# Advanced Database Technologies Course

## -Day 2-

**Vijay Gadepally, Chansup Byun, Lauren Edwards,**
**Jeremy Kepner, Julie Mullen, Scott Sawyer**

**August 2014**

# Course Overview

- Day 1 – Advanced Database Technologies
  - Introduction to Big Data/Cloud Computing [30 min]
  - Introduction to Database Technologies [45 min]
    - Different technologies
    - Interacting with different technologies
    - What do sponsors use?
  - Database technology - Apache Accumulo [45 min]
    - Overview
    - Architecture/Components
    - How it works
    - Schema/Schema design
- **Day 2 – Using Accumulo at Lincoln Laboratory**
  - **D4M technology [75 min]**
    - **About**
    - **Associative Arrays**
    - **Hands-on Example**
  - **Using Python Thrift Bindings to access Apache Accumulo [40 min]**
  - **Conclude and summarize [5]**

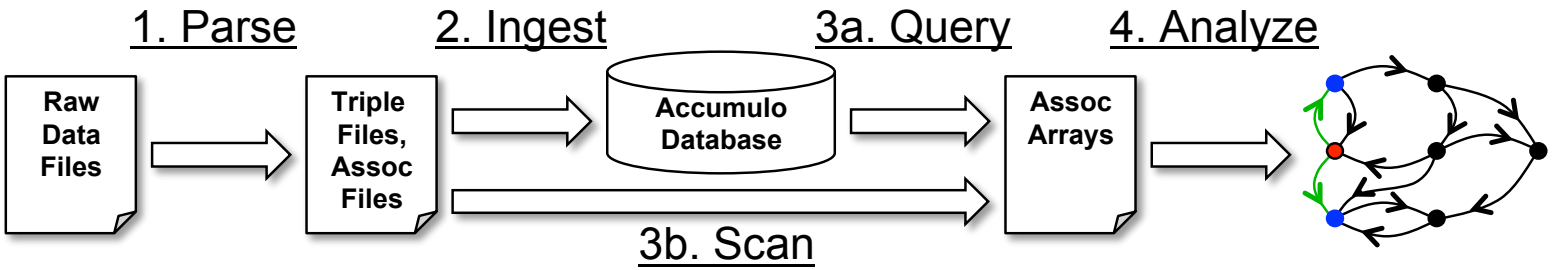
# Day 2

- **Goal: Reinforce the concepts needed through hands-on examples.**
- **Work on using Apache Accumulo through two**

**Course Goal:**
**To give you the knowledge needed to effectively use Apache Accumulo and knowledge about other DB technologies**

# Part 1 Refresher

- Big Data produces many challenges which are addressed by a computing cloud
- Many different options in this cloud and very specific to intended application
- Databases form an important part of most cloud infrastructures
- Database choices are vast and often highly depend upon the intended application
- Apache Accumulo is a popular NoSQL database



# Apache Accumulo

- High Performance open source database
- Accumulo is based on Google's BigTable* design and is a *tabular key-value store*
- Each "cell" or "entry" in Accumulo is a key (a tuple) mapped to a value:
  - (row, column) → value
  - Each row and column has a string label
- Entries are organized into tables
- Portion of table persisted as a tablet, stored on tablet servers
- We use the D4M schema to represent unstructured data in the database

# About the dataset

- **Have 8 databases loaded with approximately 2 million tweets**
  - **Organized the same was as much larger production sets available on LLGrid**
- **Raw data organized in TSV files:**

| TweetID | Time | Lat | Long | User | Source | Text |
|---|---|---|---|---|---|---|
| 334458513407488001 | 2013-05-15 00:01:38 | 41.50442327 | -71.3080676 | alesiatasso | Twitter for iPhone | @LM_Rudd okay, I'm... |
| 334458512308576256 | 2013-05-15 00:01:37 | 42.56289604 | -84.82935883 | skally_pal | Twitter for iPhone | @ItsMackk I'll see you… |
| 334458513755602944 | 2013-05-15 00:01:38 | 41.39560229 | -81.73950863 | AmbahDee | Twitter for iPhone | @NataliaProkop come… |
| 334458515798245377 | 2013-05-15 00:01:38 | 37.19058402 | -93.31629432 | RockSteady | Twitter for iPhone | @ktp35 I didn't think to... |
| 334458515383005184 | 2013-05-15 00:01:38 | 35.6337457 | 139.6607298 | karate_h | Twitter for Android | """@IngatanSekolah: … |
| 334458520244219904 | 2013-05-15 00:01:39 | 42.2041867 | -87.8126571 | yobebeau | Endomondo | Was out cycling 31.64 … |
| … | … | … | | … | | … |

# Tweet Entry Contents
# -raw data-

```
337110784792461314      2013-05-22 07:40:48      37.3801915      -5.9617655
        -663660.699597849 4492233.90526212   568424410     laura_chiri
"<a href=""http://twitter.com/download/android"" rel=""nofollow"">Twitter for Android</a>"
                          1446282319     8c86b8b4cb716103
  @zahara_fdez  guapaaa !! Al fin por aquii !! Jajaja menos mal que ya era hora... Besitoos !!
```

- Tweet ID: Unique Twitter identifier
- Time: Time sent GMT
- Latitude (if available)
- Longitude (if available)
- Tweet X coordinate: Spherical Mercator projection
- Tweet Y coordinate: Spherical Mercator projection
- Sender ID
- Sender Name
- Source: Agent used for Tweet
- Reply to User ID
- Reply to Tweet ID
- Place ID: Twitter ID when GPS not used
- Tweet Text

# Tweet Entry Reference -tabular view-

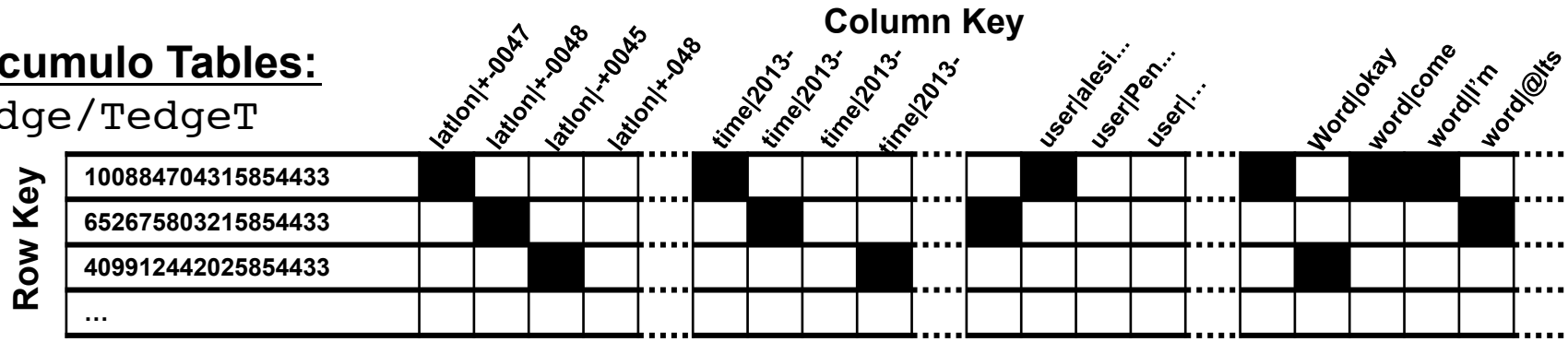
| Field | Example Entry |
|---|---|
| Tweet ID: Unique Twitter identifier | 3371107847924610000 |
| Time: Time sent GMT | 2013-05-22 07:40:48 |
| Latitude (if available) | 37.3801915 |
| Longitude (if available) | -5.9617655 |
| Tweet X coordinate: Spherical Mercator projection | -663660.6996 |
| Tweet Y coordinate: Spherical Mercator projection | 4492233.905 |
| Sender ID | 568424410 |
| Sender Name | laura_chiri |
| Source: Agent used for Tweet | "<a href=""http://twitter.com/download/android"" rel=""nofollow"">Twitter for Android</a>" |
| Reply to User ID | 1446282319 |
| Reply to Tweet ID | 8c86b8b4cb716103 |
| Place ID: Twitter ID when GPS not used | 7238f93a3e899af6 |
| Tweet Text | @zahara_fdez guapaaa !! Al fin por aquii !! Jajaja menos mal que ya era hora... Besitoos !! |

