A DBPv5 Database Concept

From: Gary Griswold, gary@shortsands.com, 513-508-6127

January 15, 2021

Rev: February 15, 2021

As a result of writing the DBP v4 load, I have an understanding of the current DBP database, and some understanding of LPTS. It is generally understood that the current design of DBP is very inconsistent with the design of LPTS, and was possibly a design based on other work done at DBS. There is wide agreement that the design of DBP should become highly consistent with LPTS, or should share the same design or even share the same physical database. This document is written suggesting that they could become two databases on the same server. Organized this way the LPTS and DBP databases could be managed independently, but used together.

In proposing this design I have not considered the full requirements of LPTS, which I do not understand, but have simply studied the parts of LPTS that are extracted into DBP. Also, I have no insight into what new requirements might be desirable for future releases of the Bible.IS app. Others will need to add these details if they find this a useful starting point.

LPTS.Stock Table - This table is based upon the current LPTS stock number record, it ties together multiple DamIds that have a common copyright and permission data. Most of the data columns mentioned exist in the current LPTS. Those that are new have a description following them. Note that giving DBP access to this data directly will eliminate significant bloat in DBP, which repeats much of this information for each fileset. The table name Stock might not be a good name.

Reg_StockNumber - PRIMARY KEY
ISO
VersionCode - Usually the last three characters of the stock number, but not always. So it should be an explicit field.
LangName
Version
LicensorOrgId - A unique id in the Organization Table
CoLicensorOrgId - A unique id in the Organization Table

copyrightTextOrgId - A unique id in the Organization Table
copyrightAudioOrgId - A unique id in the Organization Table
copyrightVideoOrgId - A unique id in the Organization Table
Copyrightc
Copyrightp
Copyright_Video
Volumne_Name

The above lists all of the LPTS fields in this table that are required by DBP, but there are other LPTS fields that should be included for LPTS purposes: AltName, Country, CountryAdditional, DBL_Load_Notes, DBL_Load_Status, EthName, HeartName, ND_DBL_Load_Notes, NTOrder, Numerals, OTOrder, Portion, PostedToServer, ND_StockNumber, NTAudioDamLoad, NDRecordingStatus, Region, Reg_RecordingStatus, Selection, Streaming, Version, and some of those fields.

LPTS.Organization Table - This table does not currently exist in LPTS, but moving it from DBP to LPTS will help to have the organizations accurately identified.

OrganizationId - PRIMARY KEY
Organization Name

There are other organization columns that might be necessary.

LPTS.AccessGroups Table - This table is nearly identical to the current LPTS.access_group_filesets table, except that it links to the LPTS.Stock table, and not the bible_filesets table. Each Bible has various permissions set, or not set. Notice, that this eliminates extensive duplication since the current DBP database repeats this data for each fileset.

Reg_StockNumber FOREIGN KEY Stock (Reg_StockNumber)
AccessGroupId FOREIGN KET AccessGroupTypes (accessId)
PRIMARY KEY (Reg_StockNumber, AccessGroupId)

LPTS.AcessGroupTypes Table - This table is identical to the current DBP.access_groups table.

id - PRIMARY KEY
name
description

Permission Exception Notes: The above two tables LPTS.AccessGroupTypes and LPTS.AccessGroups handles many of the permission types, such as: APIDevAudio, APIDevText, APIDevVideo, DBPAudio, DBPMobile, DBPText, DBPTextOT, DBPWebHub, Download, HubText, MobileText, WebHubVideo. But the above solution does not handle more complex problems, such as the following: CreativeCommonsAudio, CreativeCommonsAudioWavier, CreativeCommonsText, FairUseLimit, FairUseLimitValue, itunesPodcast, Restrictions. More study is required to use these in DBP.

LPTS.StockDialect Table - Each Stock record can have filesets that are for different dialects of the single ISO code. These variants are described in LPTS by the index 1, 2, 3. There are other fields that can vary according to this index.. This 1, 2, or 3 can be found at the end of damId field name or in other field names. Notice, that the current bibleId is included here as a data column, and is not part of the key. Also note that this formalizes a crude and informal linkage that is in LPTS and makes it part of a design that can be enforced by foreign keys.

> Reg_StockNumber FOREIGN KEY Stock Table
> dialectIndex i.e. 1, 2, or 3
> PRIMARY KEY (Reg_StockNumber, dialectIndex)
> DialectName - This planned LPTS field is the LangName for a specific dialect
> Orthography (_x0031_Orthography, _x0032_Orthography, _x0033_Orthography)
> BibleId - called DBP_Equivalent (1, 2, or 3) in LPTS It is required to access buckets.

Other fields that belong in this table are as follows: DBPDate, DBPFont, ElectronicPublisher, ElectronicPublisherWebsite, HubLink, USX_Date,

Note about bibleId: Notice that bibleId is not the key of any table in this design. Making it a key in the current design is the source of problems, but it is included here as a data field, because it is needed to access filesets.

