*Alka Patel*

*Rodrigo Pedro Batista de Oliveira*

**Travel Data Analysis**

# Database Design Document

**Version 2.1**

**03/14/2019**

**Table of Contents**

[**1. Overview 1**](#_3znysh7)

[**2. Assumptions/Constraints/Risks 2**](#_2et92p0)

[2.1 Assumptions 2](#_tyjcwt)

[2.2 Constraints 2](#_3dy6vkm)

[2.3 Risks 2](#_1t3h5sf)

[**3. Design Decisions 3**](#_4d34og8)

[3.1 Key Factors Influencing Design 3](#_2s8eyo1)

[3.2 Functional Design Decisions 3](#_17dp8vu)

[3.3 Database Management System Decisions 3](#_3rdcrjn)

[3.4 Security and Privacy Design Decisions 3](#_26in1rg)

[3.5 Performance and Maintenance Design Decisions 3](#_lnxbz9)

[**4. Detailed Database Design 5**](#_35nkun2)

[4.1 Roles and Responsibilities 6](#_1ksv4uv)

[4.2 Performance Monitoring and Database Efficiency 6](#_44sinio)

[4.2.1 Operational Implications 6](#_2jxsxqh)

[4.2.2 Data Transfer Requirements 6](#_z337ya)

[4.2.3 Data Formats 6](#_3j2qqm3)

[**Appendix A: Acronyms 7**](#_4i7ojhp)

**List of Figures**

No table of figures entries found.

**List of Tables**

[Table 1 - Acronyms 9](#qsh70q)

Overview

## 

The context of our project: we are an Airline company and we want to get a grasp on the factors that can cause an increase or decrease of ticket sales. We would basically follow international trends to see how that promotes or adversely affects travel. A recent example, would be to stream live twitter data on China and Canada relationships as an analysis of the recent happenings with Canada arresting the Chinese Huawei executive, Meng Wanzhou in Vancouver in early December. We can see how this has impacted the Canadian-Chinese relationships in relation to travel behaviours (business and pleasure).

From twitter, we can extract comments, and do a sentiment analysis of people commenting on the situation in China and Canada specifically in regards to “travel”. We can extract additional information like where the tweets are coming from, how many likes they have, how many retweets there were etc.

## Assumptions/Constraints/Risks

### Assumptions

Our project constitutes an efficient (but not extravagant) infrastructure set up to optimize the extraction, storage, cleaning, and analysis of warm temperature data. The exact technologies are described in the document below.

Below are the basic, main steps:

**

### Constraints

1. • Frequency
2. • Retention
3. • Synchronization
4. • Order
5. • Updates

### Risks

We need to take in consideration that a data model for a column-oriented database is different from an analogous model designed for an DBMS.

In order to achieve the same capabilities that a relational database provides on tables, we need to model our data differently to support *standard* relational queries*.*

## Design Decisions

For the purposes of this project, we will focus on a SSD server based approach.

* As we will be handling Live Stream Tweets, our set up for the Cassandra database will be the following:
  + Data Temperature: **Warm Data**

Dealing with daily tweets, we will not need to check them back daily, and the data will be useful for quarterly reports on tickets sales.

* + Type of Storage: **SSD**

Because of our option for **Warm Data**, we will set up our storage on **SSD** type as we would want to frequently access this data for analysis purposes. Moreover, we would want to see the trends over time. So putting it on RAM would mean we lose the data. While the original tweets can be parsed and organized in a panda’s dataframe in the RAM (as we will not want to store those), we would want the insights from those stored in our columnar database setup on the SSD.

Lastly, the business decisions will be made by the marketing team, and we would provide a tableau report for them that would have the following columns below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Topic | No. of Tweets | No. of Likes | No. of Retweets | Location of Tweet | Travel Origin Country | Travel Destination Country | Sentiment: Yes / No to Travel | Date |
|  |  |  |  |  |  |  |  |  |

### Key Factors Influencing Design

Currently, we believe **python to stream** the data and **spark for the ETL to Cassandra**. **Airflow** will be used to run the latter. Additionally, we can leverage pandas in RAM to organize the streamed data and then store in a columnar Cassandra for further analysis. We would also want to set up a tableau view reporting for our marketing team.

