Hi Kevin,

I apologize for not preparing better documentation of the database, but I had competing priorities.  Some of the database structure and variable names were my doing.  I will try to give some explanation here.

First, a basic design choice was to parse all fields in the source data that contain delimiter separated data elements into individual elements to be stored in separate records attached to their parent (case, panel, etc.).  These include case type, panel judges, legal topics, and LN topics.  Benefits of parsing include:

* A hierarchical structure, to enforce primary data relationships, such as prohibiting duplication of case types for a single case or of a judge in a panel
* Unambiguous search terms, so that the possible values in fields are elementary – case types are limited to criminal, civil, or other, although a case can have up to three types, stored in separate records referenced by its case header record

LN supplied four file types:

* Partnn files, the basic case data
* LNItoLNITreatments, contains citations and Shepard’s treatments
* Shepard’s treatment phrase, a dictionary of Shepard’s letters and definitions
* Judges, IDs and names of judges appearing in panels

The source Part files contained the following variables:

* LNI (LN assigned unique case ID)
* Date (renamed DecisionDate in the database, although we have not gotten confirmation that this is , in fact, the date of a decision)
* CourtShortName (referenced by CourtID)
* CourtLongName (referenced by CourtID)
* Title (renamed CaseTitleLexisNexis)
* ShortCaseName (renamed CaseTitleShort)
* FullCaseName (renamed CaseTitleLong)
* ThisCaseRefs (pipe delimited citations MADE OBSOLETE BY LNItoLNI file imported into Citations table)
* CaseTypes (pipe delimited criminal civil, or empy values, stored in CaseType table referenced by LNI)
* PubStatus (values “Reported” or “Unreported”)
* JudgeEntityList (pipe delimited panel judge IDs, stored in Panel table referenced by LNI)
* Judges (MADE OBSOLETE BY Judges file imported into Judges table)
* OpinionBy (judge name(s); LN to supply ID(s); to be placed in Opinion table with type “Opinion”)
* ConcurBy (judge name(s); LN to supply ID(s); to be placed in Opinion table with type “Concur”)
* DissentBy (judge name(s); LN to supply ID(s); to be placed in Opinion table with type “Dissent”)
* PerCuriam (values “per curiam” or empty)
* Outcome (text)
* LegalTopics (pipe delimited topic codes, stored in LegalTopics table, referenced by LNI)
* TreatedCites (delimited citations, MADE OBSOLETE BY LNtoLNI file, stored in Citations table referenced by LNI)
* InternalTopicIDs (second set of topic codes (?), named LNtopics in case header, one for each legal topic)

The LNItoLNITreatments file contained (the source file did not have column labels, so those here are inferred, contents saved in database table Citation):

* LNI (citing case)
* LNICited (cited case)
* NormCitation (citation in reporter form)
* ShepardTreatment (Shepard’s treatment letter, Citation column ShepardTreatmentID references corresponding record in ShepardTreatment table)

The Shepard’s treatment phrase file contained:

* SubCategory
* TreatmentPhrase
* Letter
* CurrentCategory
* Case/Statute
* UsageNotes
* ShepardSignal
* Definition

The Judges file contained:

* ID (numeric portion of ID appearing in Part files, ignored)
* GUID (text “urn-entity-“ plus judge ID appearing in Part files, “urn-entity-“ stripped, renamed JudgeID in Judges table)
* Name

The attached database diagram shows tables and column (variable) name equivalents to supplied files.  As noted above, several supplied columns were parsed and saved I separate tables.  These include:

* CaseType (individual types stored in CaseType table, referenced by case LNI)
* Panel (individual judge IDs stored in Panel table, referenced by case LNI)
* LegalTopics (individual topic IDs stored in CaseLegalTopics table, referenced by case LNI)
* InternalToics (individual topic IDs stored in CaseLNTopics table, referenced by case LNI)

Additional Partnn columns were replaced by corresponding IDs to reduce redundancy or implement hierarchy.  They include:

* Court (CaseHeader CourtID references court record, giving short and long names)
* Opinion (OpinionBy, ConcurBy, and DissentBy are stored in Opinion table, one entry for each opinion type and judge)

The case type composite table contains a single value per LNI giving concatenated case types.  It has been verified that no case type is repeated.

I hope this helps.  I suggest limiting LN directed questions to those related to their source tables (Partnn, LNItoLNITreatments, Shepard’s Treatments, and Judges.  They should be able to explain each of the associated variables listed above.  Let me know if further explanation is needed for database tables and columns resulting from the import of LN’s data into our database.

Tom

**From:** Kevin Quinn <[kmq@umich.edu](mailto:kmq@umich.edu)>   
**Sent:** Saturday, April 27, 2019 8:01 AM  
**To:** Tom Balmat <[thomas.balmat@duke.edu](mailto:thomas.balmat@duke.edu)>; Jane Bahnson <[bahnson@law.duke.edu](mailto:bahnson@law.duke.edu)>; Sean Chen <[schen@law.duke.edu](mailto:schen@law.duke.edu)>; Wickliffe Shreve, J.D. <[wickliffe.shreve@law.duke.edu](mailto:wickliffe.shreve@law.duke.edu)>; Mark Delong <[mark.delong@duke.edu](mailto:mark.delong@duke.edu)>; Stuart M. Benjamin <[benjamin@law.duke.edu](mailto:benjamin@law.duke.edu)>  
**Subject:** RE: Questions for everyone now that I have now gone over the first part of the sample

One additional thing that I think we should ask LN about is whether they have additional descriptive information about the variables in the database. Right now we just have variable names and how the variable is stored (integer, character, etc.). Many of the names appear to be self-explanatory (e.g. DecisionDate), however some aren’t totally obvious— or at least it’s not obvious what the differences are between variables that appear similar (what’s the difference between the CaseType variables in the CaseType and CaseTypeComposite tables?). We can infer what most of these variables represent by looking at the underlying values, but that doesn’t allow us to figure out whether the contents represent what was intended.

If they could minimally tell us how each of the variables was constructed (what raw information is each variable based on and how was it constructed from that raw information) that would help a lot.

Best,

KQ