Pivotal

Machine Learning, Graph, Text and Geospatial on PostgreSQL and Greenplum

Frank McQuillan
Bharath Sitaraman

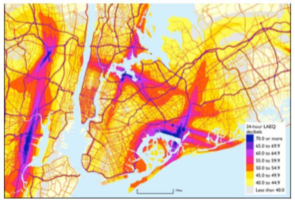
Greenplum Integrated Analytics

Data Transformation

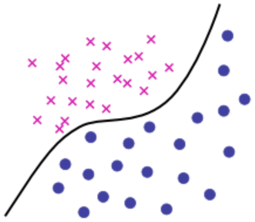
Text

Traditional BI

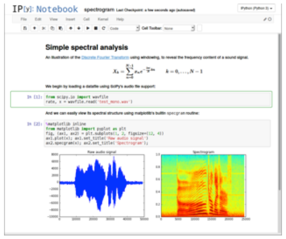
Geospatial



Machine Learning



Data Science Productivity Tools



spectrogram Last Checkpoint: a few seconds ago (autosaved)

Simple spectral analysis

An illustration of the Discrete Fourier Transform using windowing, to reveal the frequency content of a sound signal.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

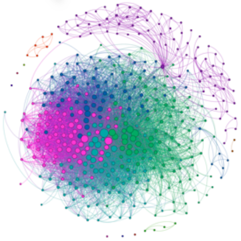
We begin by loading a datafile using SciPy's audio file support:

```
In [1]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin specgram routine:

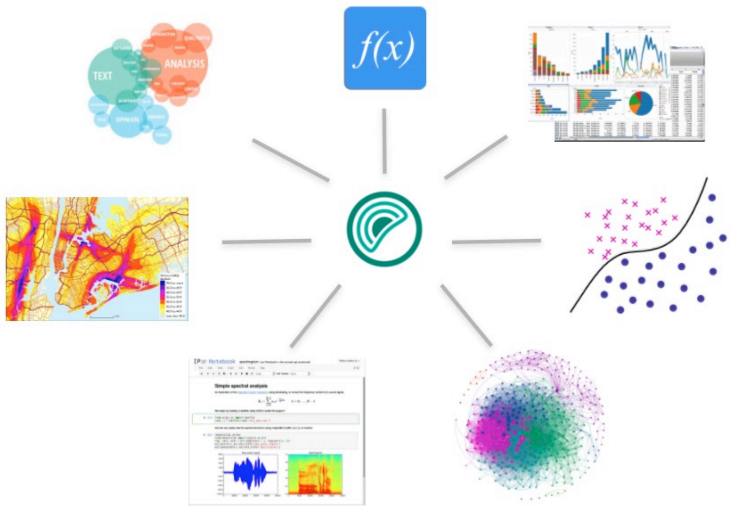
```
In [2]: %matplotlib inline
        from matplotlib import pyplot as plt
        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```

Graph

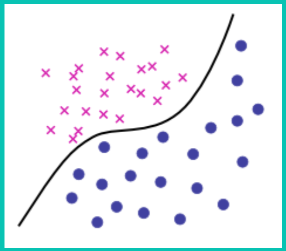


Agenda

1. Machine learning with Apache MADlib
2. Data transformation
3. Graph
4. Data science productivity tools
5. Geospatial with PostGIS
6. Text analytics with GPText
7. Connectivity
8. Example use cases
9. Looking ahead



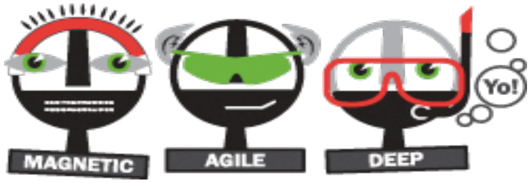
1. Machine Learning with Apache MADlib



MADlib

Pivotal

Scalable, In-Database Machine Learning



Apache MADlib: Big Data Machine Learning in SQL

Open source, top level Apache project

For PostgreSQL and Greenplum Database

Powerful machine learning, graph, statistics and analytics for data scientists

- Open source https://github.com/apache/madlib
- Downloads and docs http://madlib.apache.org/
- Wiki https://cwiki.apache.org/confluence/display/MADLIB/

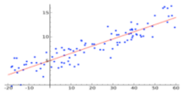
History

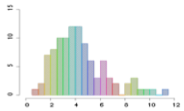
MADlib project was initiated in 2011 by EMC/Greenplum architects and Professor Joe Hellerstein from University of California, Berkeley.

