# Initial Load

- Script -

Der Initial Load wird ausgeführt, sofern die Datenbank “`socialNetwork`” nicht existiert. Da sich eine sehr große Datenmenge durch den InitialLoad ansammelt, haben wir der Funktion “runInitalLoad” einem Parameter hinzugefügt, um die Anzahl an Tweets zu vermindern und somit stark die Anzahl an Verbindungen zu reduzieren. Tweets sind dabei der Haupt Ausschlaggeber für die große Anzahl an Daten. Die Zahl repräsentiert dabei die Anzahl an Tweets die genutzt werden soll. Für alle Tweets kann der Parameter leer gelassen werden oder -1 übergeben werden.
Für den Initial Load werden folgende Objekte wie folgt angelegt:

1. User:
    - es wird die gegebene Datei “twitter_combined.txt” ausgelesen und die Zeilen mit dem Seperator “ “ in zwei Spalten getrennt und in ein Set gespeichert, um Duplicate zu vermeiden. Das gewonnene Zahlen Set dient als Liste an IDs. Mit der Bibliothek Faker.js werden anschließend zu jeder ID ein Name generiert. DIe Daten werden anschließend in die Collection “users” hochgeladen

2. User follows User:
    - um die Follower abzubilden, wird nochmals die Datei “twitter_combined.txt” genutzt und die zwei Spalten ausgelesen. Die 1. Spalte bildet die IDs des Followers ab und die 2. Spalte die des Followers.
      Um den RAM nicht überlasten, ist der Funktion ein Parameter hinzugefügt worden, um die Edges in Batches rüber zu schicken. So werden z.B. immer N (abhängig vom Parameter) Zeilen ausgelesen und in die DB übertragen, bevor es die nächsten N nimmt.

3. Tweets:
    - die Tweets werden aus der gegebenen Datei tweets.csv gelesen. Zur Vereinfachung verwenden wir den CSV-Parser “csv-parse”. Anschließend entfernen wir die ID, da diese nicht richtig in der CSV-Datei vorliegt und den Autor, da dieser später durch eine Edge repräsentiert wird. Zudem müssen wir die NumberOfLikes und NumberOfShares zu Integern parsen, da diese sonst zum Teil als String vorliegen können und somit nicht mehr für spätere Queries nutzbar wären. Anschließend werden die Tweets hochgeladen und die generierten IDs zwischengespeichert.

4. User wrote Tweet:
    - es werden die unverarbeiteten Tweets aus der Tweet.csv wieder genutzt. Es werden anschließend alle Autoren der Tweets und die Indices derer Tweets rausgesucht und die Autoren einer User-ID zugewiesen. Anschließend werden die Edges erstellt, in dem die Tweet IDs aus 2. den User-IDs mit Hilfe der jeweiligen Indices zugewiesen werden und in die Datenbank abgespeichert werden.

5. User likes Tweet:
    - da es zu viele Likes gibt, als es User gibt, haben wir die Anzahl an Maximalen Likes durch die Anzahl an User geteilt und das Ergebnis abgerundet. Das Ergebnis teilen wir anschließend durch jede NumberOfLikes bei jedem Tweet und weißen anschließend jedem übrig gebliebenen Like einen User zu. Da auch hier eine erhebliche Anzahl an Daten entsteht, starten wir bei jedem Tweet eine Speicherung an die DB, um anschließend das Array wieder zu leeren und so wieder Arbeitsspeicher freizuräumen.

6. Fanout:
    - beim Fanout war schon abzusehen, dass die Erstellung des Caches nochmals größer sein wird als die Anzahl an Edges bei der Likes-Collection. Deswegen gibt es auch hier einen optionalen Limiter, mit dem man die Anzahl der Nutzer reduzieren kann, die ein Fanout-Cache bekommen und so die Auslastung für Testzwecke zu reduzieren.