# Twitter Schema/Organization

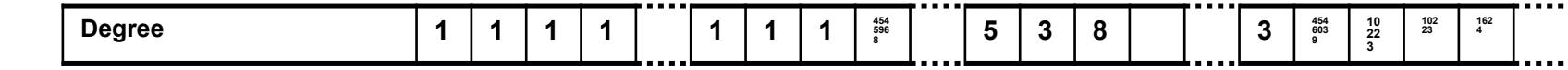
- **Use the D4M Schema to "Explode" TSV files:**

**Accumulo Tables:**

`Tedge/TedgeT`




`TedgeDeg`$^{t}$




`TedgeTxt`

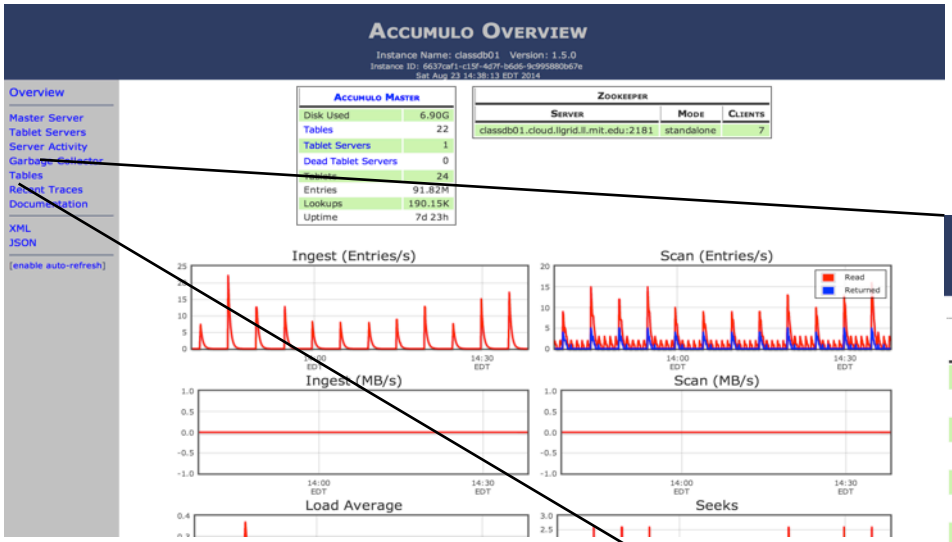
| Row Key | text |
|---|---|
| 100884704315854433 | @LM_Rudd okay, I'm gonna get in comfy clothes then drive to your house then we can adventure! lol |
| 652675803215854433 | @ItsMackk I'll see you tomorrow while we're moving |
| 409912442025854433 | Was out cycling 31.64 miles with #Endomondo. See it here: http://t.co/9CpxSipkIs |
| ... | ... |

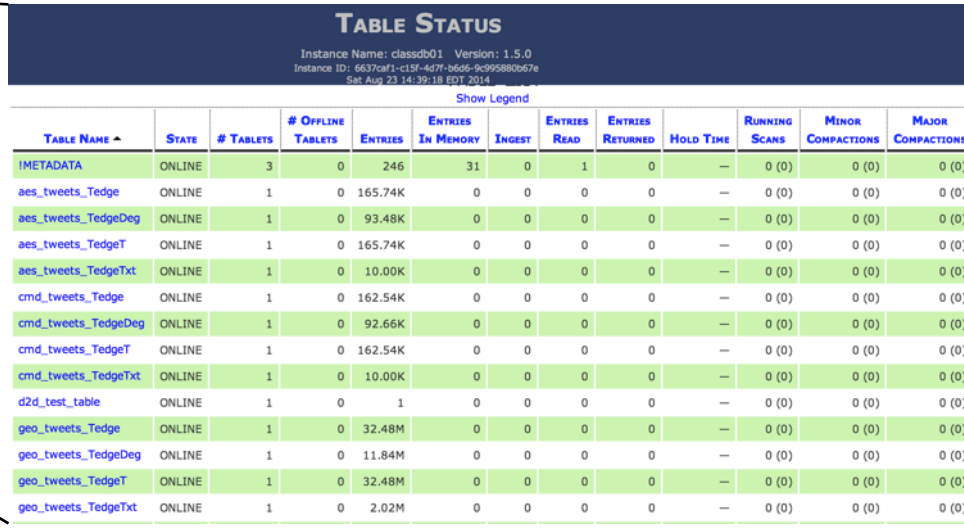
# 4 tables to rule them all…

- Tedge and TedgeT
  - Table pair that contain the raw data
  - Named edge table because they represent the edge table of a graph
  - Tedge Rows: Unique Tweet ID, Tedge Columns: Tweet values, Tedge Values: Usually just 1
- TedgeDeg
  - Table that contains a count of all columns in Tedge
  - Named degree table because it represents the degree of the graph (number of edges from a vertex)
  - TedgeDeg Rows: Unique Tweet ID, TedgeDeg Columns: Tweet values, TedgeDeg Values: Count
- TedgeTxt
  - Table that contains the raw tweet text
  - TedgeTxt Rows: Unique Tweet ID, TedgeTxt Columns: The string ‘text’, TedgeTxt Values: Raw text

# Online Monitoring System

- Navigate a web browser to: https://dbstatus.llgrid.ll.mit.edu/
  - Log in using LLGrid credentials
- Click on “View Info” next to your assigned classdb
  - Please try not to hit “Stop” ☺