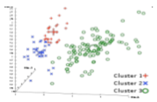
UrbanDictionary.com:
mad *(adj.): an adjective used to enhance a noun.*

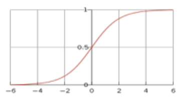
1- dude, you got skills.
2- dude, you got ***mad*** *skills.*

MADlib Functions









Supervised Learning
Neural Networks
Support Vector Machines (SVM)
Regression Models
- Clustered Variance
- Cox-Proportional Hazards Regression
- Elastic Net Regularization
- Generalized Linear Models
- Linear Regression
- Logistic Regression
- Marginal Effects
- Multinomial Regression
- Naïve Bayes
- Ordinal Regression
- Robust Variance

Tree Methods
- Decision Tree
- Random Forest

Conditional Random Field (CRF)

Unsupervised Learning
Association Rules (Apriori)
Clustering (k-Means)
Topic Modelling (Latent Dirichlet Allocation)

Nearest Neighbors
- k-Nearest Neighbors

Graph
All Pairs Shortest Path (APSP)
Breadth-First Search
Hyperlink-Induced Topic Search (HITS)
Average Path Length
Closeness Centrality
Graph Diameter
In-Out Degree

PMML Export
Sampling
- Balanced
- Random
- Stratified

Sessionize
Term Frequency for Text Analysis

Time Series Analysis
- ARIMA

Data Types and Transformations
Array and Matrix Operations
Matrix Factorization
- Low Rank
- Singular Value Decomposition (SVD)

Norms and Distance Functions
Sparse Vectors
Principal Component Analysis (PCA)
g Categorical Variables
g

s
ve Statistics
inality Estimators
elation and Covariance
mary
l Statistics
- Hypothesis Tests

