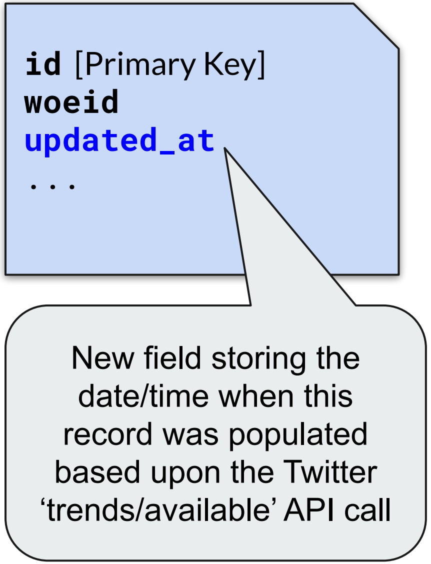
GeoTweet+

Database Changes

‘Locations’ Table

Table: ‘`locations`’



`id` [Primary Key]
`woeid`
`updated_at`
. . .

New field storing the date/time when this record was populated based upon the Twitter ‘trends/available’ API call

Grooming Strategy - ‘`Locations`’

- ***Goal:* Retain all locations that have ever been provided via the Twitter ‘`trends/available`’ API call**
  - The ‘`trends/available`’ API returns the list of locations for which Twitter trends are available
  - Currently the API returns the same 64 locations, all of which are in the U.S. However, this could change - some locations could be added and others no longer used.
  - This strategy will ensure that any location that has ever been obtained will always be available in the ‘`locations`’ table.

- ***Strategy to Avoid Exceeding Database Size Limits:***
  - No action for ‘`locations`’ table:
    Bulk of usage will be in the ‘trends’ table

‘Trends’ Table

Table: ‘**`trends`**’

`id` [Primary Key]
`woeid`
`twitter_as_of`
`twitter_created_at`
`updated_at`
. . .

New field storing the date/time when this record was populated based upon the Twitter ‘trends/available’ API call

Grooming Strategy - ‘**`Trends`**’

- ***Goal:*** **Retain trends obtained via the Twitter ‘`trends/place`’ API call for all locations in the ‘`locations`’ table**
  - Each ‘**`trends/place`**’ API returns info on the top 50 trending tweets for a specific location identified by ‘**`woeid`**’
  - GeoTweet+ will seek to retain all trends data for every entry in the ‘locations’ table, within database size constraints on Heroku
  - Update period options:
    10M records max is sufficient for up to 9 years of trend data collected every 30 mins

- ***Strategy to Avoid Exceeding Database Size Limits:***
  - Trends data will be updated on a periodic basis (hourly?) using update Python code run by Heroku Scheduler
  - The update code will first check if the database is close to limits (10 million DB records).
  - If close to limits, the update code will remove the oldest update of trends data across all locations based upon the value of the ‘update_at’ field

GeoTweet+

API Changes

‘/update’ APIs

API Endpoint	Description	New?	Comment
`/update`	● Returns rate limits on Twitter APIs and count of all entries in ‘locations’ and ‘trends’ tables	Existing	● More output fields: ○ Min/Max date/time of the ‘update_at’ field for both ‘locations’ and ‘trends’ tables ● Same as current GeoTweet behavior
`/update/locations`	● Updates the ‘locations’ table using the Twitter ‘trends/available’ API. ● Returns a count of all entries in the ‘locations’ table.	Existing	● More output fields: ○ Date/Time in the ‘update_at’ field for this update of the the ‘locations’ table ● Same as current GeoTweet behavior
`/update/trends`	● Updates the ‘locations’ table using the Twitter ‘trends/available’ API. ● Returns a count of all entries in the ‘locations’ table.	Existing	● More output fields: ○ Date/Time in the ‘update_at’ field for this update of the the ‘locations’ table ● Same as current GeoTweet behavior

‘/locations’ APIs (1 of 2)

API Endpoint	Description	New?	Comment
`/locations`	● Returns the most recent entries in ‘locations’ table ● For example, if the ‘locations’ table contains locations from updates performed in 5 different date/time periods, locations will be returned only for the most recent update	Existing	● More output fields: ○ ‘Update_at’ field added ○ Min/Max date/time for ‘update_at’ field across all entries in the ‘trends’ table for all locations ● Same as current GeoTweet behavior
`/locations/`*`<woeid>`*	● Returns the entry in the ‘locations’ table for the specified <*woeid*>, if that location was a part of the most recent update of the ‘locations’ table	Existing	● See ‘/locations’ - except mix/max is based upon values returned for this API endpoint ● Same as current GeoTweet behavior
`/locations/tweet/`*`<tweet>`*	● Returns the entries in the ‘locations’ table which have the specified <*tweet*> in their entries in ‘trends’ table, if the location and the trends were a part of the most recent update of the ‘locations’ and ‘trends’ tables	Existing	● See ‘/locations’ - except mix/max is based upon values returned for this API endpoint ● Same as current GeoTweet behavior