ISSUE: The current dbp-etl uses the 1,2,3 index solely to tie the correct orthography to each fileset, and yet Taylor has informed me that the primary purpose of the 1,2,3 is to identify different dialects, even though the current LPTS is lacking a DialectName or DialectCode field to associate a name with the 1,2,3. Trying to do both things in one table might be a problem, because an audio translation could be in multiple dialects. And a text Bible could be presented in multiple scripts, and it is also true that a Bible might be in multiple scripts and multiple dialects.

LPTS.DamId Table - This table has one record for each damId in the LPTS database. Each record in this table points back to a parent Stock table, record. And the StockDialect table points at many of these. This table does not describe bible_filesets, which are defined in a different table, but is only concerned with 10 digit DamId's as defined by LPTS.

DamId PRIMARY KEY
Media: text, audio, or video - derived from DamId ET, DA, DV
Drama: Reg or ND - derived from DamId 1,2
Scope: NT, OT, P, S - derived from DamId
dialectIndex: 1, 2, 3 - This is derived from the field name of the DamId
Status: Live, etc.
StockNumber FOREIGN KEY Bibles (Reg_StockNumber)
FOREIGN KEY (StockNumber, dialectIndex) StockDialect (StockNumber, dialectIndex)

Note about DamId's: The information about media type, scope, are found in both the DamId itself and the field name that contains it. The dialectIndex is only found in the field name. In the case, below the dialectIndex is 2.
<Reg_NTAudioDamID2>ACRWB1N2DA</Reg_NTAudioDamID2>
<Reg_NTAudioDamIDStatus2>Live</Reg_NTAudioDamIDStatus2>
In the new design, it would best if these field names were very difficult to change, since their revision in the past has been a source of errors in DBP.

DBP.BibleFilesets Table - This table contains one record for each actual fileset as stored in the s3 bucket or the DBP database.

PRIMARY KEY filesetId - PRIMARY KEY
DamId FOREIGN KEY LPTS.DamId (DamId)
Container - mp3, aac, opus, webm, etc
Bitrate - bitrate for audio or video and blank for text, or could be hls or dash
UNIQUE KEY (DamId, Container, Bitrate)
Bucket - same as asset_id
FilenamePrefix - this is the prefix used to store the related data in S3, e.g. text/ENGESV/ENGESVN2DA/. Having this explicitly stored in this table will provide flexibility in how the storage of data is revised in the future.

DBP.Bibles View - The Bibles table is removed in this design, which might seem like a great loss, but it is necessary, because the data the bibles table contains cannot be created with great accuracy. The way that bible_ids are defined, they can sometimes include multiple stock records, and those records might contain different copyright data. What should replace the bibles table is simply a view that joins the tables LPTS.Stock, LPTS.StockDialect, LPTS.DamId, and

DBP.BibleFilesets. This will be a very useful view for doing queries that tie together related filesets in a variety of ways.

ISO - One can query the view to get all of the stock records for a specific ISO.

VersionCode - This has now been defined as a field, but it is usually the last 3 characters.

(ISO, VersionCode) - A query of these two fields alone will get the same FilesetsId as the query of a bible_id 95% of the time.

(ISO, VersionCode, Orthography) A query of these three fields will retrieve exactly what the current bible_id is intended to do.

DBP.BibleFiles Table - This table would be very similar to the current bible_files table.

Id PRIMARY KEY - auto generated

FilesetId FOREIGN KEY BibleFilesets (filesetId)
book_id
chapter_start
chapter_end
verse_start
verse_end
file_name
file_size
duration

ISSUE: The only change with the current BibleFiles table is the elimination of hashId, replaced with filesetId. Doing this eliminates the ability to store one fileset in multiple buckets. Is the a problem?

DBP.BibleVerses Table - This table would be very similar to the current bible_verses table, except that it has a link back to BibleFileset using the filesetId, not the hashId.

Id PRIMARY KEY - auto generated

filesetId FOREIGN KEY BibleFileset (filesetId)
book_id
chapter
verse_start
verse_end
verse_text

DBP.BibleBooks Table - This should be identical to the existing table.

Note on deleted tables: In this design, many existing tables, which are important have been eliminated, because the data they contain can be accessed directly from LPTS. Deleted tables includes: bibles, bible_tags, bible_fileset_tags, bible_fileset_copyrights, bible_fileset_copyright_organizations, language_translations.

DBP.BibleFileStreamBandwidths Table - This table should be included as is.

DBP.BibleFileStreamTS Table - This table could be included as is, but I think other solutions should be investigated, including storing the m3u8 file contents in an S3 bucket, or in the bandwidth record. AWS does provide a means to generate signed URLS in C++. That code is used in the IOS version of my own App. The name of the AWS C++ files are: AWSS3PreSignedURL.h, AWSS3PreSignedURL.m