Storing and Accessing Lifelog Data with MySQL & NoSQL (MongoDB) Databases

LUZ LOPEZ
DIVISION OF SCIENCE AND MATHEMATICS
UNIVERSITY OF MINNESOTA, MORRIS
NOVEMBER 18TH, 2017
Outline

1. INTRODUCTION
2. LIFELOG MASHUP MySQL WITH RDB IMPLEMENTATION
3. LIFELOG MASHUP NoSQL WITH MongoDB IMPLEMENTATION
4. CONCLUSION
1. INTRODUCTION
   i. What is lifelogging?
   ii. Lifelog Mashup
   iii. Lifelog Common Data Model (LLCDM)
   iv. Lifelog API (LLAPI)
   v. Previous Work Limitations
2. LIFELOG MASHUP MySQL WITH RDB IMPLEMENTATION
3. LIFELOG MASHUP NoSQL IMPLEMENTATION WITH MongoDB
4. CONCLUSION
What is Lifelogging? /Importance /Lifelog Mashup

**Lifelogging:**
Also known as “life catching”
A social act to record and share human life events in an open and public form [1,2]

**Lifelog Mashup:**
Integrating scattered lifelogs would implement more sophisticated and value-added services, than using them separately [1]

- **Personal**
  - Personal health achievements
  - Productivity
  - Self-enhancement

- **Public**
  - Memories
  - Photos
  - Connections
Lifelog Common Data Model

**LLCDM:**

*Lifelog common data model* prescribes a generic data schema for lifelog records, which does not rely on any specific lifelog service.

Designed with standard attributes of what, who(m), why, where, how. [2]
Importance of LLCDM

Data record of Twitter

```json
{
  "created_on": "Friday Jul 05 2013 03:45:35+000",
  "id": 3353155350876845002,
  "text": "Working outside today",
  "source": "http://twitter.com",
  "geo": {"type": "Point", "coordinates": [32.8753586, 135.874874]}
}
```

Data record of SensorLoggingService

```json
{
  "Time": "12:46:57",
  "User": "koupe",
  "Weather": "Sunny",
  "TempF": 76.73,
  "Brightness": 310,
  "Temperature": 26.6,
  "Id": 23654,
  "Date": "2013-07-05"
}
```
Importance of LLCDM

Data record of Twitter

```json
{
    "created_on": "Friday Jul 05 2013 03:45:35+000",
    "id": 3353155350876845002,
    "text": "Working outside today",
    "source": "<a href="http://twitter.com/"/>
}
```

Data record of SensorLoggingService

```json
{
    "Time": "12:46:57",
    "User": "koupe",
    "Weather": "Sunny",
    "TempF": 76.73,
    "Brightness": 310,
    "Temperature": 26.6,
    "Id": 23654,
    "Date": "2013-07-05"
}
```
Lifelog Mashup API

**LLAPI:**

*Lifelog mashup API* is for searching and retrieving lifelog data conforming to the LLCDM [1] by matching specific given queries.
Lifelog Mashup API

LLAPI:

- Lifelog mashup API is for searching and retrieving lifelog data conforming to the LLCDM [1] by matching specific given queries.

- Using `getLifeLog()` heterogeneous lifelogs can be accessed uniformly without proprietary knowledge of lifelog services.
Lifelog Mashup API

LLAPI:

- *Lifelog mashup API* is for searching and retrieving lifelog data conforming to the LLCDM [1] by matching specific given queries.

- Using `getLifeLog()` heterogeneous lifelogs can be accessed uniformly without proprietary knowledge of lifelog services.

Using `getLifeLog` example wrapping an *SQL* statement [2]:

```sql
getLifeLog(s_date, e_date, s_time, e_time, user, party, object, location, application, device, select)
```
Lifelog Mashup API

LLAPI:

- Lifelog mashup API is for searching and retrieving lifelog data conforming to the LLCDM [1] by matching specific given queries.