| Table Name | State | # Tablets | # Offline Tablets | Entries | Entries In Memory | Ingest | Entries Read | Entries Returned | Hold Time | Running Scans | Minor Compactions | Major Compactions |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| !METADATA | ONLINE | 3 | 0 | 246 | 31 | 0 | 1 | 0 | – | 0 (0) | 0 (0) | 0 (0) |
| aes_tweets_Tedge | ONLINE | 1 | 0 | 165.74K | 0 | 0 | 0 | 0 | – | 0 (0) | 0 (0) | 0 (0) |
| aes_tweets_TedgeDeg | ONLINE | 1 | 0 | 93.48K | 0 | 0 | 0 | 0 | – | 0 (0) | 0 (0) | 0 (0) |
| aes_tweets_TedgeT | ONLINE | 1 | 0 | 165.74K | 0 | 0 | 0 | 0 | – | 0 (0) | 0 (0) | 0 (0) |
| aes_tweets_TedgeTxt | ONLINE | 1 | 0 | 10.00K | 0 | 0 | 0 | 0 | – | 0 (0) | 0 (0) | 0 (0) |
| cmd_tweets_Tedge | ONLINE | 1 | 0 | 162.54K | 0 | 0 | 0 | 0 | – | 0 (0) | 0 (0) | 0 (0) |
| cmd_tweets_TedgeDeg | ONLINE | 1 | 0 | 92.66K | 0 | 0 | 0 | 0 | – | 0 (0) | 0 (0) | 0 (0) |
| cmd_tweets_TedgeT | ONLINE | 1 | 0 | 162.54K | 0 | 0 | 0 | 0 | – | 0 (0) | 0 (0) | 0 (0) |
| cmd_tweets_TedgeTxt | ONLINE | 1 | 0 | 10.00K | 0 | 0 | 0 | 0 | – | 0 (0) | 0 (0) | 0 (0) |
| d2d_test_table | ONLINE | 1 | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 (0) | 0 (0) | 0 (0) |
| geo_tweets_Tedge | ONLINE | 1 | 0 | 32.48M | 0 | 0 | 0 | 0 | – | 0 (0) | 0 (0) | 0 (0) |
| geo_tweets_TedgeDeg | ONLINE | 1 | 0 | 11.84M | 0 | 0 | 0 | 0 | – | 0 (0) | 0 (0) | 0 (0) |
| geo_tweets_TedgeT | ONLINE | 1 | 0 | 32.48M | 0 | 0 | 0 | 0 | – | 0 (0) | 0 (0) | 0 (0) |
| geo_tweets_TedgeTxt | ONLINE | 1 | 0 | 2.02M | 0 | 0 | 0 | 0 | – | 0 (0) | 0 (0) | 0 (0) |

# D4M and MATLAB Fundamentals

# D4M

- Library that allows unstructured data to be represented as triples in Associative Arrays and can be manipulated using standard linear algebraic operations.
  - Currently implemented in MATLAB
  - Complex database database operations and queries are composable as algebraic operations on associative arrays
  - D4M API makes it easy to develop analytics (just a few lines of code)
- Website: d4m.mit.edu
- A collection of tutorials:
  - https://wiki.llan.ll.mit.edu/display/LLGRID/LLGrid+Home

    ->User Portal -> D4M

# Associative Arrays

- **Extends associative arrays to 2D and mixed data types**

```
A('#a1','#b2') = 'same_tweet'
```

- **Key innovation: 2D is 1-to-1 with triple store**

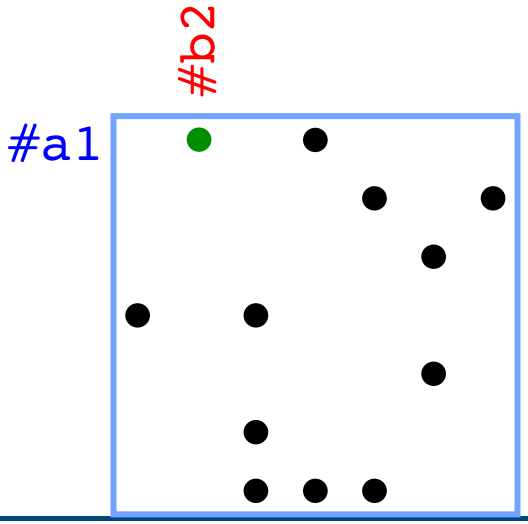
```
('#a1','#b2','same_tweet')
```

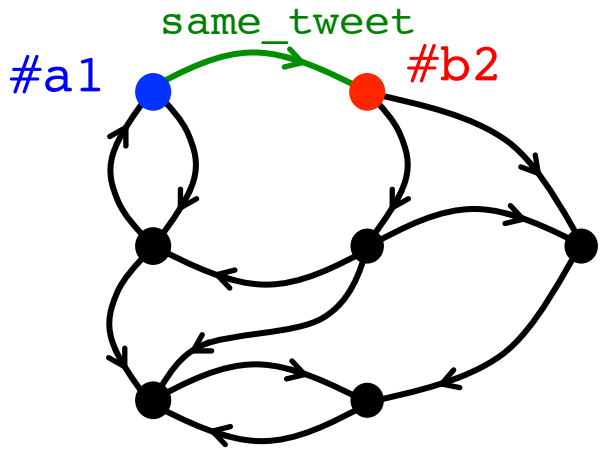
- **Enables composable mathematical operations**

```
A + B      A - B      A & B      A|B      A*B
```

- **Enables composable query operations via array indexing**

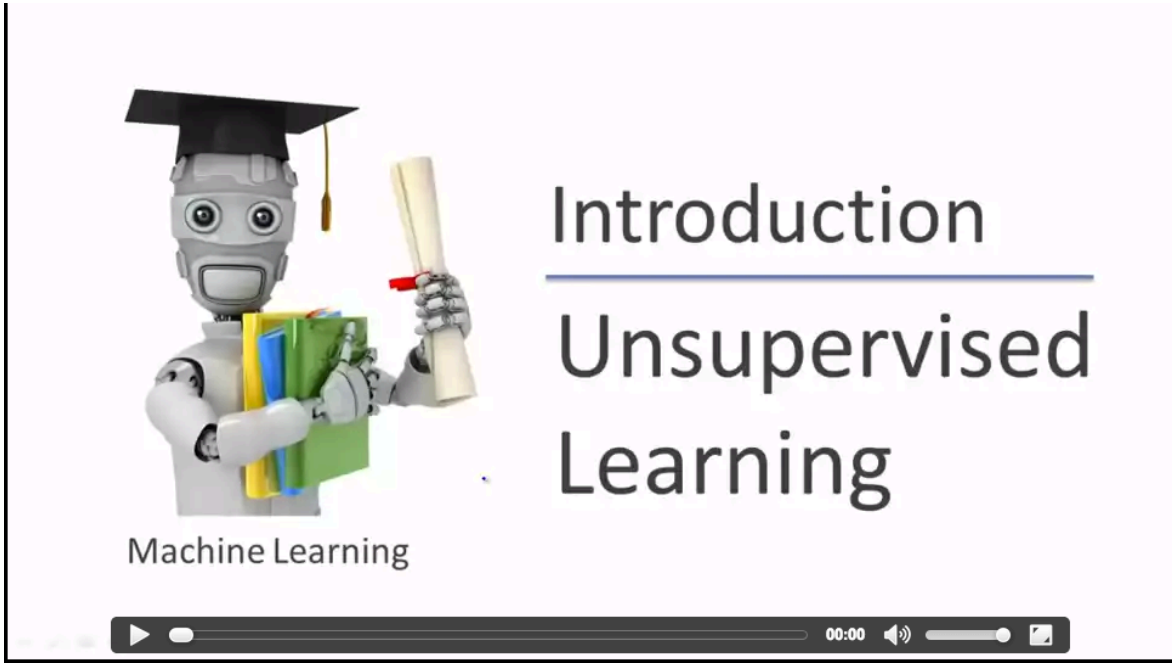
```
A('#al b2',:)     A('#al,',:)     A('#a* ',:)
A('#al: b2',:)    A(1:2,:)          A == #b2
```



# Why MATLAB or Octave?

- High level languages
- Very very useful in algorithm design - Don't trust me?
- LINK



# MATLAB Fundamentals

- MALTAB Command Prompt:

```
>>
```

- The help command:

```
>>help topic_name
```

- Find workspace variable:

```
>>whos
```

- Get information on a specific VAR:

```
>>whos VAR
```

- Saving and loading workspace variables:

```
>>save filename VAR1 VAR2 …
>>save filename
>>load filename
```