* **Our DB setup would work best with BASE properties.**
  + Because we want the latest and the greatest of trends on twitter, we care more about the **speed**, and **throughput** rather than consistency. We also want to keep things simple and **avoid locks**. Twitter data may not always be consistent, and we care more about the **trend analysis** than a numerical hard calculation. Therefore, we believe the **BASE** set up is best.Currently, we believe **python** coding will be sufficient for our streaming, but ever needed, we can use **spark** to scale up the streaming data extraction. We can use pandas in RAM to organize the streamed data and then store in a columnar Cassandra for further analysis. We would also want to set up a **tableau** view reporting for our marketing team.

* **Cassandra:**
  + On **Eric Brewer’s CAP**, our project falls into the **AP category**. We care most about **availability and partition-tolerance**, as opposed to consistency. We need the system to continue operating even despite node failures. Additionally, because we want to **avoid bottlenecks** and functionality despite network failures, we also need partition tolerance. Basically, **no downtime** should be felt! Therefore, we believe **Cassandra** is the best set up. It will allow us the columnar set up we need to do analysis over time. Additionally, Cassandra is also **scalable;** if business grows, or there a multiple social/political trends affecting the market at a given time, we know we can scale using this architectural choices.

### Functional Design Decisions

*Instructions: Describe decisions about how the database will behave in meeting its requirements from a user's point of view (i.e., functionality of the database from an application perspective), ignoring internal implementation, and any other decisions affecting further design of the database. Include decisions regarding inputs the database will accept and outputs (displays, reports, messages, responses, etc.) it will need to support, including interfaces with other systems. Describe the general types of processing (sequential versus random for inserts, updates, deletes and queries) required both for data entering the database, and data most frequently accessed. Also include decisions on how databases/data files will appear to the user.*

### Database Management System Decisions

*Instructions: Describe design decisions regarding the DBMS intended for the initial implementation. Provide the name of the DBMS, the reason for selection, and the type of flexibility built into the database for adapting to changing requirements.*

### Security and Privacy Design Decisions

The types of access we can have are:

* **Full:** Complete access to delete, update and create tables
  + Executives, DBA team
* **Intermediate:** Querying and Viewing: Manipulate the data from the ssd, and view everything on the RAM.
  + Data Analysts
* **Limited:** View access of Tabulated and Queried Data
  + Marketing Team

**Ref:** <http://cassandra.apache.org/doc/latest/operating/security.html?highlight=security>

The following security features are provided from Cassandra, and we deem that as satisfactory for our purposes:

* **TLS/SSL Encryption**
  + Ensures data in flight is not compromised and is transferred securely.
* **Roles**
  + We can create different user (or group) roles using this feature
* **Password Authentication**
  + When a server is launched, our database will be have authentication enabled for additional safety.

