AIDR Persister File Parsing

|  |  |
| --- | --- |
| REQ No.  1 | Project  AIDR Persister Files to Database |

Abstract

This document covers requirements for aidr persister files to database . Comparing No-sql vs Relational DB

| **Rev** | **Date** | **Author** | **Status** |
| --- | --- | --- | --- |
| 1 |  | Ji Lucas | First draft |
| 2 |  |  |  |
|  |  |  |  |

Distribution List

|  |  |
| --- | --- |
| **Reviewers** | **FYI** |
| Ji Lucas |  |

Table of Contents

1. Introduction 4

2. Related Documents/Links/People 4

3. Glossary 4

4. Scope 5

4.1. Goals and Objectives 5

4.2. Assumptions 5

4.3. Constraints 5

4.4. Dependencies 5

4.5. Risks 5

5. Analysis 5

5.1. Overview 5

5.2. Database Requirements 6

5.2.1. General Requirements 6

5.2.2. File Details 6

5.3. Databases 7

5.3.1. MongoDB 7

5.3.2. PostgreSQL 7

5.3.3. Postgres vs MongoDB 10

5.4. Non-functional Related 12

5.4.1. Security 12

5.4.2. QA/Testing 12

Issues/Questions 13

6. Revision History 13

AIDR Persister Files

# Introduction

# Related Documents/Links/People

References in the text throughout this document appear in square brackets (e.g., [1], [JS]).

| **Reference** | **Document/Link/Person/Application** |
| --- | --- |
|  | MicroMappers http://clickers.micromappers.org/ |
|  | AIDR http://aidr-dev.qcri.org/AIDRFetchManager/ |
|  | Digital Humanitarian volunteers coordinator |
|  | Image Clicker http://clickers.micromappers.org/app/MM\_ImageClicker/ |
|  | Text Clicker http://clickers.micromappers.org/app/MM\_TextClicker/ |
|  | MicroFilters |
|  | Video Clicker http://clickers.micromappers.org/app/MM\_VideoClicker/ |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Glossary

| **Term** | **Definition** |
| --- | --- |
| AIDR | Artificial Intelligence for Disaster Response : <http://aidr-dev.qcri.org/AIDRFetchManager/>  To use the application, user has to have twitter account |
| MicroMappers | A cloned customized of Pybossa platform for tagging by Digital Humanitarians http://clickers.micromappers.org/ |
| AIDR Collection Data | Twitter data that is collected by AIDR based on configuration. AIDR Collection output |
| Text Clicker | App that displays tweet text only. Then, user selects one of options |
| Image Clicker | App that displays image only. Then, user selects one of options |
| Video Clicker | App that displays Youtube video only. Then, user selects all scenes that is related damage assessment. |
| Aerial Clicker |  |
|  |  |
|  |  |
|  |  |

# Scope

## Goals and Objectives

## Assumptions

* Predictable data access pattern
* Well-structured dataset(JSON)

## Constraints

* Social Media api
* Application driven

## Dependencies

## Risks

# Analysis

## Overview



## Database Requirements

### General Requirements

| **ID** | **Requirement** | **Cat** |
| --- | --- | --- |
| 1 | Able to handle big data | M |
| 2 | Able to handle JSON | M |
| 3 | Community supports | M |
| 4 | Geospatial support | M |
| 5 | Proven user cases | M |
| 6 | Developer Momentum / Developer consideration | M |
| 7 | Driver fit |  |
| 8 | Does it simplify your world? |  |
| 9 | Does it fail nicely? |  |
| 10 | Low latency, real-time access required |  |
| 11 | Be accessed by users & applications |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### File Details

|  |  |
| --- | --- |
| **Filename:** | {**short\_name**}\_{yyyymmdd}\_vol-{sequence}.json |
| **Zipped?** | No |
| **Zip filename:** | n/a |
| **Field Delimiter:** | n/a |
| **Record Delimiter:** | n/a |
| **Character set:** | UTF-8 |
| **Generation Frequency:** | Record size or daily |
| **Header row?** | N/A |
| **Footer row?** | No |
| **Retrieval location:** | User’s local or URL |
| **Import database:** | NO |

Keys:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **keys** | **Type** | **Len** | **M/O** | **Comments/Data Mapping** |
| 1 | filter\_level |  |  |  |  |
| 2 | retweeted |  |  |  |  |
| 3 | in\_reply\_to\_screen\_name |  |  |  |  |
| 4 | possibly\_sensitive |  |  |  |  |
| 5 | truncated |  |  |  |  |
| 6 | lang |  |  |  |  |
| 7 | in\_reply\_to\_status\_id\_str |  |  |  |  |
| 8 | id |  |  |  |  |
| 9 | extended\_entities |  |  |  |  |
| 10 | in\_reply\_to\_user\_id\_str |  |  |  |  |
| 11 | timestamp\_ms |  |  |  |  |
| 12 | in\_reply\_to\_status\_id |  |  |  |  |
| 13 | created\_at |  |  |  |  |
| 14 | favorite\_count |  |  |  |  |
| 15 | place |  |  |  |  |
| 16 | coordinates |  |  |  |  |
| 17 | retweeted\_status |  |  |  |  |
| 18 | contributors |  |  |  |  |
| 19 | text |  |  |  |  |
| 20 | geo |  |  |  |  |
| 21 | entities |  |  |  |  |
| 22 | **aidr** |  |  |  | * Features * crisis\_code * nominal\_labels * doctype * crisis\_name |
| 23 | source |  |  |  |  |
| 24 | favorited |  |  |  |  |
| 25 | in\_reply\_to\_user\_id |  |  |  |  |
| 26 | retweet\_count |  |  |  |  |
| 27 | user |  |  |  |  |