‘/locations’ APIs (2 of 2)

API Endpoint	Description	New?	Comment

`/locations/period/<yyyy-mm-dd>:<yyyy-mm-dd>`
`/locations/period/all`
	● Similar to ‘/locations’, except returns all of the entries in ‘locations’ table where the ‘update_at’ field is between the indicated dates.
	● The first date is the start date. The date after the hyphen (-) is the end date.
	● If only one date is specified (with no hyphen), then only that one date will be considered
	● If “/all” is specified or if both start and end dates are omitted, then all dates will be considered
		New	●

`/locations/period/<yyyy-mm-dd>:<yyyy-mm-dd>/<woeid>`
`/locations/period/all/<woeid>`
	● Similar to ‘/locations/*<woeid>*’, except returns locations where the ‘update_at’ field is between the indicated dates.	New	●

`/locations/period/<yyyy-mm-dd>:<yyyy-mm-dd>/tweet/<tweet>`
`/locations/period/all/tweet/<tweet>`
	● Similar to ‘/locations/*<tweet>*’, except returns locations where the ‘update_at’ field is between the indicated dates.	New	●

‘/trends APIs (1 of 2)

API Endpoint	Description	New?	Comment
`/trends`	● Returns the most recent entries in ‘trends’ table for the most recent entries in the ‘locations’ table ● For example: ○ If the ‘locations’ table has 70 entries, 64 of which were obtained in its most recent update, and ○ If the ‘trends’ table has over 400 entries for each location, but only 50 entries per location were obtained in its most recent update ○ In this case, ‘/trends’ will return the 50 most recent trends for the 64 locations in the most recent ‘locations’ table update	Existing	● More output fields: ○ ‘Update_at’ field added ○ Min/Max date/time for ‘update_at’ field across all entries in the ‘trends’ table for all locations ● Same as current GeoTweet behavior
`/trends/<woeid>`	● Returns the most recent entries in ‘trends’ table, if the *<woeid>* was a part of the most recent update of the ‘locations’ table	Existing	● Same as current GeoTweet behavior
`/trends/top/<woeid>`	● Returns the Top 10 tweets (by Tweet Volume) of the most recent entries in ‘trends’ table, if the *<woeid>* was a part of the most recent update of the ‘locations’ table	Existing	● Same as current GeoTweet behavior

‘/trends APIs (2 of 2)

API Endpoint	Description	New?	Comment

`/trends/period/<yyyy-mm-dd>:<yyyy-mm-dd>`
`/trends/period/all`
	● Similar to ‘/trends’, except returns the entries in ‘trends’ table for locations in the ‘locations’ table where the ‘update_at’ field in both tables is between the indicated dates.
	● For example:
	○ If the ‘locations’ table has 70 entries, 64 of which were obtained in its most recent update, and
	○ If the ‘trends’ table has over 400 entries for each location, but only 50 entries per location were obtained in its most recent update
	○ In this case, ‘/trends/period/all’ will return the all of the entries in the trends table
		New	● TBD

`/trends/period/<yyyy-mm-dd>:<yyyy-mm-dd>/<woeid>`
`/trends/period/all/<woeid>`
	● Returns the entries in ‘trends’ table for the *<woeid>* in the ‘locations’ table where the ‘update_at’ field in both tables is between the indicated dates.	New	● TBD

`/trends/period/<yyyy-mm-dd>:<yyyy-mm-dd>/top/<woeid>`
`/trends/period/all/top/<woeid>`
	● Returns the Top 10 tweets (by Tweet Volume) of the entries in ‘trends’ table for the *<woeid>* in the ‘locations’ table where the ‘update_at’ field in both tables is between the indicated dates.	New	● TBD

'/prune' APIs

API Endpoint	Description	w?	Comment
~~`/update/prune/locations`~~ ~~`/update/prune/locations/<n_days>`~~ ~~`/update/prune/locations/<woeid>`~~ ~~`/update/prune/locations/<woeid>/<n_days>`~~	- ~~Removes entries from the '~~ ~~'updated_at' is older '~~ ~~<n_days> or a d~~ - ~~If location~~ ~~locati~~ - ~~ed, then that~~ ~~ions' table only if~~ ~~ays>~~		- ~~This API might mostly be useful for debugging, etc. Ordinarily there will be no need to manually prune the entries in this table.~~
~~`/update/prune/trends`~~ ~~`/update/prune/trends/<n_days`~~ ~~`/update/prune/trends/<wo`~~ ~~`/update/prune/trends/`~~	~~the 'trends' table that are older~~ ~~integer <n_days> or a default value~~	~~New~~	- ~~This API might mostly be useful for debugging, etc.~~

Will Not Implement This
/prune API is too dangerous --
Would allow any user to delete some/all of our data