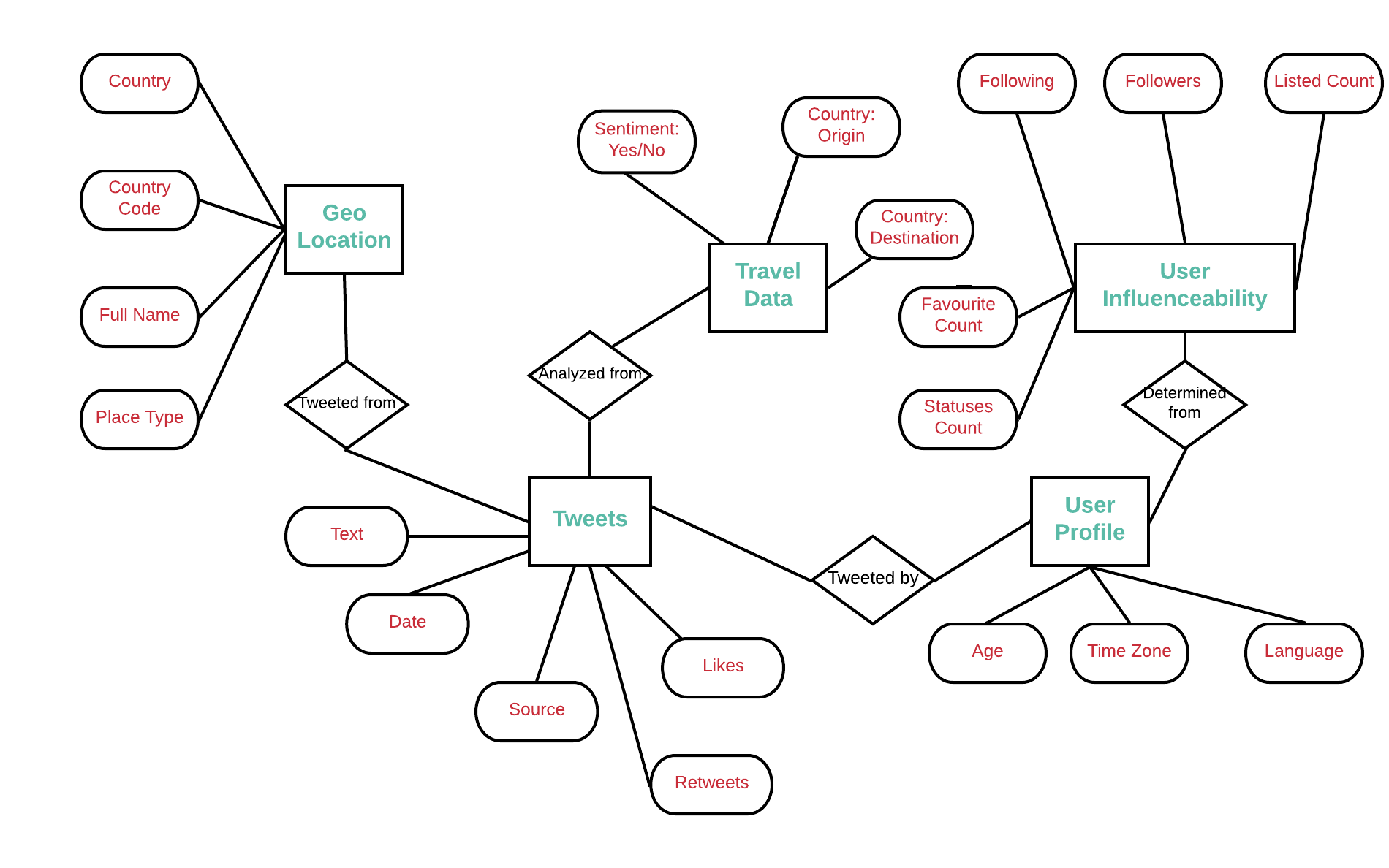
### Performance and Maintenance Design Decisions

*Instructions: Describe how performance and availability requirements will be met. Examples include:*

* *Describe design decisions on database distribution (such as client/server), master database file updates and maintenance, including maintaining consistency, establishing/ reestablishing and maintaining synchronization, enforcing integrity and business rules.*
* *Describe design decisions to address concurrence issues (e.g., how the data are partitioned or distributed to support multiple applications or competing update functions, if applicable).*
* *Describe design decisions to support Service Level Agreements (SLAs) for key functions supported by the database.*
* *Describe design decisions on backup and restoration including data and process distribution strategies, permissible actions during backup and restoration, and special considerations for new or non-standard technologies such as video and sound. Describe the impact this maintenance will have on availability.*
* *Describe design decisions on data reorganization (i.e., repacking, sorting, table and index maintenance), synchronization, and consistency, including automated disk management and space reclamation considerations, optimizing strategies and considerations, storage and size considerations (e.g., future expansion), and population of the database and capture of legacy data. Describe the impact this maintenance will have on availability.*
* *Describe design decisions to support purging and/or archiving of data to ensure performance and storage objectives are met. Describe the impact this maintenance will have on availability. Describe any needs to recall archived data back into the database.*