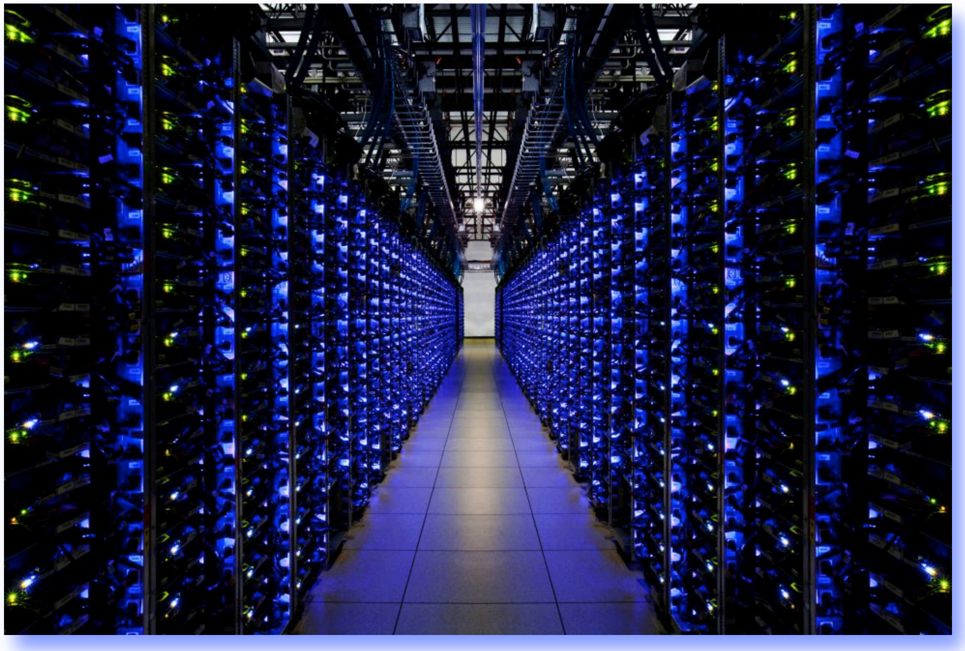
Probability Functions

Model Selection
Cross Validation
Prediction Metrics
Train-Test Split

Comprehensive and mature data science library

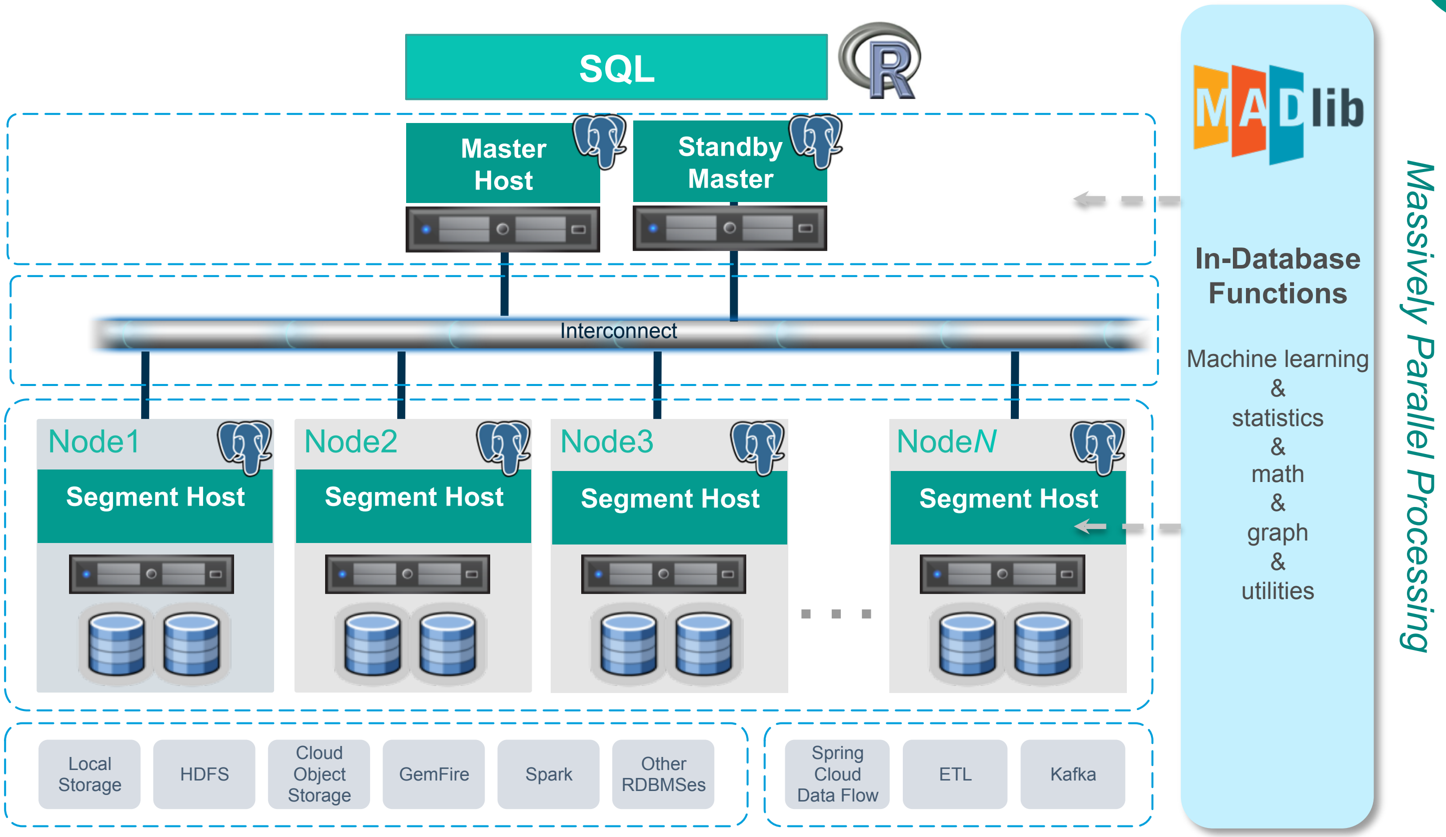
Why MADlib on Greenplum?

