Talk on the SPLDB

“I do research on New Interfaces for Musical Expression, so I study audiovisual perception, action and feedback in the context of interactive, musical systems. I work on large-scale interactive, distributed audiovisual systems like the AlloSphere. Along the way, I explain things to myself, my colaborators and my students. Part of that is making data graphics.” — Karl Yerkes, 2016-01-28

Here’s a summary of what I want to leave you with today…

1. Read Edward Tufte on Data Visualization.
2. To do good data visualization you have to establish a deep understanding of the data.
3. Start with data analysis: make 2d graphs of the data to learn the truth about it.
4. Write questions in English and then try to write the answer as a MySQL query. Repeat.
5. You can and should add tables to the SPLDB. Give your queries to George and Rodger, so they can make new tables out of them.
6. Do frequency analysis (FFT) of checkouts as a time series.
7. Use `x_splOrgWebScraping`. It contains “external keys” (e.g. ISBN) that can be used to tie SPL data to other datasets.
8. Avoid REGEX() at all cost! Use the keyword table instead. It’ll be 2 or 3 or orders of magnitude faster.

What I left out of this document…

- In my talk I went over all the `x_*` tables in the `spl3` database. A treatment of those tables is available upon request.
- In my talk I designed and executed several queries on the fly using `JOIN`. Only a couple of those are found in this document. More are available upon request.

Edward Tufte

My aesthetic and conceptual framework for data visualization starts and ends with Edward Tufte. Read The Visual Display of Quantitative Information by Edward Tufte for an introduction to the feild. Tufte introduces a couple important ideas that I’ll quote here:

Data-ink

“Maximize the Data-ink ratio” (i.e., Show the data)

Data-ink ratio = data-ink / total ink used to print the graphic

Data-ink ratio = proportion of the graphic's ink devoted to the non-redundant display of data-information

Data-ink ratio = 1.0 - the proportion of a graphic that can be erased without loss of information

Graphical Integrity

1. The representation of numbers, as physically measured on the surface of the graph itself, should be directly proportional to the numerical quantities represented
2. Clear, detailed and thorough labeling should be used to defeat graphical distortion and ambiguity. Write out explanations of the data on the graph itself. Label important events in the data.
3. Show data variation, not design variation.
4. In time-series displays of money, deflated and standardized units of monetary measurement are nearly always better than nominal units.
5. The number of information carrying (variable) dimensions depicted should not exceed the number of dimensions in the data. Graphics must not quote data out of context.

These ideas guide my thoughts on the matter: Design only in service to the data—Design should never take center stage in Data Visualization.

The Datapath: Provenance and Process

The data that you use in your projects goes through many stages. In order to tell the truth with the data, you must understand its provenance.

- Humans interact with items from the library

The library makes a record in a database called a "Library Information System" (or LIS) each time a person takes an item (check *out*). Each time a person returns and item, another separate record is made in the LIS. (You will see many check *in* events map to the same check *out* event because the SLP LIS uses check *in* events to represent library-internal processes like holds and transfers.)

The SPL LIS keeps track of the current state of the library (i.e. which items are present or absent and who has them), but it is designed to forget (see privacy notice and confidentiality of borrower records) who and when after some period.

- SPL's Library Information System

We do not have direct access to the SPL LIS. Each hour, a special computer at the SPL runs two SQL queries to creates two XML files. These queries record data on all check *out* and *in* events that occured in the last hour.

These files are a "snapshot" of the current state of the SPL LIS for a given hour—In this way, we capture

changes in the state of the SPL LIS. This XML is saved in a place only we have access to.

- XML on SPL server

Hourly, we copy the XML to our server here at MAT.

- XML on MAT server

The XML is about 200GB which is too much to handle. We have to translate the data into something smaller and easier to work with. So, annually, we process the XML to build a MySQL database. We're working on rebuilding daily or hourly, but for now it's annual.

- MySQL database for MAT259

Building the MySQL SPL database was (and is) not trivial. A lot of work went into understanding the data and design the structure of the database. Part of this process is database normalization.