## Detailed Database Design

* **Conceptual Data Model (CDM)**
  + Our project is based on Tweets feed, the concept design for it would be the following (as per our endpoint file):



**Endpoint snippet:**

{

"statuses": [

{

"created\_at":

"id":

"id\_str":

"text":

"truncated":

"entities": {

"hashtags": [],

"symbols": [],

"user\_mentions": [],

"protected":,

"followers\_count":

"friends\_count":,

"listed\_count":,

"created\_at": "

"favourites\_count":

"utc\_offset":

"time\_zone":

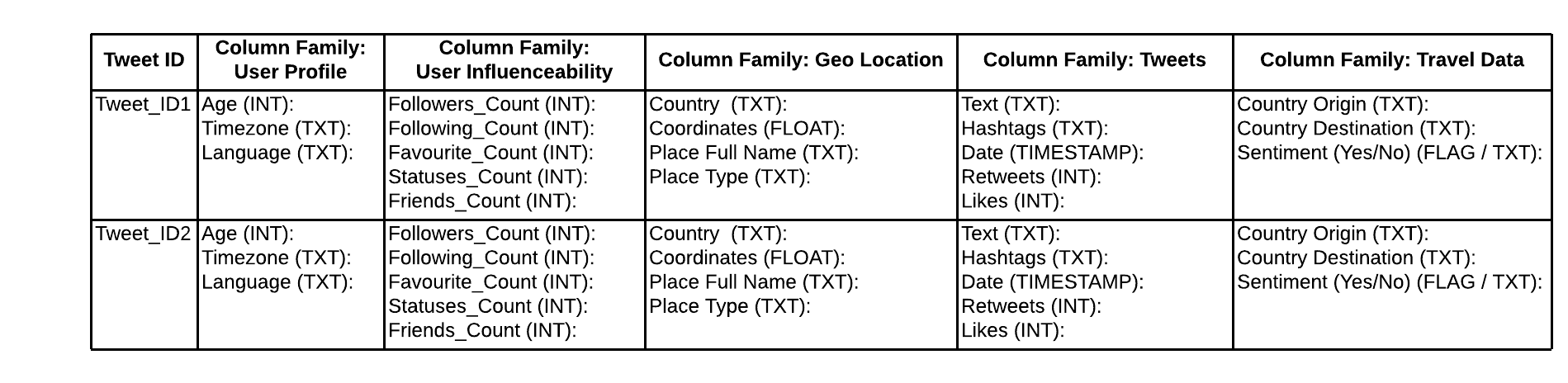
"geo\_enabled":

"verified":

"statuses\_count":

"lang":

* **Logical Data Model (LDM) and LDM Entity Relationship Diagram (ERD)**.
  + Our database will be **Columnar based NoSql, Cassandra** on this case, our ERD would be something like the figure below (still needs to think about the relationships and primary keys):

**

* **Physical Data Model (PDM) with a description of the DBMS schemas, sub-schemas, records, sets, tables.**

CREATE TYPE user\_profile (

age int,

timezone txt,

language txt

);

CREATE TYPE user\_influenceability (

followers\_count int,

following\_count int,

favourite\_count int,

statuses\_count int,

friends\_count int

);

CREATE TYPE geo\_location (

country txt,

coordinates float,

place\_full\_name txt,

place\_type txt

);

CREATE TYPE tweets (

text txt,

hashtags txt,

date timestamp,

retweets int,

likes int

);

CREATE TYPE travel\_data (

country\_origin txt,

country\_destination txt,

travel\_sentiment flag / txt

);

CREATE TABLE tweet\_travel\_analysis (

tweet\_id int,

user\_profile frozen<user\_profile>

user\_influenceability frozen<user\_influenceability>

geo\_location frozen<geo\_location>

tweets frozen<tweets>

travel\_data frozen<travel\_data>

);

* *Planned implementation factors (e.g., distribution and synchronization) that impact the design.*
* *Estimate of the DBMS file size or volume of data per entity.*

Our python script maintains an **open connection to the Twitter Streaming API**, and writes the data into a **json file.**

The idea is to let the program to run daily, get a meaningful data, then when then file reaches the file size of 100 MB, it generates a new file. That limit was based on the number of retweets that we can have on a tweet.