- Better parallelism
- Better scalability
- Higher predictive accuracy
- Top level ASF project



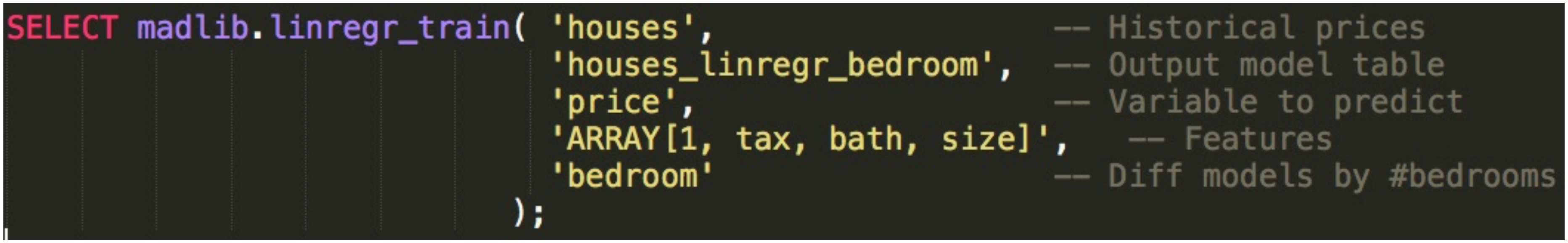
“Apache MADlib Comes of Age”, Frank McQuillan, Oct. 2017, https://content.pivotal.io/blog/apache-madlib-comes-of-age

Greenplum Database with MADlib



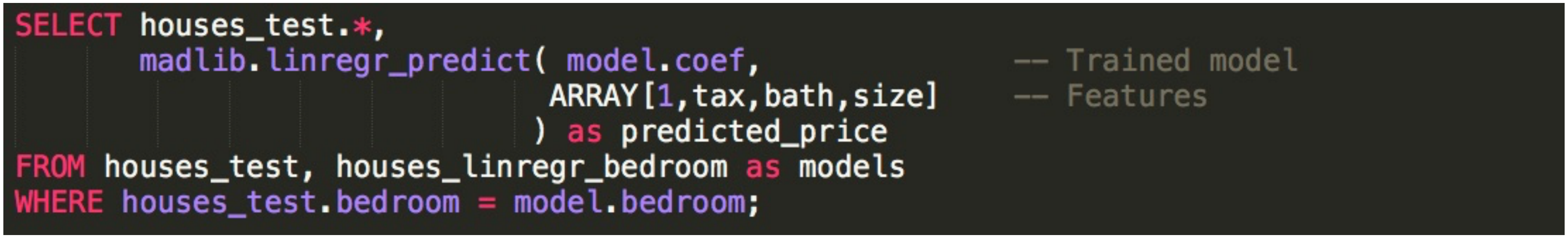
Familiar SQL Interface

Train (build a predictive model)



```
SELECT madlib.linregr_train( 'houses',                     -- Historical prices
                             'houses_linregr_bedroom',     -- Output model table
                             'price',                      -- Variable to predict
                             'ARRAY[1, tax, bath, size]',    -- Features
                             'bedroom'                     -- Diff models by #bedrooms
                           );
```