## Databases

### MongoDB

| **ID** | **Use Case** | **Source** |
| --- | --- | --- |
| 1 | Document oriented Storage |  |
| 2 | Index Support |  |
| 3 | Straightforward Queries. In addition, MongoDB supports MapReduce, which allows for easy lookups in the data |  |
| 4 | Sorting Documents: Finding the Most Recent tweets. MongoDB can’t sort the data in a manageable amount of time, however with an index it is very fast |  |
| 5 | You need High Availability in an Unreliable Environment (Cloud and Real Life)  Setting replicaSet (set of servers that act as Master-Slaves) is easy and fast. Moreover, recovery from a node (or a data center) failure is instant, safe and automatic |  |
|  | Flexible schema & secondary indexing |  |

### PostgreSQL

| **ID** | **Use Case** | **Ref** |
| --- | --- | --- |
|  | **250K tweets**  ANALYZE select substring(data->> 'created\_at'::text [...]  Total runtime: 14602.845 ms, about 14sec. | http://www.rmnd.net/some-experiments-with-postgresql-and-simple-twitter-analysis/ |
|  | **2 million tweets**  ANALYZE select substring(data->> 'created\_at'::text [...]  Total runtime: 96553.320 ms, about 1 minute and a half | http://www.rmnd.net/some-experiments-with-postgresql-and-simple-twitter-analysis/ |
|  | * HStore 1. Key-value pair 2. Simple, fast and easy 3. Postgres v.8.2 – pre-dated many NoSQL-Only solution 4. Ideal for fast data structure that sparsely populated * JSON : Perfected v.9.3 * JSONB : Binary version of JSON. Faster, more operators and even more robust. V.9.4 |  |
|  | Sample postGres, Java, Spring, Hibernate implementation :  http://www.jamesward.com/2012/10/14/nosql-inside-sql-with-java-spring-hibernate-and-postgresql |  |
|  | Postgres + Big data. Release on v.9.5 |  |
|  | POSTGIS : Geospatial support |  |
|  | PostgreSQL has a long track record of use in data warehouses. For a decade before Hadoop launched, PostgreSQL was the only pure open-source option for large data volumes and complex analytics. Today, it is still heavily used in mid-sized data warehouses and "data marts", meaning databases in the one to ten terabyte range. Big data and analytics users are a major subset of the PostgreSQL community, and thus its contributor base, resulting in the addition of new features for large databases with each release. |  |
|  | JSONB vs BSON  JSONB – Backward compatibility with JSON RFC, A variant of JSON optimized for storing objects  BSON – Unable to represent an integer or floating point number with greater than 64bits of precision. Not practical inter-change format |  |
|  | European Union has chosen PostgreSQL as a key part of a publicly funded analytics technology project in the [EU's Seventh Framework Programme](https://ec.europa.eu/research/fp7/understanding/fp7inbrief/what-is_en.html) |  |
|  | PostgreSQL project has been a source of code for many big data startups over the years. These include Netezza, Greenplum, ParAccel, Truviso, Aster Data Systems, and CitusDB. Both Yahoo! Everest and Amazon Redshift are PostgreSQL forks as well.   In July, startup PipelineDB released an [open-source streaming database](https://github.com/pipelinedb/pipelinedb) that was forked from PostgreSQL 9.4.  Streaming databases are used to process huge amounts of incoming data.  These startups are generally attracted to PostgreSQL because of its sophisticated query planner and executor, liberal license, and well-documented code. |  |
|  | - Faster Sorts **- Tablesample :** This clause returns a pseudo-random sample of the rows in the result set. When used with large tables, TABLESAMPLE lets users get a quick "glimpse" of the data so that they can do further analysis.  For example, imagine that I have a table of 10 million user profiles in JSON. I want to take a look at a handful of rows from this table so that I can find out what kinds of keys the JSON has, but I don't want just the rows at the beginning of the table. In 9.5, I can do this:  SELECT \* FROM user\_profiles TABLESAMPLE SYSTEM ( 0.001 );   * **CUBE, ROLLUP, and GROUPING SETSCUBE, ROLLUP, and GROUPING SETS** * **BRIN indexes (Block-Range Index) :** These indexes are much smaller than conventional B+ trees — as little as 1% of the size of the standard indexes. This makes them much faster, since they usually fit in memory, and sometimes even in the CPU cache. * **Foreign Data Wrappers** * **More effective use of large allocations of memory in shared buffers** |  |
|  |  |  |