- Using `getLifeLog()` heterogeneous lifelogs can be accessed uniformly without proprietary knowledge of lifelog services.

Using `getLifeLog` example wrapping an SQL statement [2]:

```sql
getLifeLog(s_date, e_date, s_time, e_time, user, party, object, location, application, device, select)
```

Parameters:

- `s_date, e_date` : Query of <date> (start, end)
- `s_time, e_time` : Query of <time> (start, end)
- `user, party, object` : Query of <user, party, object>
- `location` : Query of <location>
- `application` : Query of <application>
- `service` : Query of <device>
- `select` : Query of <select>
Lifelog Mashup Platform

Devices and Web services
Introduce the LLCDM
Lifelog Mashup Platform

Retrieve data from services API
Lifelog Mashup Platform

LLAPI requests lifelog data from LLCDM using method put/getLifelog()
LLAPI retrieves lifelog data from LLCDM using method

Lifelog API (LLAPI)  
(put/getLifelog())
Mashup applications return user-friendly visuals
Lifelog Mashup Experiments

Experiment 1: First they proposed a lifelog mashup LLCDM and LLAPI to access standardized data

- Poor portability
- Low performance
Lifelog Mashup Experiments

**Experiment 1**: First they proposed a lifelog mashup LLCDM and LLAPI to access standardized data

- Poor portability
- Low performance

**Experiment 2**: Then re-engineered it with relational MySQL and Web services.

- Evaluated
Lifelog Mashup Experiments

**Experiment 1**: First they proposed a lifelog mashup LLCDM and LLAPI to access standardized data
- Poor portability
- Low performance

**Experiment 2**: Then re-engineered it with relational MySQL and Web services.
- Evaluated

**Experiment 3**: Once again re-engineered, this time with NoSQL
- Evaluated
Limitations with XML Prototype

- Low performance
  - Had to convert data into raw XML files then store it

- Poor Portability
  - Prototype was written in Perl language, no choice for developers to use other languages to build mashup applications
How to improve limitations

- **Low performance**
  - Had to convert data into raw XML files then store it

  Put data in *relational database (RDB)* instead of having data as raw XML files.

  Faster data search and access.

- **Poor Portability**
  - Prototype was written in *Perl* language, no choice for developers to use other languages to build mashup applications

  Programmers create and implement two versions of the mashup application to evaluate the feasibility of new implementation
Outline

1. INTRODUCTION
2. LIFEOG MASHUP MySQL WITH RDB IMPLEMENTATION
   i. Process
   ii. Evaluation
   iii. Limitations
3. LIFEOG MASHUP NoSQL WITH MongoDB IMPLEMENTATION
4. CONCLUSION
Process

1. Importing lifelog data to LLCDM repository
2. Re-engineering LLAPI
3. Evaluate Performance
   ▶ SOAP and REST Web-service Protocols
   ▶ Mashup Example TabelaLog

**Goal:** To show the practical feasibility of the proposed LLAPI.
1. Importing lifelog data to LLCDM Repository

Steps to import data from heterogenous lifelog services to the LLCDM repository:
1. **Importing lifelog data to LLCDM Repository**

Steps to import data from heterogenous lifelog services to the LLCDM repository:

1. Obtain original data
   1. Obtain the original data from service and store data in XML
1. **Importing lifelog data to LLCDM Repository**

Steps to import data from heterogenous lifelog services to the LLCDM repository:

1. **Obtain original data**
   1. Obtain the original data from service and store data in XML

2. **Transform data to LLCDM**
   1. Raw data to the LLCDM format
1. **Importing lifelog data to LLCDM Repository**

Steps to import data from heterogeneous lifelog services to the LLCDM repository:

1. Obtain original data
   1. Obtain the original data from service and store data in XML

2. Transform data to LLCDM
   1. Raw data to the LLCDM format

3. Insert data into database
   1. Insert the XML into the database
   2. Parses the converted XML data
   3. Extracts the attributes and inserts the values to appropriate tables.
Comparison of execution times

