Appendix A: An introduction to Lucerna's database structure

Lucerna is structured as a relational database. In this form, information is stored as a series of interrelated tables. Each text, event, organization, slide, and slide set appears as a row in one of these tables. Each row is assigned a unique string of seven numbers, known as a primary key. In the documentation surrounding *A Million Pictures*, the primary key is often referred to as the Lucerna ID. Expressing the relationships between rows as a number (instead of using a prose description or a proper name) mitigates issues caused by spelling mistakes and alternate forms of a name. Information stored in separate tables can be linked by adding the primary key from the first table to the second. Since a row cannot have two primary keys, the second key becomes a foreign key. Finding relationships between events, locations, host organizations, and texts becomes a matter of locating rows in multiple tables with shared keys. Storing information in this manner makes it possible to tie an address or an organization to multiple events. For example, the Lucerna ID, or primary key, for the Royal Albert Memorial Museum (1507743) appears once as a primary key in the table "tbladdress," but its listed 48 in the table "tbleventaddress" because there were 48 of lantern shows given there in this corpus. To make matters more complex, the RAMM's primary key does not appear in the table that ties texts to events. So, in order to search for the address of an event mentioned by an account of a magic lantern show, one would need to information from three different tables-- tblkeywordlink, tbleventaddress, and tbladdress.

Structured Query Language, or SQL, provides a means to filter information from one table with criteria from another in ways that are not possible through the search interface on the website. Figure 1 maps the connecting logic of the tables used in this study. The diagram and the following code include the column names which contain primary and foreign keys. Due to the complexity of Lucerna, I used three separate queries to find the information that I needed: one to find quotes from texts that describe magic lantern shows (tblkeywordlink + tbltext), one to find information about the date and time of the event (tblevent + tbleventaddress + tbladdress), and one to find information about organisations who hosted events (tblorganisationevent + tblorganisation). The results could then be exported as a comma separated value file, which can easily be opened by any spreadsheet software like Google sheets or Microsoft Excel. These three queries gave me three tables that I would then review, edit, topic model, and map. SQL not only identifies potential items of interest in Lucerna, but it can also be used to link layers in maps, particularly through proprietary tools like ArcGIS. Carto, the mapping tool used in this study, has limited SQL functionality. Future iterations of this study might consider mirroring Lucerna's relational database structure more closely by using a more powerful mapping tool.

https://a-million-pictures.wp.hum.uu.nl/wp-content/uploads/sites/210/2018/05/Lucerna-Manual-02-Cataloguing-slide-sets.pdf

Find all quotations from primary sources that describe an event

```
SELECT * FROM tblkeywordlink
WHERE keywordtable="tblevent" AND itemtable="tbltext"
AND LENGTH(keywordlinkquote) > 0
ORDER BY keywordlinkquote
```

Expressing this query, line by line, in prose form would look something like this:

```
Find all information from the table named ‘tblkeywordlink.’
Look for entries that include ‘tblevent’ and ‘tbltext’; ignore entries that link slides and
    people.
Give me entries that have quotations from primary sources; the column that contains
    quotes will not be empty.
Sort the results from shortest to longest quotation.
```

Find information about the event and its location for all events that have text

```
SELECT * FROM tblevent
INNER JOIN tbleventaddress ON tblevent.eventid=tbleventaddress.eventid
INNER JOIN tbladdress ON tbleventaddress.addressid=tbladdress.addressid
WHERE tbleventaddress.eventid IN(SELECT keywordid FROM tblkeywordlink
WHERE keywordtable="tblevent" AND itemtable="tbltext"
AND LENGTH(keywordlinkquote) > 0)
```

This is perhaps the most complex query because it involves four tables. It is best understood as a two-step process. The first part of the query unites information about events in 'tblevent' and their location in 'tbleventaddress' by looking for the links expressed in a table called 'tbleventaddres'. The query matches primary keys from columns 'eventid' in 'tblevent' and 'addressid' in 'tbladdress' to columns that include the same number in 'tbleventaddress' (also named 'eventid' and 'addressid'.) Locating primary and foreign keys, or matching Lucerna IDs, enables the query to express information from both of these tables in a single table.
Without the bit in parenthesis, this query would return information for all events that have addresses. For the purposes of this study, I am only interested in the events that also have a quote from a primary source. The part in parenthesis is a slightly modified version of the query that I used to find information about events with texts. Instead of returning all information about texts from 'tblkeywordlink', the query only returns the column 'keywordid' which contains the LucernaID for the event. The query then matches Lucerna IDs with those listed in the column 'eventid' in the table 'tbleventaddress.'

SQL queries can also help identify entries with missing information. My query about events with texts returned 2,262 results, but there were only 2,212 addresses. This meant that 50 events have no location or that 50 took place in the same location. The query below found 28 events that had no location (IS NULL); these were mostly reports of touring lectures. So 22 locations hosted more than one lantern show.

```
SELECT * FROM tblevent
LEFT JOIN tbleventaddress ON tblevent.eventid=tbleventaddress.eventid
WHERE tbleventaddress.eventid IS NULL
AND tblevent.eventid IN(SELECT keywordid FROM tblkeywordlink WHERE
keywordtable="tblevent" AND itemtable="tbltext"
AND LENGTH(keywordlinkquote) > 0)
```

Find all information about organisations connected with events that have text

```
SELECT tblorganisation.* FROM tblorganisation
INNER JOIN tblorganisationevent ON
tblorganisation.organisationid=tblorganisationevent.organisationid
WHERE tblorganisationevent.eventid IN(SELECT keywordid FROM tblkeywordlink
WHERE keywordtable="tblevent" AND itemtable="tbltext"
AND LENGTH(keywordlinkquote) > 0)
```

Like the query to find information about the location of lantern shows, this query uses a two-part structure to filter search results from four tables. The part in parenthesis is the same, but the query looks for connections between organisation and event through 'tblorganisationevent'.

Additional queries

After completing these searches, I discovered that I had accidentally excluded bibliographic information about the texts. The query below finds all information about the text from 'tbltext' for all texts that describe events. I then reunited this information with texts using Microsoft Excel's VLOOKUP formula.

```
SELECT * FROM tbltext
WHERE tbltext.textid IN(SELECT itemid FROM tblkeywordlink WHERE
keywordtable="tblevent" AND itemtable="tbltext"
AND LENGTH(keywordlinkquote) > 0)
```