MAT 259 students like you form questions about the data and write answers in the form of MySQL queries. Then you execute the queries and use the result in your visualizations.

- MAT259 visualization projects

There are several ways to classify MAT 259 visualizations. You can think 1d, 2d, and 3d or interactive versus "artful". The way I like to split them is *internal* versus *external* to the SPL itself. I tried to make all my visualizations about the library itself—What are it's internal processes? Where is the noise in the data?

`itemNumber` and `bibNumber`

`itemNumber` and `bibNumber` are auto-incrementing database keys in the SPL LIS. Whenever an item is added to the library, the item is asigned a new `itemNumber` by adding 1 to the last, largest known `itemNumber` (same with `bibNumber` for brand new titles). We can analyse these keys to get information about that system. In particular, we can estimate the rate of acquisition of new materials by determining the slope of the plot of check out time versus `itemNumber` (or `bibNumber`). We can estimate when big events happened by investigating the gaps in the data on this plot.

Question: *What proportion of items have never been checked out? (i.e., Which are the loneliest items?)*

Because `itemNumber` is an auto-incrementing key at the SPL LIS, we only see certain keys (the ones that get checked out) and not others (the ones that never get checked out) in our database. We can estimate the percentage of "lonely" items like this:

```
select (1.0 - (select count(*) from itemType) / (select max(itemNumber) from itemType
)) * 100 as '% lonely'
```

About 28% of items are lonely—We haven't seen them checked out in 10 years of data. But, there's a problem with our calculation. It assumes that `itemNumber` was never incremented by more than 1. Sometimes database administrators increment by large numbers when databases are rebuilt or moved. A more in depth analysis of `itemNumber` is required and that analysis is left as an exercise for the reader.

Use `JOIN`

`JOIN` is a fundamental operation for SQL. It's how you merge tables together. While there are many different kinds of join, 99% of the time we mean `INNER JOIN` which merges tables together on a common column. Here's an example that joins two tables on the `itemType` column:

```
SELECT
        formatDetail,count(*) as itemCount
FROM
        itemType INNER JOIN itemTypeDecode
WHERE
        itemType.itemType = itemTypeDecode.itemType
GROUP BY
        formatDetail
ORDER BY
        itemCount DESC
```

The query above joins two tables: *itemType* and *itemTypeDecode*. `INNER JOIN` is so common that there is a shorthand syntax for it. Instead of typing `INNER JOIN`, you can just use a `,` (comma). So, here's the same query:

```
SELECT
        formatDetail,count(*) as itemCount
FROM
        itemType,itemTypeDecode
WHERE
        itemType.itemType = itemTypeDecode.itemType
GROUP BY
        formatDetail
ORDER BY
        itemCount DESC
```

Here's a more complicated and userful example of join:

```
SELECT
        formatDetail,
        SUM(CASE WHEN YEAR(checkOut) = 2006 THEN 1 ELSE 0 END) AS '2006',
        SUM(CASE WHEN YEAR(checkOut) = 2007 THEN 1 ELSE 0 END) AS '2007',
        SUM(CASE WHEN YEAR(checkOut) = 2008 THEN 1 ELSE 0 END) AS '2008',
        SUM(CASE WHEN YEAR(checkOut) = 2009 THEN 1 ELSE 0 END) AS '2009',
        SUM(CASE WHEN YEAR(checkOut) = 2010 THEN 1 ELSE 0 END) AS '2010',
        SUM(CASE WHEN YEAR(checkOut) = 2011 THEN 1 ELSE 0 END) AS '2011',
        SUM(CASE WHEN YEAR(checkOut) = 2012 THEN 1 ELSE 0 END) AS '2012',
        SUM(CASE WHEN YEAR(checkOut) = 2013 THEN 1 ELSE 0 END) AS '2013',
        SUM(CASE WHEN YEAR(checkOut) = 2014 THEN 1 ELSE 0 END) AS '2014',
        SUM(CASE WHEN YEAR(checkOut) = 2015 THEN 1 ELSE 0 END) AS '2015'
FROM
        itemType,itemTypeDecode,transactions
WHERE
        itemType.itemNumber = transactions.itemNumber
AND
        itemType.itemType = itemTypeDecode.itemType
GROUP BY
        formatDetail
ORDER BY
        SUM(CASE WHEN (YEAR(checkOut) >= 2006 OR YEAR(checkOut) <= 2015) THEN 1 ELSE
0 END) DESC
```

This query joins three tables (i.e., *itemType*, *itemTypeDecode*, and *transactions*) on two columns (i.e., *itemNumber* and *itemType*).