<table>
<thead>
<tr>
<th></th>
<th>Query 1</th>
<th>Query 2</th>
<th>Query 3</th>
<th>Query 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAP (sec)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REST (sec)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OLD (sec)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># OF ITEMS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA SIZE (kB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Comparison of execution times**

<table>
<thead>
<tr>
<th>November 15 - November 16</th>
<th>Query 2</th>
<th>Query 3</th>
<th>Query 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAP (sec)</td>
<td>0.131</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REST (sec)</td>
<td>0.015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OLD (sec)</td>
<td>4.238</td>
<td></td>
<td></td>
</tr>
<tr>
<td># OF ITEMS</td>
<td>36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA SIZE (kB)</td>
<td>118</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Comparison of execution times

<table>
<thead>
<tr>
<th></th>
<th>November 15-November 16</th>
<th>September 1-September 30</th>
<th>Query 3</th>
<th>Query 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAP (sec)</td>
<td>0.131</td>
<td>1.006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REST (sec)</td>
<td>0.015</td>
<td>0.100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OLD (sec)</td>
<td>4.238</td>
<td>4.028</td>
<td></td>
<td></td>
</tr>
<tr>
<td># OF ITEMS</td>
<td>36</td>
<td>119</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA SIZE (kB)</td>
<td>118</td>
<td>381</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Comparison of execution times

<table>
<thead>
<tr>
<th></th>
<th>November 15-16</th>
<th>September 1-30</th>
<th>9:00:00-10:15:00 On any date</th>
<th>Query 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAP (sec)</td>
<td>0.131</td>
<td>1.006</td>
<td>0.281</td>
<td></td>
</tr>
<tr>
<td>REST (sec)</td>
<td>0.015</td>
<td>0.100</td>
<td>0.019</td>
<td></td>
</tr>
<tr>
<td>OLD (sec)</td>
<td>4.238</td>
<td>4.028</td>
<td>4.254</td>
<td></td>
</tr>
<tr>
<td># OF ITEMS</td>
<td>36</td>
<td>119</td>
<td>195</td>
<td></td>
</tr>
<tr>
<td>DATA SIZE (kB)</td>
<td>118</td>
<td>381</td>
<td>1,450</td>
<td></td>
</tr>
</tbody>
</table>
## Comparison of execution times

<table>
<thead>
<tr>
<th></th>
<th>November 15-November 16</th>
<th>September 1-September 30</th>
<th>9:00:00-10:15:00 On any date</th>
<th>User – “Shimojo”</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAP (sec)</td>
<td>0.131</td>
<td>1.006</td>
<td>0.281</td>
<td>0.422</td>
</tr>
<tr>
<td>REST (sec)</td>
<td>0.015</td>
<td>0.100</td>
<td>0.019</td>
<td>0.025</td>
</tr>
<tr>
<td>OLD (sec)</td>
<td>4.238</td>
<td>4.028</td>
<td>4.254</td>
<td>0.581</td>
</tr>
<tr>
<td># OF ITEMS</td>
<td>36</td>
<td>119</td>
<td>195</td>
<td>449</td>
</tr>
<tr>
<td>DATA SIZE (kB)</td>
<td>118</td>
<td>381</td>
<td>1,450</td>
<td>630</td>
</tr>
</tbody>
</table>
TabetaLog – FoodLogService + Flickr

Weight (lbs) vs. Day out of the year

- Blue dots: Shimojo
- Red dots: Tokunga
TabetaLog was an experimental evaluation lifelog mashup application.