* *Definition of the update frequency of the database tables, views, files, areas, records, sets, and data pages. Also provide an estimate of the number of transactions, if the database is an online transaction-based system.*

While the streaming can be live, we would process the columnar extraction from the jsons in batches. To do this we would make use of the open source **apache Airflow**. We can run the job **daily**, to have to most recent data and trends.

### Roles and Responsibilities

For this project, we will need the following personnel to be able to maintain the environment:

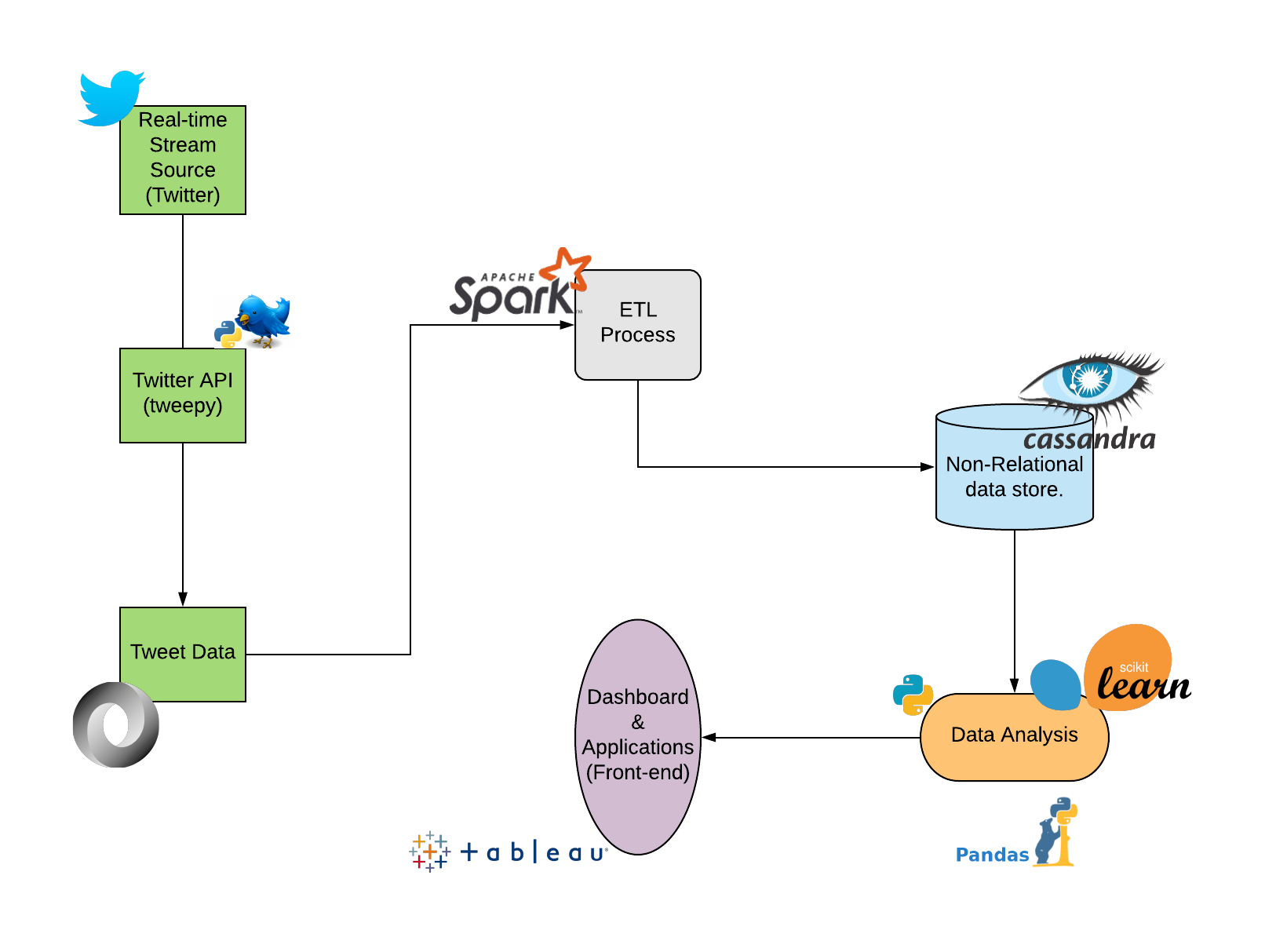
* Role: **DB administrators**
  + Responsibilities: capacity planning, installation, configuration, database design, migration, performance monitoring, security, troubleshooting, as well as backup and data recover on Cassandra DB.
* Role: **Network Administrator**
  + Responsibilities: installing and configuring computer networks and systems, identifying and solving any problems that arise with computer networks and systems, budgeting for equipment and assembly costs, assembling new systems, monitoring computer networks and systems to identify how performance can be improved
* Role: **System Administrator**
  + Responsibilities: user administration (setup and maintaining account), maintaining system, verify that peripherals are working properly, monitor system performance, create file systems, install software, create a backup and recovery policy, setup security policies for users, documentation, automation software such as puppet, chef, etc.
* Role: **Python developer**
  + Responsibilities: Write effective, scalable code, develop back-end components to improve responsiveness and overall performance, integrate user-facing elements into applications, test and debug programs, improve functionality of existing systems, implement security and data protection solutions, assess and prioritize feature requests, expertise in pandas and in the following API: tweepy.
* Role: **Data Analysts**
  + Responsibilities: Organize and query the data to provide useful Business intelligence insights. They will also be responsible for setting up the Tableau reports needed for the marketing team. The marketing team will in turn use these reports to make business decisions in regards to where and how to executive different marketing tactics to ensure growth of ticket sales.

