS 6.0 Design Document - Detailed Design - Single DB

**Document Information** 

Author: Craig Stancl Email: Stancl.craig@mayo.edu Team: LexEVS Contract: CBITT BOA Subcontract# 29XS223 Client: NCI CBIIT National Institutes of Heath US Department of Health and Human Services

## **Revision History**

Version	Date	Description of Changes	Author
1.0	5/14/10	Initial Version Approved via Design Review	Team

This document describes use of a Single Database Table versus Multiple Database Tables.

LexEVS 6.0 will include an option to load CodingSchemes into a single, common set of tables, or individual sets of tables for each CodingSchemes. Some considerations for each option:

- Single Table Advantages:
  - Limits Table proliferation in the database.
  - All Coding Schemes centralized into one table.
- Single Table Disadvantages:
  - Possible performance penalty. A query will be executed against the common table, which will be many times larger than the ontology loaded as an individual table.

To the underlying API, the actual table that data is stored in will be abstracted. All queries will except a Prefix -- whether that prefix is from the common table, or from an individual table, the query remains the same.

The prefix will be stored in the *registry* table of the database. The prefixed is then obtained given a CodingScheme URI and Version. Every CodingScheme will be loaded in the Registry, and the prefix will be retrievable given its URI and Version.

dm Registry /

registry	registryMetaData
*column»*PKregistryGuid: VARCHAR2(36)resourceURI: VARCHAR2(250)resourceType: VARCHAR2(50)resourceType: VARCHAR2(50)dbURI: VARCHAR2(250)dbURI: VARCHAR2(250)dbName: VARCHAR2(50)prefix: VARCHAR2(50)status: VARCHAR2(50)tag: VARCHAR2(50)lastUpdateDate: TIMESTAMPactivationDate: TIMESTAMPdeactivationDate: TIMESTAMPbaseRevision: VARCHAR2(50)fixedAtRevision: VARCHAR2(50)isLocked: CHAR(1)dbSchemaVersion: VARCHAR2(50)dbSchemaDescription: VARCHAR2(250)	«column» IastUpdateTime: TIMESTAMP IastUsedDBIdentifer: VARCHAR2(50) IastUsedHistoryIdentifer: VARCHAR2(50)
«PK» + PK_registry(VARCHAR2)	
«unique» + UQ_registry(VARCHAR2, VARCHAR2, VARCHAR2)	
resourceType: 1. CodingScheme 2. ValueDomain 3. PickList 4. NCIHistory	

Prefixes are automatically injected into SQL statements to direct the SQL to the appropriate table. For instance:

SELECT FROM \$prefix\$entity WHERE ...

Prior to the execution of the SQL statement, the '\$prefix\$' placeholder will be replaced with the actual prefix retrieved from the Registry.