### Postgres vs MongoDB

| **ID** | **Use Case** |  |
| --- | --- | --- |
|  | selecting, loading and inserting complex document data in key workloads involving **50 million records**  **: http://www.enterprisedb.com/postgres-plus-edb-blog/marc-linster/postgres-outperforms-mongodb-and-ushers-new-developer-reality** | 1. Ingestion of high volumes of data was approximately 2.1 times faster in Postgres 2. MongoDB consumed 33% more the disk space 3. Data inserts took almost 3 times longer in MongoDB   Data selection took more than 2.5 times longer in MongoDB than in Postgres |
|  | Concurrency Issues | When you perform a write operation in MongoDB, it creates a lock on the entire database, not just the affected entries, and not just for a particular connection. This lock blocks not only other write operations, but also read operations. |
|  | You Expect a High Write Load | MongoDB by default prefers high insert rate over transaction safety. If you need to load tons of data lines with a low business value for each one, MongoDB should fit |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Solr

| **ID** | **Use Case** | **Source** |
| --- | --- | --- |
| 1 | Even though Solr is a great tool there are some clear drawbacks of a MySQL + Solr setup.   * **Maintenance**: two software to configure, to monitor * **Continuous Integration**: you need to maintain to schema, two datasource which makes continuous deployment hard and slows down development * **Testing**: testing has to happen from both sources to cover every cases * **Competing for resources**: if the MySQL and Solr are running on the same server they are competing for the same resources (memory, disk) * **Complexity**: Data retrieval and processing algorithms have to be aware of where the origin of the data was * **Indexing**: the data in MySQL needs to be indexed to keep Solr in sync. It isn’t particularly hard but what if we could get rid of that? | http://charlesnagy.info/it/postgresql/postgresql-full-text-search-vs-solr |
| 2 | Query : Postgresql with reconnect for every query is significantly slower but that was expected. What is surprising that with using the existing connection **Postgresql can do 29.6% more queries than Solr per second**. That not just being in par but a significant performance difference. | http://charlesnagy.info/it/postgresql/postgresql-full-text-search-vs-solr |
| 3 | Query time : Roughly the same results. **Postgresql is faster with 27% than Solr in terms of query time**. | http://charlesnagy.info/it/postgresql/postgresql-full-text-search-vs-solr |
|  | Solr is a standalone application and you need to take care about Solr configuration. It is a part of a code since some search business logic depends on it. This might require:   1. different solr config files across different branches 2. test suite that covers some config options 3. different solr instances launched for development and test environments 4. a need to rebuild search index when you switch branches |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| 4 |  |  |
| 5 |  |  |
|  |  |  |

### Evaluation

| **Use Case** | **Postgres** | **MongoDB** |
| --- | --- | --- |
| Will the application need to handle data of varying structure and types? |  |  |
| How large can each data type be – is our data made up of simple integers, strings and timestamps or can it also be large binary files such as images or videos? |  |  |
| Can our data just be represented as a set of opaque values, or does it need to be typed so other applications can make sense of it? |  |  |
| Do we know the data structure will remain constant, or will it vary as we introduce new sensor data and as the business updates application requirements? |  |  |
| Does the application require its data to be strongly consistent (i.e. read our own writes), or can eventually consistent data be tolerated (and do our developers know how to handle the complexity it introduces?).  Do we end up trading performance and availability if we configure the database to only return the freshest data? |  |  |
| What sort of queries are we going to run against the database? Is it simple key-value lookups that we know in advance or do we need to execute ad-hoc queries and complex aggregations to support real-time analytics that the business wants to see? |  |  |
| Will the application be handling geospatial queries and text search? |  |  |
| Which languages will our engineers be using to develop the application, and does the database have drivers available for them? |  |  |
| What capabilities does the database offer to maintain availability during routine maintenance? Are there tools available to manage this or do we need to script something ourselves? |  |  |
| How do we expect this application to grow? Will the database need to scale beyond the limits of just a few servers? |  |  |
| Can we run analytics directly against the database, or do we need to replicate data to dedicated search or analytics engines? |  |  |

## Non-functional Related

### Security

| **ID** | **Requirement** | **Source** | **Cat** |
| --- | --- | --- | --- |
| .1 |  |  |  |

### QA/Testing

| **ID** | **Use Case** | **Date/Status** |
| --- | --- | --- |
| .1 |  |  |

# Issues/Questions

| **Issue #** | **Issue/Resolution Description** | **Date/Status** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Revision History

Changes to the text of this document are indicated by bars in the outside margin adjacent to the affected text.

| **Date** | **Change Description** |
| --- | --- |
|  | Initial draft. |
|  |  |
|  |  |
|  |  |