- Operate (+,-,/,*) on VAR1 and VAR2 and save in VAR3

```
>>VAR3 = VAR1 operator VAR2
```

# D4M API
# -Using Twitter Corpus-

- **Setting Bindings for Tables Tedge, TedgeT, TedgeDeg, TedgTxt:**

```
>> T = DB('MyTableName');
```

- **Querying for data from Table T and storing in Assoc A**

```
>> A = T(row_key, column_key)
```

- **To get all data from Table T and store in Assoc A**

```
>> A = T(:,:) %":"  colon indicate all results.
```

- **row_key and column_key have a format:**
  - **'#a,#b,#c,' queries for #a, #b, #c**
  - **The ending comma is necessary delimiter to include in query string.**
  - **'#a,:,#c,' will query for a range, from #a through #c**
  - **column_key in Tedge contains 'field' appended with word. For example:**

```
column_key1 = 'word|#randomtext,'
column_key2 = 'time|2013-05-20 00:01:01,')
column_key3 = 'latlon|++003301..01267540583244000000,'
```

  - **row_key uses reversed tweet id. For example:**

```
row_key1 = '080661751978409633,'
```

# Some useful D4M Commands

```
help D4M %D4M help and documentation

%Setting up DB Bindings%

>>DB = DBsetupLLGrid(Dbname); %Returns DB bindings

>>Tedge = DB(['Tedge'],['TedgeT']); %Bindings for Tedge table

>>addSplits(TableName,rowSplit) %Pre process ingest to split
table at rowSplit

%Data Ingesting%

>>put(TableName, A) % put associative array A into TableName

%Data Querying – TableName can be any Table such as Tedge%

>>keywordFull = StartsWith(keyword) %return string that can be
used as range query

>>nnz(TableName) % returns number of entries in TableName

>>A = TableName(:, 'keyword,') %returns all entries in Table name
that have column keyword

>>[r c v] = find(A) %returns all entries in Assoc A

>>Assoc2KML(A) %generates an kml formatted file with geo
information contained in associative array A
```

# Some basic D4M examples

- Open MATLAB

```
$matlab -nodisplay
```

- Set up database bindings for https://dbstatus.llgrid.ll.mit.edu/dbfw/classdb01/Accumulo/

```
>>dbSetup
```

- See workspace variables

```
>>whos
```

- Get information about variable Tedge

```
>>whos Tedge
```

- Find number of elements in Tedge

```
>>nnz(Tedge)
```

- Find number of tweets in corpus?
- Find number of unique strings?

Can also use
online DB monitor:
https://dbstatus.llgrid.ll.mit.edu/

# Iterators

- Allows a programmer to traverse the elements of a table without worrying about current position.
- VERY important to use when dealing with large queries!
- Syntax:

```
Ti = Iterator(TableName,'elements',5000);
A = Ti(:, keyword)
while nnz(A)
   …do something...
   A = Ti();
end
```

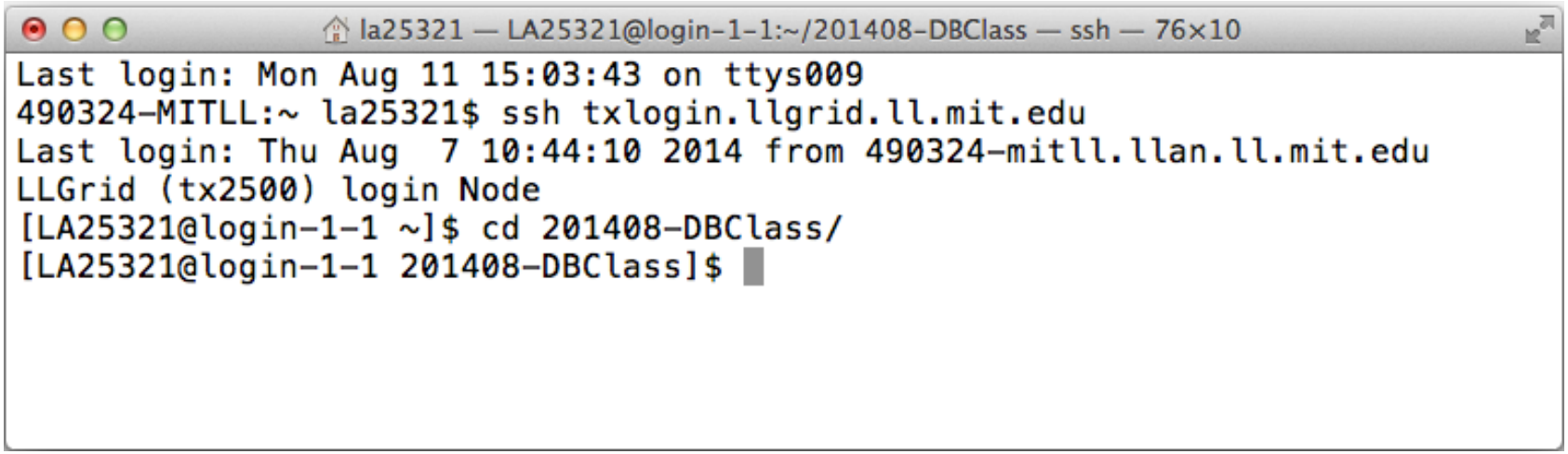
# Log into TX-2500

- Open a terminal window
- At the prompt ($), type the following command:

```
$ssh txlogin.llgrid.ll.mit.edu
```

- Then go to the directory containing the class files:

```
$cd 201408-DBClass/
```




**Tip: Write the first few characters of the directory/file name, then press tab. This will autocomplete the directory/file name.**

# MATLAB Exercises: Editing Files

- If you have LLGrid mounted on your desktop, you can open the file in MATLAB and edit them, or use any other editor you like
- For LLGrid in the terminal, use vi, emacs, nano, etc.
  - To start these editors, use the respective command below:

```
$vi dbSetup.m        $emacs dbSetup.m

$nano dbSetup.m
```

- For help with either navigating to the correct folder in MATLAB or using one of these editors, see additional slides and cheat sheets

# MATLAB Exercises: Before You Start…

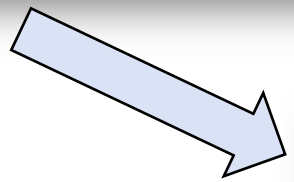
- Each exercise that accesses the database has its own dbSetup.m file which says which database to connect to
- These need to be edited to reflect the one you have been given
  - Go to the main MATLAB Exercises directory
  - Edit the dbSetup.m as described on the following slide
  - Copy this file into the directories that need it

```
$cd ~/201408-DBClass/MATLABExamples

$vi dbSetup.m

$cp dbSetup.m 1BasicQuery/
$cp dbSetup.m 3PerfectPowerLaw/
$cp dbSetup.m 5PopularWords/
```

# MATLAB Exercises: Before You Start…

- Comment out the disp('Set your DB Number') line (3)
- Uncomment the following line and change YOURDBNUMBER to the database number you have been given

```
% Create/Bind to a DB.

disp('Set your DB Number');
% DB = DBsetupLLGrid('classdbYOURDBNUMBER');
```