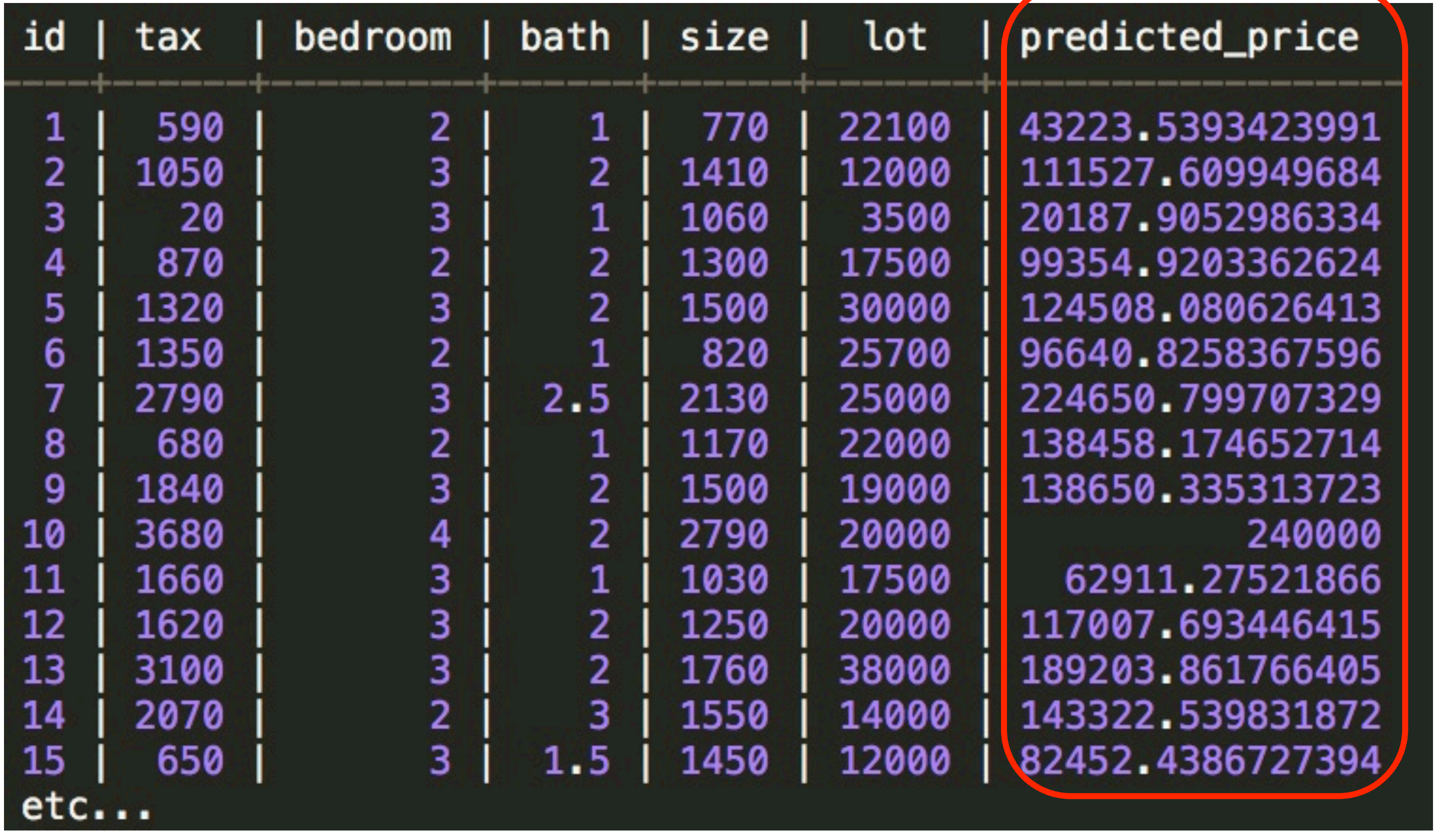
Predict (use model on new data)



```
SELECT houses_test.*,
       madlib.linregr_predict( model.coef,               -- Trained model
                               ARRAY[1,tax,bath,size]    -- Features
                             ) as predicted_price
FROM houses_test, houses_linregr_bedroom as models
WHERE houses_test.bedroom = model.bedroom;
```