Steps for creating the TabetaLog:

1. Obtain original lifelog records
   - Web-service API
2. Extract data items
   - Parsing records
3. Join data items
   - Joined records are stored in JSON format file
4. Create TabetaLog
   - Using ActionScript, visualize the JSON data
## Evaluation/Results

<table>
<thead>
<tr>
<th>Programmer</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order of Development</td>
<td>$P_{llapi}$</td>
<td>$P_{conv}$</td>
<td>$P_{conv}$</td>
<td>$P_{llapi}$</td>
<td>$P_{conv}$</td>
<td>$P_{llapi}$</td>
</tr>
<tr>
<td>Programming Language</td>
<td>Perl</td>
<td>Perl</td>
<td>Perl</td>
<td>Perl</td>
<td>Java</td>
<td>Java</td>
</tr>
<tr>
<td>Source lines of code</td>
<td>115</td>
<td>365</td>
<td>227</td>
<td>379</td>
<td>480</td>
<td>612</td>
</tr>
<tr>
<td>SLOC (w.out blank and comments)</td>
<td>71</td>
<td>223</td>
<td>103</td>
<td>188</td>
<td>351</td>
<td>426</td>
</tr>
<tr>
<td># of source-code classes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td># of source-code files</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Man-hour (man minute)</td>
<td>114</td>
<td>196</td>
<td>54</td>
<td>205</td>
<td>96</td>
<td>252</td>
</tr>
<tr>
<td># of weight records &lt;Shimojo&gt;</td>
<td>53</td>
<td>54</td>
<td>53</td>
<td>54</td>
<td>32</td>
<td>53</td>
</tr>
<tr>
<td># of weight records &lt;Togunaga&gt;</td>
<td>102</td>
<td>101</td>
<td>102</td>
<td>101</td>
<td>52</td>
<td>103</td>
</tr>
<tr>
<td># of picture records &lt;Shimojo&gt;</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td># of picture records &lt;Tokunaga&gt;</td>
<td>85</td>
<td>86</td>
<td>85</td>
<td>86</td>
<td>60</td>
<td>87</td>
</tr>
<tr>
<td>Programmer</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>Correct</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td>Order of Development</td>
<td>$P_{llapi}$</td>
<td>$P_{conv}$</td>
<td>$P_{llapi}$</td>
<td>$P_{conv}$</td>
<td>$P_{llapi}$</td>
<td>$P_{conv}$</td>
</tr>
<tr>
<td>Source lines of code</td>
<td>115</td>
<td>365</td>
<td>227</td>
<td>379</td>
<td>480</td>
<td>612</td>
</tr>
<tr>
<td>Programming Language</td>
<td>Perl</td>
<td>Perl</td>
<td>Perl</td>
<td>Perl</td>
<td>Java</td>
<td>Java</td>
</tr>
<tr>
<td>Man-hour (man minute)</td>
<td>114</td>
<td>196</td>
<td>54</td>
<td>205</td>
<td>96</td>
<td>252</td>
</tr>
</tbody>
</table>
3. Evaluating Performance

- Compared to previous prototype.
- 1,591 records of data were stored in MySQL database

Five subjects implement a program generating the TabetaLog JSON file. Subjects implement two versions of the program: one with the proposed LLAPI and one with the conventional LLAPI. [1]

The subjects were instructed to mashup the weight records and the picture records of user “Shimojo” and “Tokunaga” for one year (May 18th, 2010-May 17th, 2011) and to output the resulting JSON file.
Outline

1. INTRODUCTION
2. LIFELOG MASHUP MySQL WITH RDB IMPLEMENTATION
3. LIFELOG MASHUP NoSQL WITH MongoDB IMPLEMENTATION
   i. Limitations
   ii. Process
   iii. Evaluation
4. CONCLUSION
Limitations with MySQL Prototype

1. Could not specify application-specific attributes (stored in the <content> column) for data query [2]
2. Scalability
Limitations with MySQL Prototype

1. Could not specify application-specific attributes (stored in the <content> column) for data query [2]
Limitations with MySQL Prototype

1. Could not specify application-specific attributes (stored in the `<content>` column) for data query [2]

   - Data was stored in an unstructured plain text file, which SQL cannot interpreted.