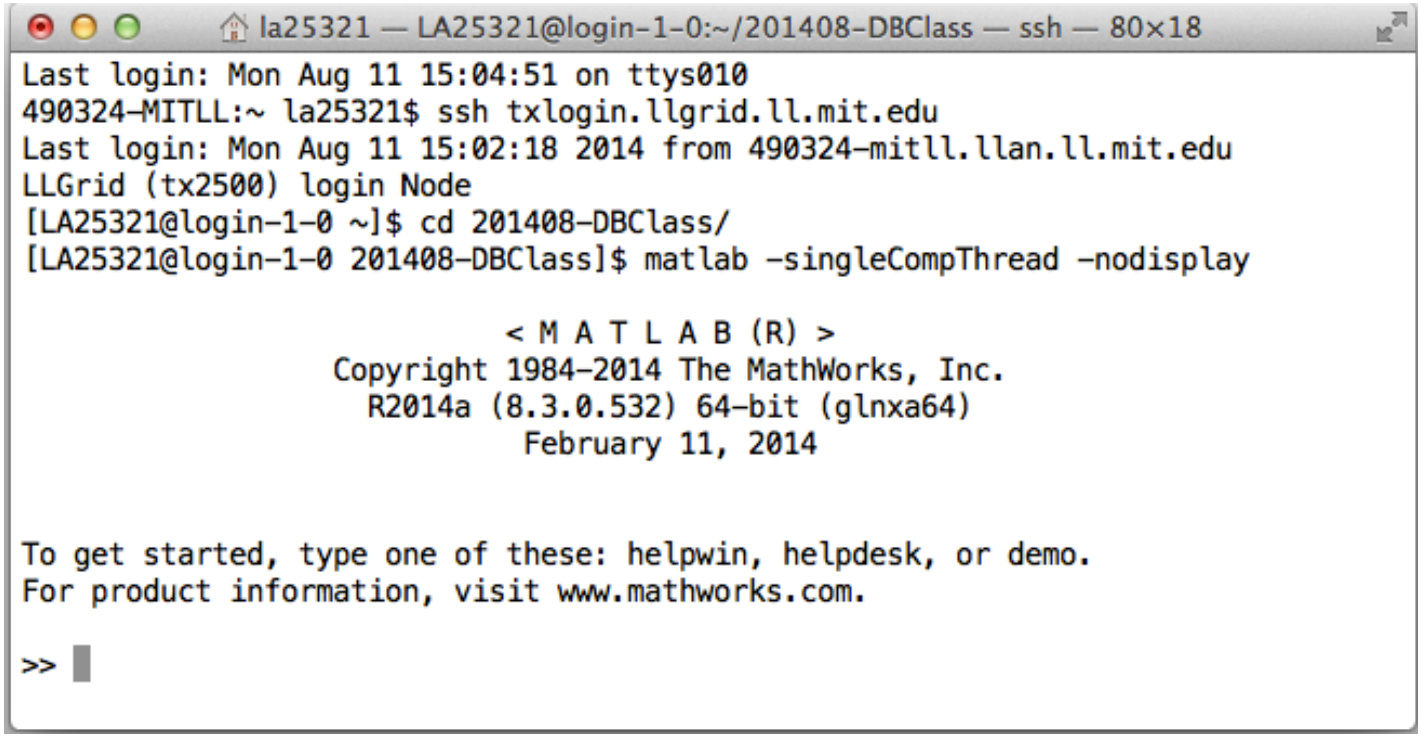


```
% Create/Bind to a DB.

% disp('Set your DB Number');
DB = DBsetupLLGrid('classdb01');
```

# Start MATLAB in Terminal

- After logging into LLGrid in the terminal…
- Start MATLAB by issuing the following command:

```
$matlab –singleCompThread –nodisplay
```



# Exercise 0: Get info from the DB

- Find and display tweets with #happiness in them
- Procedure:
  - Step 0) Go to the directory with the example
  - Step 1) Setup bindings to table (don't forget to edit your dbSetup.m)
  - Step 2) Use table TedgeDeg to find number of count of '#happiness'
  - Step 3) Do we need an iterator?
  - Step 4) Query for Row Keys using Tedge, save to Assoc B
  - Step 5) Use Row Keys and TedgeTxt to see tweets

```
>>cd ~/201408-DBClass/MATLABExamples
>>dbSetup;
>>A1 = TedgeDeg('word|#happiness,',:);
>>A2 = Tedge(:,'word|#happiness,');
>>r = Row(A2);
>>display(TedgeTxt(r,:));
```

# Exercise 1: Basic Query

- This exercise:
  - Sets up the database connection using the DBsetup.m script
  - Finds the number of times "#prayforoklahoma" is used
  - Gets all tweets using "#prayforoklahoma"
  - Outputs these tweets to a KML file
  - Can be opened in Google Earth to show tweet location

In MATLAB (`$matlab –singleCompThread –nodisplay`)

Go to directory for the first exercise.

Run the exercise.

```
>>cd ~/201408-DBClass/MATLABExamples/1BasicQuery
>>exercise1
```

# Exercise 1: Basic Query

- **Procedure**
  1. Setup bindings to database.
  2. Set the keyword
  3. Find the number of times the query appears by querying the TedgeDeg table.
  4. Query the table. If keyCount is large, use an iterator (next slide).
     a. Get the row keys of the tweets using the keyword.
     b. Query Tedge and TedgeTxt for the tweets.

```
>>dbSetup;
>>keyword = 'word|#prayforoklahoma,';
>>keyCount=TedgeDeg(keyword,:);
>>Atmp = Tedge(:, keyword);
>>A = Tedge(Row(Atmp),:);
>>Atxt = TedgeTxt(Row(Atmp),:);
```

# Exercise 1: Basic Query

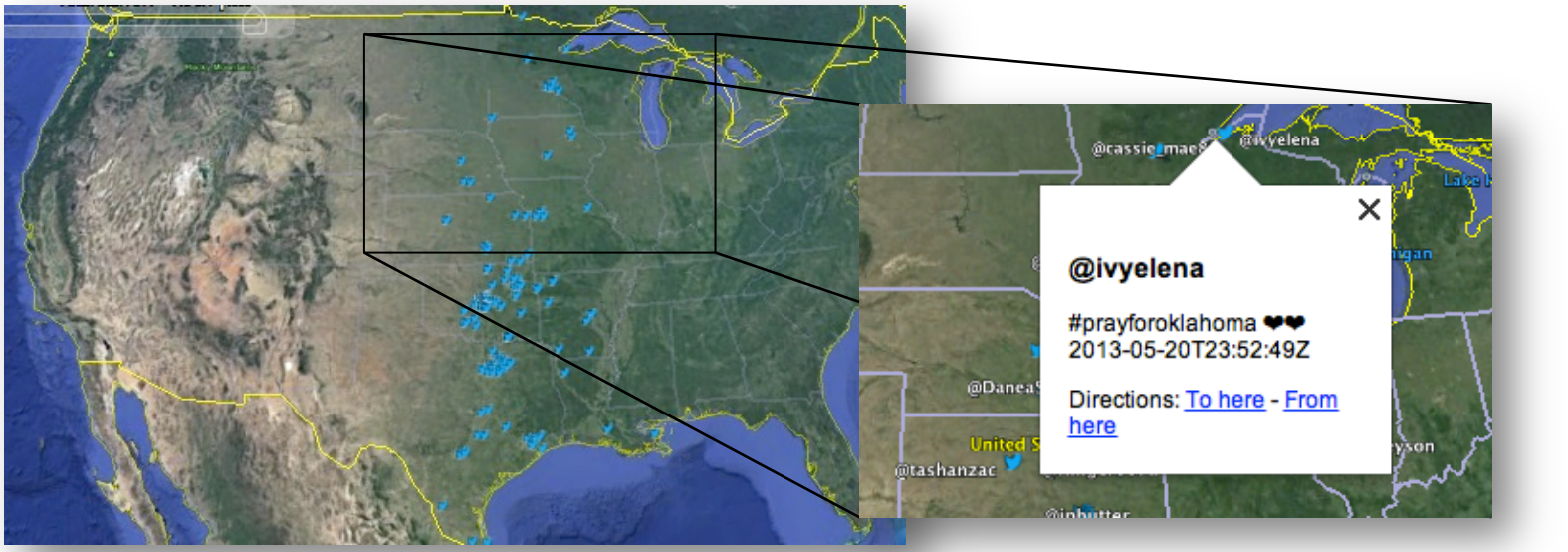
- To use an iterator:
  1. Set up an iterator that iterates over Tedge, 500 elements at a time.
  2. Get the first 500 and make Associative Arrays to collect the output.
  3. Iterate through the table.

```
>>Ti = Iterator(Tedge, 'elements', 500);

>>Arow = Ti(:,Row(keyCount));

>>A = Assoc('','',''); Atxt = Assoc('','','');

while nnz(Arow)          %while no nonzero elements in Arow
 r = Row(Arow);          %get row keys in Arow
 A = A + Tedge(r,:);     %query Tedge and add to A
 Atxt=Atxt+TedgeTxt(r,:); %query TedgeTxt and add to Atxt
 Arow = Ti();            %get next 500 rows
end
```