### Performance Monitoring and Database Efficiency

#### Operational Implications

*Instructions: Describe operational implications of data transfer, refresh and update scenarios and expected windows.*

#### Data Transfer Requirements

**

#### Data Formats

* Type of File: **JSON**

Using JSON over XML is better because it is lighter and also JSON data is formatted serially and contain no tags like XML, which makes it easier to read. Because it contains no tags it makes it easier and faster to parse also, and takes less character to represent data.

* Type of Compression: **LZ4**

Compression maximizes the storage capacity of Cassandra nodes by reducing the volume of data on disk and disk I/O, particularly for read-dominated workloads. We will use LZ4 because is fastest to decompress.

**Appendix A: Acronyms**

**Table 1 - Acronyms**

|  |  |
| --- | --- |
| **Acronym** | **Literal Translation** |
| AP | Available and Partition Tolerant |
| API | Application Programming Interface |
| AWS | Amazon Web Services |
| AWS S3 | Amazon Simple Storage Service |
| CAP | Consistency, Availability and Partition tolerance |
| CDM | Conceptual Data Model |
| DB | Database |
| DBA | Database Administrator |
| DBMS | Database Management System |
| ERD | Entity Relationship Diagram |
| Geo | Geostationary Earth orbit |
| I/O | Input/Output |
| INT | Integer |
| LDM | Logical Data Model |
| LZ4 | Lossless Data Compression Algorithm |
| PDM | Physical Data Model |
| RAM | Random Access Memory |
| SSD | Solid State Disk |
| SSL | Secure Sockets Layer |
| TLS | Transport Layer Security |
| TXT | Text |
| XML | eXtensible Markup Language |