   - Queries with application-specific attributes had to be managed by individual mashup applications. Causing large application overhead and expensive development cost [2].
Limitations with MySQL Prototype

1. Could not specify application-specific attributes (stored in the <content> column) for data query [2]
   - Data was stored in an unstructured plain text file, which SQL cannot interpreted.
   - Queries with application-specific attributes had to be managed by individual mashup applications. Causing large application overhead and expensive development cost [2].

2. Scalability
Limitations with MySQL Prototype

1. Could not specify application-specific attributes (stored in the `<content>` column) for data query [2]
   - Data was stored in an unstructured plain text file, which SQL cannot interpreted.
   - Queries with application-specific attributes had to be managed by individual mashup applications. Causing large application overhead and expensive development cost [2].

2. Scalability
   - As more lifelog services appear, the platform should be scalable enough to keep up with larger data.
Benefits of MongoDB

Resolves limitation 1
- Document-orientated storage
  - MongoDB BSON object represents dynamically-typed data in the `<content>` column
- Full index support
  - Useful for queries over the `<content>` column

Resolves limitation 2
- Supports MapReduce
  - Programming model and an associate implementation for processing and generating large datasets of a variety of real-world tasks [2]
Process

- Design LLAPI with MongoDB
  - Implementation
  - Evaluation with SensorLoggingService
Designing LLAPI with MongoDB

Once the lifelog data is stored in the LLCDM, the data is retrieved using a greater queries language MongoDB offers.

Expanding the capability of the previous LLAPI implemented with SQL.

Improved `getLifelog` method is as follows:

```plaintext
getLifeLog([s_date, e_date, s_time, e_time, s_term, e_term, user, party, object, s_alt, e_alt, s_lat, e_lat, s_long, e_long, loc_name, address, location, application, device, content, select, limit, order, offset])
```
Evaluation

Experiment using environmental sensor log from SensorLoggingService, deployed in their smart home.

This service measures environment inside/outside of their laboratory using various sensors including temperature, humidity, brightness, pressure, motion, and the number of people. The sensor has recorded every minute for three years, a total of 1,664,937.

Records are then imported to new (MongoDB, NoSQL) and old (RDB, MySQL) platform.

A client application was developed where it picks out summery days, which means a day that between 9 AM and 6 PM, the maximum temperature exceeds 25 degrees Celsius.
Graph of time (sec) to retrieve data within given time period

- **getSummeryOld**
- **getSummeryNew**

<table>
<thead>
<tr>
<th>Time Period</th>
<th>getSummeryOld</th>
<th>getSummeryNew</th>
</tr>
</thead>
<tbody>
<tr>
<td>1day</td>
<td>0.7926</td>
<td>1.5536</td>
</tr>
<tr>
<td>1week</td>
<td>1.9896</td>
<td>1.572</td>
</tr>
<tr>
<td>10days</td>
<td>2.6586</td>
<td>1.5822</td>
</tr>
<tr>
<td>20days</td>
<td>3.7234</td>
<td>1.6016</td>
</tr>
<tr>
<td>1month</td>
<td>5.3946</td>
<td>1.646</td>
</tr>
<tr>
<td>2months</td>
<td>13.1358</td>
<td>1.7446</td>
</tr>
</tbody>
</table>
Outline

1. INTRODUCTION
2. LIFELOG MASHUP MySQL WITH RDB IMPLEMENTATION
3. LIFELOG MASHUP NoSQL WITH MongoDB IMPLEMENTATION
4. CONCLUSION
   i. Comparisons between SQL and NoSQL
SQL vs. NoSQL

The experimental results showed that the application with the new LLAPI with MongoDB achieves a higher performance and scalability with lower application complexity, compared to the XML and MySQL implementation.
Thank you for your time and attention

QUESTIONS?

Contact:

E-mail:
Lopez477@morris.umn.edu

LinkedIn URL:
www.linkedin.com/in/luz-m-lopez-her
References