# Exercise 1: Basic Query

- To visualize the output:
  1. Remove word columns from A and add full text.
  2. Use the Assoc2KML function to make a KML file.
  3. Open the KML file in Google Earth to view.

```
>>A = A - A(:, StartsWith('word|,')) + Atxt;

>>Assoc2KML(A);
```



# Exercise 2: Dimensional Analysis

- This exercise runs on 2 processors on LLGrid and finds, for each column type, the number of columns, rows, and nonzero entries
- In MATLAB (`$matlab –singleCompThread –nodisplay`)

  Go to directory for the second exercise

  Then run the exercise

```
>>cd ~/201408-DBClass/MATLABExamples/2DimAnalysis

>>eval(pRUN('dimAnalysis',2,'grid&'));

>>load('Atally.mat')
```

- To view the job status, enter

```
>>LLGrid_myjobs
```

# Exercise 3: Perfect Power Law

- This exercise:
  - Queries TedgeDeg to find the number of tweets each user has made
  - Plots a histogram of this data
  - Finds the Power Law Parameters
  - Finds the Power Law Fit
  - Rebins the measured data to the Power Law Fit
  - Creates plots of each piece above

In MATLAB (`$matlab –singleCompThread –nodisplay`)

Go to directory for the first exercise

Then run the exercise

```
>>cd ~/201408-DBClass/MATLABExamples/3PerfectPowerLaw
>>PerfPowerLaw
```

# Exercise 4: Topic Modeling

- This exercise conducts topic modeling on tweets by selecting language as the Topic IDs and then finds the popular words in each topic
- In MATLAB (`$matlab –singleCompThread –nodisplay`)

  Go to directory for the first exercise

  Then run the exercise

```
>>cd ~/201408-DBClass/MATLABExamples/4TopicModeling

>>D4M_TwitterExample
```

# Fill-in Exercise 1: Popular Words

- **On LLGrid (outside MATLAB):**
  - **Go to directory for the second exercise.**
  - **Edit the "PopularWords.m" script as instructed.**
    - **If you have LLGrid mounted on your desktop, you can open the files in MATLAB and edit them, or use any other editor you like**
    - **For LLGrid in the terminal, use vi, emacs, nano, etc.**

```
$cd ~/201408-DBClass/MATLABExamples/5PopularWords
$vi PopularWords.m          $emacs PopularWords.m
                $nano PopularWords.m
```

**For help with either navigating to the correct folder in MATLAB or using one of these editors, see additional slides and cheat sheets**

# Fill-in Exercise 1: Popular Words

- Find words that occur often with some given keyword
- Use what you have seen in previous examples
  - Hints may tell you where to look if you are stuck
- Fill in code after comments with an asterisk *

```
%% Find Popular Words
% Finds words that appear often with keywo

keyword='word|water,';

% Create binding to Database*

%% Find All Tweets that Contain the Keywor
% Use iterators whenever needed
% Hint: example1 may help with this sectio
% changing the keyword may change your nee
% you account for that if you change the k

% Find how many times the keyword is used to check if an iterator is needed*
```

```
% Create binding to Database*

%% Find All Tweets that Contain the Keyword
% Use iterators whenever needed
% Hint: example1 may help with this section. Also note that
% changing the keyword may change your need for an iterator, so make sure
% you account for that if you change the keyword.

% Find how many times the keyword is used to check if an iterator is needed*
```

# Some Cheat Sheets

- Unix
  - https://wiki.llan.ll.mit.edu/display/LLGRID/Basic+Linux+Commands
- Vi:
  - http://www.viemu.com/a_vi_vim_graphical_cheat_sheet_tutorial.html
  - http://www.atmos.albany.edu/daes/atmclasses/atm350/vi_cheat_sheet.pdf
- Emacs:
  - http://www.rgrjr.com/emacs/emacs_cheat.html
  - http://www.andrew.cmu.edu/course/98-172/labitations/emacs-cheatsheet.pdf
- Nano:
  - http://mintaka.sdsu.edu/reu/nano.html

# Python Exercises Outline

- Basic Examples
  - Connect to Accumulo DB and Display Tables
  - Scan Tweets For a Given Term
  - Search Tweets For a Given Term
  - Search Tweets With Multiple Terms
  - Print Tweets For a Given Term
  - Find Frequent Words That Appear With Given Terms
  - Find Popular Words That Appear With Given Terms
- Bonus Examples
  - Equivalent MATLAB D4M Examples
- Advanced Examples
  - Create and Delete Tables
  - Create a Word Count Table
    - Summing Combiner
    - Batch Writer
  - Find The Top N Words in The Word Count Table

# Thrift Proxy API Basics

- **Create the client for communicating with the Accumulo Thrift Proxy Server**

```
transport = TSocket.TSocket(accumuloProxyServer,
                            accumuloProxyServerPort)
transport = TTransport.TFramedTransport(transport)
protocol  = TCompactProtocol.TCompactProtocol(transport)
client    = AccumuloProxy.Client(protocol)
```

- **Create the login object**

```
transport.open()
login = client.login(accumuloUsername,
                     {'password':accumuloPassword})
```