Don't use `REGEX` —Use the `keyword` table instead

```
SELECT
        title
FROM
        title
WHERE
        title REGEXP 'bowie'
```

The query above uses `REGEX` to search through the title table. Using `REGEX` is slow. The query above took about 2 seconds to complete (just to search titles, not subjects or call numbers). In most cases we can do better. The `keyword` table contains all the words from the `subject`, `title`, and `callNumber` tables so keyword lookups can be made extremely fast. Here's a query that uses the `keyword` table to search for titles that match 'bowie':

```
SELECT
        title
FROM
        keyword,title
WHERE
        keyword.bibNumber = title.bibNumber
AND
        keyword = 'bowie'
```

The query above completes in 18 milliseconds and does the equivalent of searching the `subject`, `title`, and `callNumber` tables for the string 'bowie'. Note that using `JOIN` is required when you use the `keyword` table.

Analyse your queries with `EXPLAIN`

Built into MySQL is a way of analysing MySQL queries. Put the keyword `EXPLAIN` before any valid query to get an analysis of the query. With some reading and practice, you can tell whether a given query will take forever or run quickly. Read MySQL EXPLAIN Explained and/or Explaination of EXPLAIN. There is also plain old documentation.

Consider the `EXPLAIN` output for the two queries that we just looked at…

```
EXPLAIN SELECT
        title
FROM
        title
WHERE
        title REGEXP 'bowie'

+----+-------------+-------+-------+---------------+-------+---------+------+--------
+--------------------------+
| id | select_type | table | type  | possible_keys | key   | key_len | ref  | rows
| Extra                    |
+----+-------------+-------+-------+---------------+-------+---------+------+--------
+--------------------------+
|  1 | SIMPLE      | title | index | NULL          | title | 768     | NULL | 880027
| Using where; Using index |
+----+-------------+-------+-------+---------------+-------+---------+------+--------
+--------------------------+

EXPLAIN SELECT
        title
FROM
        keyword,title
WHERE
        keyword.bibNumber = title.bibNumber
AND
        keyword = 'bowie'

+----+-------------+---------+------+-----------------------------------+----------
-+---------+----------------------+------+-------------+
| id | select_type | table   | type | possible_keys                     | key
 | key_len | ref                  | rows | Extra       |
+----+-------------+---------+------+-----------------------------------+----------
-+---------+----------------------+------+-------------+
|  1 | SIMPLE      | keyword | ref  | bibNumber_keyword,bibNumber,keyword | keyword
 | 767     | const                |   32 | Using where |
|  1 | SIMPLE      | title   | ref  | bibNumber_title,bibNumber           | bibNumber
 | 4       | spl.keyword.bibNumber |    1 | Using where |
+----+-------------+---------+------+-----------------------------------+----------
-+---------+----------------------+------+-------------+
```

One important value to look at is the `rows` column. The more rows that must be searched, the slower the query will be.

Frequency Pattern Mining

I used Frequent Pattern Mining to look for interesting patterns within the SPL. This approach associates items when they are checked out in the same minute and looks for recurring associations. This lead to the positive, but mundane insight that *items that are part of a set tend to be checked out together*. So, all the Lemony Snicket books get checked out together. If someone checks out a Star Wars DVD, it's highly probable that they'll checkout IV, V, *and* VI together.

Looking forward…

To maximize the value of the SPL project for students, creators, and the Seattle Public Library community, the best platform for MAT 259 projects is the World Wide Web. Many projects use Processing which is a solid tool but it's www features are pathetic compared to newer, more www-savvy tools.

p5.js, d3.js, node.js, and NoSQL are the way forward. I envision a "stack" for MAT 259 built around these tools.