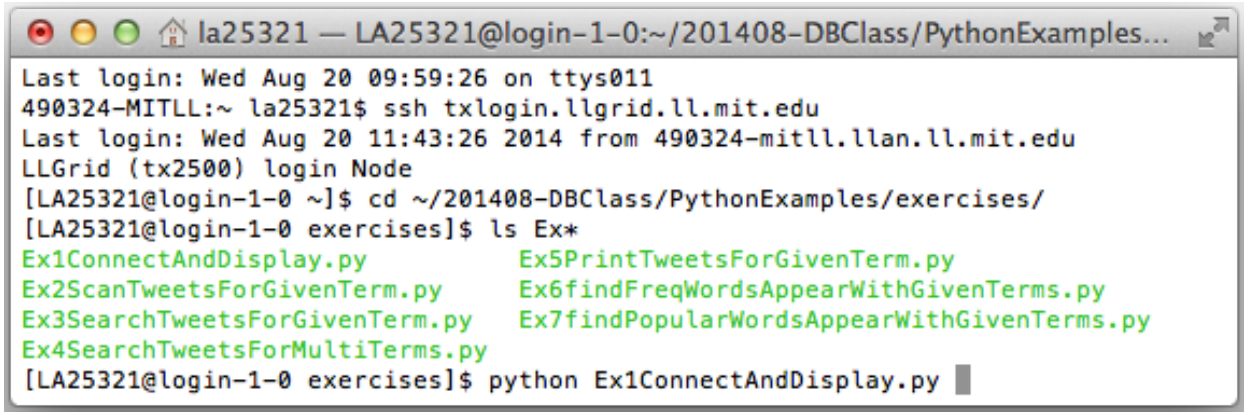
# Python Examples

- While logged in on LLGrid…
- Go to the following directory

  **`$cd ~/201408-DBClass/PythonExamples/exercises`**
- To see all the example names, enter:

  **`$ls Ex*`**
- Open the examples in your editor and follow the instructions
- Then run the example by calling:

  **`$python EXAMPLE_NAME.py`**



# Example 1: Connect and Display

- While logged in on LLGrid…
  Open the example in your editor and follow the instructions.
  Be sure to change 'classdb04' to your database number
  Run the example.

```
$vi Ex1ConnectAndDisplay.py
(or $emacs …, $nano …)
$python Ex1ConnectAndDisplay.py
```

- This example:
  - Connects to Accumulo database specified
  - Displays the tables present in the database

# Ex 1: Connect and Display

```
# Import the utility
from ThriftProxySetupLLGrid import *

# Connect to Accumulo DB using Thrift Proxy API calls
client, login = ThriftProxySetupLLGrid('classdb04')

# Get the list of tables in the Accumulo instance.
tables = client.listTables(login)

if len(tables) > 0:
   print 'Tables present in Accumulo'
   for table in tables:
                print ('    ' + table)
```

# Example 2: Scan Tweets (Slow)

- While logged in on LLGrid…
  - Open the example in your editor and follow the instructions.
    - Be sure to change 'classdb04' to your database number
  - Run the example.

```
$vi Ex2ScanTweetsForGivenTerm.py
(or $emacs …, $nano …)
$python Ex2ScanTweetsForGivenTerm.py
```

- This example:
  - Connects to Accumulo database specified
  - Specifies a keyword
  - Uses Tedge table
  - Scans through each row (tweet) to see if has the given term

# Ex 2: Scan Tweets (Slow)

```
# Import the utility
from ThriftProxySetupLLGrid import *

# Connect to Accumulo DB using Thrift Proxy API calls
client, login = ThriftProxySetupLLGrid('classdb04')

# A word to look for
term =  'FLU'

# Scanning for Tweet IDs for Tweets containing a given term
tweetIds =
scanForTweetIdForSingleWord(client,login,term)
```

# Ex 2: Scan Tweets (Slow, Part 2)

```
def scanForTweetIdForSingleWord(client,login,term):
    # Select the table to scan
    tableToScanForTweetIds = 'geo_tweets_Tedge'
    # Item to store results into
    results = set()

    # Create the scanner options using a column scan for the scanner)
    scanForTweetIdsScannerOptions =
    ScanOptions(columns=[ScanColumn(colFamily='',
                                    colQualifier='word|' + term)])
    # Create the scanner to scan through the table.
    scanForTweetIdsScanner = client.createScanner(login,
              tableToScanForTweetIds, scanForTweetIdsScannerOptions)
    # Scan through all the entries in the table to scan for Tweet IDs.
    while client.hasNext(scanForTweetIdsScanner):
        # Get the next entry in the scanner.
        entry = client.nextEntry(scanForTweetIdsScanner)
        results.add(entry.keyValue.key.row)

    # Return the resulting Tweet IDs
    return results
```

# Example 3: Search for Tweets (Fast)

- While logged in on LLGrid…
  - Open the example in your editor and follow the instructions.
    - Be sure to change 'classdb04' to your database number
  - Run the example.

```
$vi Ex3SearchTweetsForGivenTerm.py
(or $emacs …, $nano …)

$python Ex3SearchTweetsForGivenTerm.py
```

- This example:
  - Connects to Accumulo database specified
  - Specifies a keyword
  - Uses TedgeT table
  - Searches for the row 'word|keyword,' and returns all columns (tweets) with entries in this row

# Ex 3: Search Tweets (Fast)

```
# Import the utility
from ThriftProxySetupLLGrid import *
from searchForTweetIdForSingleWord import *

# Connect to Accumulo DB using Thrift Proxy API calls
client, login = ThriftProxySetupLLGrid('classdb04')

# A word to look for
term =  'FLU'

# Scanning for Tweet IDs for Tweets containing a given
term
tweetIds =
searchForTweetIdForSingleWord(client,login,term)
```

# Ex 3: Search Tweets (Fast, Part 2)

```
def searchForTweetIdForSingleWord(client,login,term):
    # Select the table to scan
    tableToScanForTweetIds = 'geo_tweets_TedgeT'
    # Item to store results into
    results = set()

    # Create the scanner options using a range scan for the scanner)
    scanForTweetIdsScannerOptions =
    ScanOptions(range=Range(start=Key(row='word|'+term),
                            stop=Key(row='word|'+term+'\0')))
    # Create the scanner to scan through the table.
    scanForTweetIdsScanner = client.createScanner(login,
              tableToScanForTweetIds, scanForTweetIdsScannerOptions)
    # Scan through all the rows in the table to search for Tweet IDs.
    while client.hasNext(scanForTweetIdsScanner):
        # Get the next entry in the scanner.
        entry = client.nextEntry(scanForTweetIdsScanner)
        results.add(entry.keyValue.key.colQualifier)

    # Return the resulting Tweet IDs
    return results
```

# Example 4: Search Tweets with Multiple Terms

- **While logged in on LLGrid…**
  - **Open the example in your editor and follow the instructions.**
    - **Be sure to change 'classdb04' to your database number**
  - **Run the example.**

```
$vi Ex4SearchTweetsForMultiTerms.py
(or $emacs …, $nano …)

$python Ex4SearchTweetsForMultiTerms.py
```

- **This example:**
  - **Connects to Accumulo database specified**
  - **Specifies a group of keyword**
  - **Uses TedgeT table**
  - **Searches for the rows for each keyword and returns all columns (tweets) with entries in these rows**

# Ex 4: Search Tweets With Multiple Terms

```
# Import the utility
from ThriftProxySetupLLGrid import *
from searchForTweetIdForSingleWord import *

# Connect to Accumulo DB using Thrift Proxy API calls
client, login = ThriftProxySetupLLGrid('classdb04')

# A set of words to look for
term =  ['FLU', 'Flu', 'flu']

# Search for Tweet IDs for Tweets containing any of the
given terms
tweetIds =
searchForTweetIdForAnyOfGivenWords(client,login,term)
```

# Ex 4: (Continued)

```
def searchForTweetIdForAnyOfGivenWords(client,login,term):
    # Select the table to scan
    tableToScanForTweetIds = 'geo_tweets_TedgeT'
    # Create a structure to store our results in.
    resultSets = dict()
    # Use the method developed in Ex 3 for all the terms
    for term in terms:
        resultSets[term] =
              searchForTweetIdForSingleWord(client,login,term)
    # Find the union of all the result sets
    setCount = 0
    result = set()
    for term in terms:
        if setCount == 0:
            # First term set so we copy the result set.
            result = resultSets.get(term).copy()
            setCount += 1
        else:
            # Not the first term so we do a union of the results.
            result = result.union(resultSets.get(term))
    return result
```

# Example 5: Print Tweets

- While logged in on LLGrid…
  Open the example in your editor and follow the instructions.
  Be sure to change 'classdb04' to your database number
  Run the example.

```
$vi Ex5PrintTweetsForGivenTerm.py
(or $emacs …, $nano …)
$python Ex5PrintTweetsForGivenTerm.py
```

- This example:
  - Connects to Accumulo database specified
  - Specifies a keyword
  - Uses function from Example 3 to find tweets with given keyword
  - Prints the full text of these tweets (from TedgeTxt)

# Ex 5: Print Tweets

```
# Import the utility
from ThriftProxySetupLLGrid import *
from printTweetsContainingSingleWord import *

# Connect to Accumulo DB using Thrift Proxy API calls
client, login = ThriftProxySetupLLGrid('classdb04')

# A word to look for
term =  'FLU'

# Print the Tweets that contain the single term.
printTweetsContainingSingleWord(client,login,term)
```

# Ex 5: (Continued)

```
def printTweetsContainingSingleWord(client,login,term):
    # Select the table to get the tweet texts
    tableTextTable = 'geo_tweets_TedgeText'

    # Get the tweet IDs for the term using the method coded from Ex 3.
    tweetIds = searchForTweetIdForSingleWord(client, login, term)

    for tweetId in tweetIds:
        # Create the scanner options
        scannerOptions =
                    scanOptions(range=Range(start=Key(row=tweetId),
                                            stop=Key(row=tweetId+'\0')))
        # Create the scanner to scan through the Tweet text table
        scanner = client.createScanner(login, tweetTextTable,
                                               scannerOptions)
        # Print the Tweet text from the entry
        entry     = client.nextEntry(scanner)
        tweetText = entry.keyValue.value
        # print the tweet text
        print '    ' + tweetText
```

# Example 6: Find Frequent Words

- **While logged in on LLGrid…**
  - **Open the example in your editor and follow the instructions.**
    - **Be sure to change 'classdb04' to your database number**
  - **Run the example.**

```
$vi Ex6findFreqWordsAppearWithGivenTerms.py
(or $emacs …, $nano …)

$python Ex6findFreqWordsAppearWithGivenTerms.py
```

- **This example:**
  - **Connects to Accumulo database specified**
  - **Specifies a group of keywords**
  - **Uses function from Example 4 to find tweets with given keywords**
  - **Sums number of times each other word appears with keywords in tweets and prints those used more than 5 times**

# Ex 6: Find Frequent Words

```
# Import the utility
from ThriftProxySetupLLGrid import *
from printFreqWordsAppearWithAnyOfGivenWords import *
# Connect to Accumulo DB using Thrift Proxy API calls
client, login = ThriftProxySetupLLGrid('classdb04')

# A set of words to look for
term =  ['FLU', 'Flu', 'flu']
# Find frequent words that appear with the given terms.
frequency = 5
words = printFreqWordsAppearWithAnyOfGivenWords(client,
                                    login, terms, frequency);
# Print the results.
for key, value in words.iteritems():
    print ' '+key+' appeared '+str(value)+' times'
```

# Ex 6: (Continued)

```
def printFreqWordsAppearWithAnyOfGivenWords(client,login,terms,
                                            frequency):
    # Get the Tweet IDs using the Ex 4 method for multiple terms.
    tweetIds = searchForTweetIdForAnyOfGivenWords(client, login, terms)
    tweetTable = 'geo_tweets_Tedge'
    results = set()
    occurences = collections.defaultdict(int)
    for tweetId in tweetIds:
        scannerOptions = scanOptions(range=Range(start=Key(row=tweetId),
                                                 stop=Key(row=tweetId+'\0')))
        scanner = client.createScanner(login, tweetTable, scannerOptions)
        while client.hasNext(scanner):
            entry = client.nextEntry(scanner)
            # Store only the word entry with its frequency
            if re.search('^word\|',entry.keyValue.key.colQualifier):
               results.add(entry.keyValue.key.colQualifier)
               if occurences.has_key(entry.keyValue.key.colQualifier):
                  occurences[entry.keyValue.key.colQualifier] += 1
               else:
                  occurences[entry.keyValue.key.colQualifier] = 1
    # Find all the words that appear frequently with the terms
    freqWords = dict((k,v) for k,v in occurences.items()
                                    if v > frequency)
    return freqWords
```

# Example 7: Find Popular Words

- While logged in on LLGrid…
  - Open the example in your editor and follow the instructions.
    - Be sure to change 'classdb04' to your database number
  - Run the example.

```
$vi Ex7findPopularWordsAppearWithGivenTerms.py
(or $emacs …, $nano …)
$python Ex7findPopularWordsAppearWithGivenTerms.py
```

- This example:
  - Connects to Accumulo database specified
  - Specifies a group of keywords
  - Uses function from Example 6 to find frequent words for given keywords
  - Checks TedgeDeg table and returns those with less than 1000 total uses

# Ex 7: Find Popular Words

```
# Import the utility
from ThriftProxySetupLLGrid import *
from printPopularWordsAppearWithAnyOfGivenWords import *
# Connect to Accumulo DB using Thrift Proxy API calls
client, login = ThriftProxySetupLLGrid('classdb04')

# A set of words to look for
term =  ['FLU', 'Flu', 'flu']
# Find frequent words that appear with the given terms.
frequency = 5
words =
     printPopularWordsAppearWithAnyOfGivenWords(client,
                                  login, terms, frequency);
# Print the results.
for key, value in words.iteritems():
    print ' '+key+' appeared '+str(value)+' times'
```

# Ex 7: (Continued)

```
def printPopularWordsAppearWithAnyOfGivenWords(client,login,terms,
                                                frequency):
    # Find frequent words that appear with the given terms.
    freqWords = printFreqWordsAppearWithAnyOfGivenWords(client, login,
                                                        terms, frequency)

    # Look up how popular those frequent words are
    wordKeys = []
    for key,value in freqWords.items():
        wordKeys.append(key)

    # Look up popularity for the terms.
    thresholdFrequency = 10000
    popularWords = searchForPopularityForAnyGivenWords(client, login,
                                                       wordKeys, thresholdFrequency)

    # Return the resulting words
    return popularWords
```

# Ex 7: (Continued, 2)

```
def searchForPopularityForAnyGivenWords(client, login, wordKeys,
                                        thresholdFrequency):
    tableToScan = 'geo_tweets_TedgeDeg'
    results = dict()
    for wordKey in wordKeys:
        # Scanner options using a range scan for the scanner).
        scanForTweetIdsScannerOptions =
                    ScanOptions(range=Range(start=Key(row=wordKey),
                                            stop=Key(row=wordKey+'\0')))
        # Create the scanner to scan through the table.
        scanForTweetIdsScanner = client.createScanner(login,
                            tableToScan, scanForTweetIdsScannerOptions)

        while client.hasNext(scanForTweetIdsScanner):
                # Get the next entry in the scanner.
                entry = client.nextEntry(scanForTweetIdsScanner)
                # store only if it appears < thresholdFrequency
                if int(entry.keyValue.value) < thresholdFrequency:
                    results[wordKey] = entry.keyValue.value
    return results
```

# Conclusions

- Many ways to connect to Accumulo instances
- Enjoy the extra exercises
- Feel free to contact grid-help@ll.mit.edu if you have any questions.