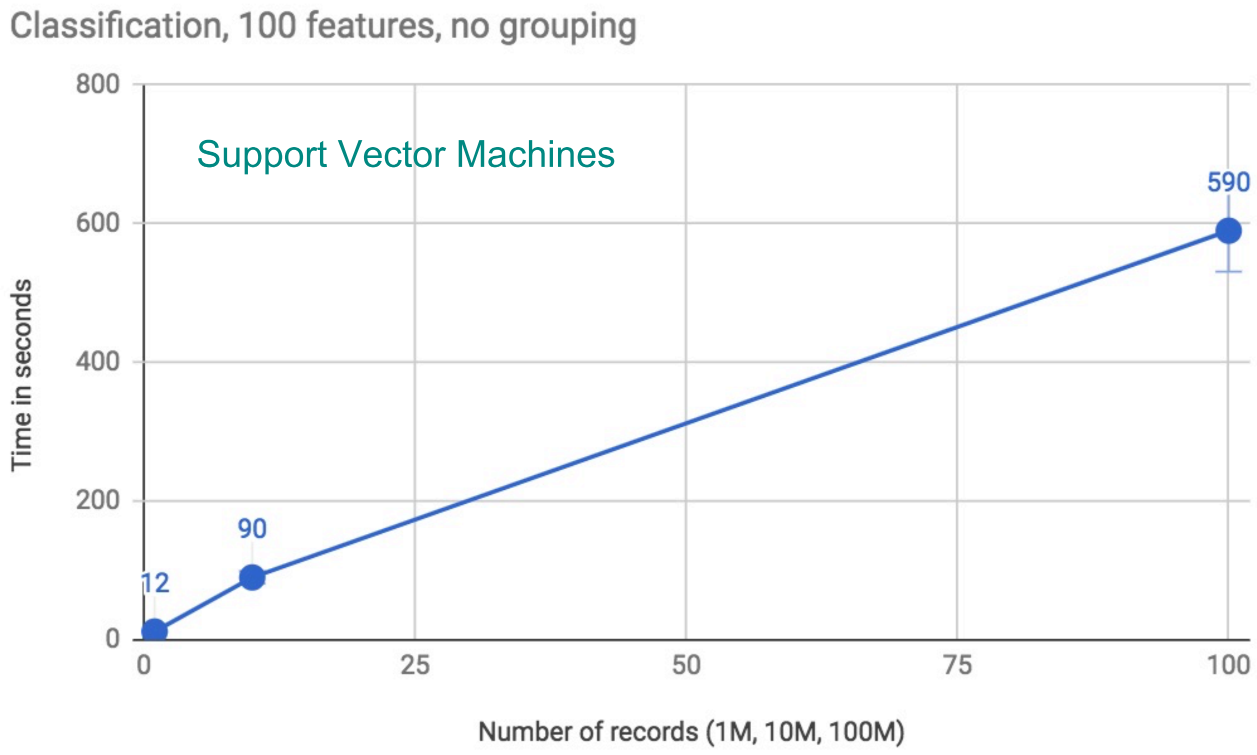
Familiar SQL Interface

From house pricing model



```
 id | tax  | bedroom | bath | size |  lot  | predicted_price
----+------+---------+------+------+-------+------------------
  1 |  590 |       2 |    1 |  770 | 22100 | 43223.5393423991
  2 | 1050 |       3 |    2 | 1410 | 12000 | 111527.609949684
  3 |   20 |       3 |    1 | 1060 |  3500 | 20187.9052986334
  4 |  870 |       2 |    2 | 1300 | 17500 | 99354.9203362624
  5 | 1320 |       3 |    2 | 1500 | 30000 | 124508.080626413
  6 | 1350 |       2 |    1 |  820 | 25700 | 96640.8258367596
  7 | 2790 |       3 |  2.5 | 2130 | 25000 | 224650.799707329
  8 |  680 |       2 |    1 | 1170 | 22000 | 138458.174652714
  9 | 1840 |       3 |    2 | 1500 | 19000 | 138650.335313723
 10 | 3680 |       4 |    2 | 2790 | 20000 |           240000
 11 | 1660 |       3 |    1 | 1030 | 17500 |   62911.27521866
 12 | 1620 |       3 |    2 | 1250 | 20000 | 117007.693446415
 13 | 3100 |       3 |    2 | 1760 | 38000 | 189203.861766405
 14 | 2070 |       2 |    3 | 1550 | 14000 | 143322.539831872
 15 |  650 |       3 |  1.5 | 1450 | 12000 | 82452.4386727394
etc...
```

Built to Scale



Greenplum cluster:

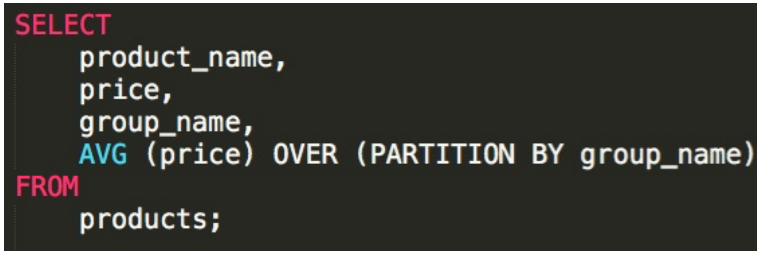
- 1 master
- 4 segment hosts with 6 segments per host

Pivotal

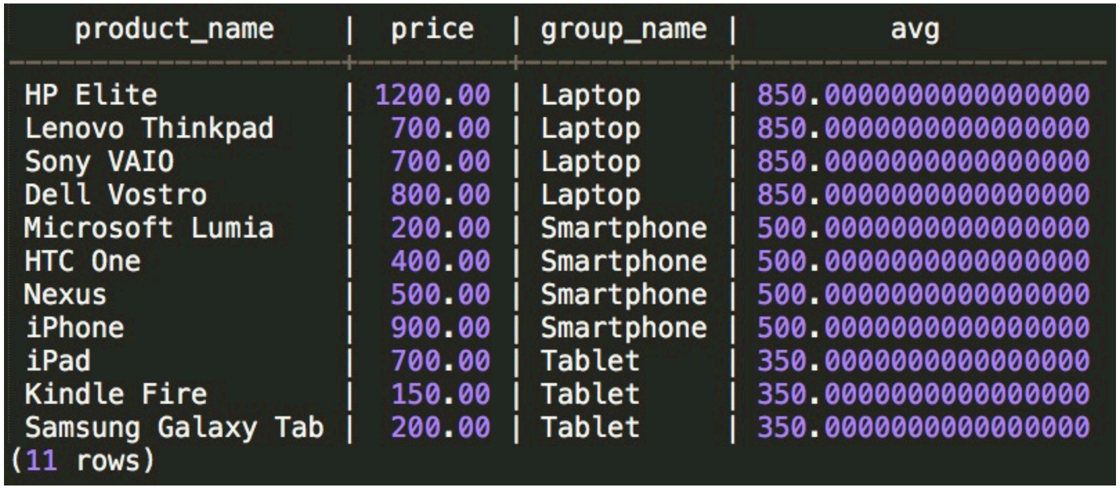
2. Data Transformation

Native PostgreSQL Data Transformations

- Rich library of functions and operators
  - Array functions
  - Aggregate functions
  - Window functions



```
SELECT
    product_name,
    price,
    group_name,
    AVG (price) OVER (PARTITION BY group_name)
FROM
    products;
```



```
   product_name       | price  | group_name |         avg
----------------------+--------+------------+----------------------
 HP Elite             | 1200.00 | Laptop     | 850.0000000000000000
 Lenovo Thinkpad      |  700.00 | Laptop     | 850.0000000000000000
 Sony VAIO            |  700.00 | Laptop     | 850.0000000000000000
 Dell Vostro          |  800.00 | Laptop     | 850.0000000000000000
 Microsoft Lumia      |  200.00 | Smartphone | 500.0000000000000000
 HTC One              |  400.00 | Smartphone | 500.0000000000000000
 Nexus                |  500.00 | Smartphone | 500.0000000000000000
 iPhone               |  900.00 | Smartphone | 500.0000000000000000
 iPad                 |  700.00 | Tablet     | 350.0000000000000000
 Kindle Fire          |  150.00 | Tablet     | 350.0000000000000000
 Samsung Galaxy Tab   |  200.00 | Tablet     | 350.0000000000000000
(11 rows)
```

“Comparing Window Function Features by Database Vendors”, Jiri Mauritz, Sonra Intelligence, Sept. 15, 2017

MADlib Data Transformations

Array and Matrix Operations
Conjugate Gradient
Encoding Categorical Variables
Linear Solvers

- Dense Linear Systems
- Sparse Linear Systems

Matrix Factorization

- Low Rank
- Singular Value Decomposition (SVD)

Norms and Distance Functions
Path

Pivot
PMML Export
Principal Component Analysis (PCA)
Sampling

- Balanced
- Random
- Stratified

Sessionize
Sparse Vectors
Stemming
Term Frequency for Text Analysis

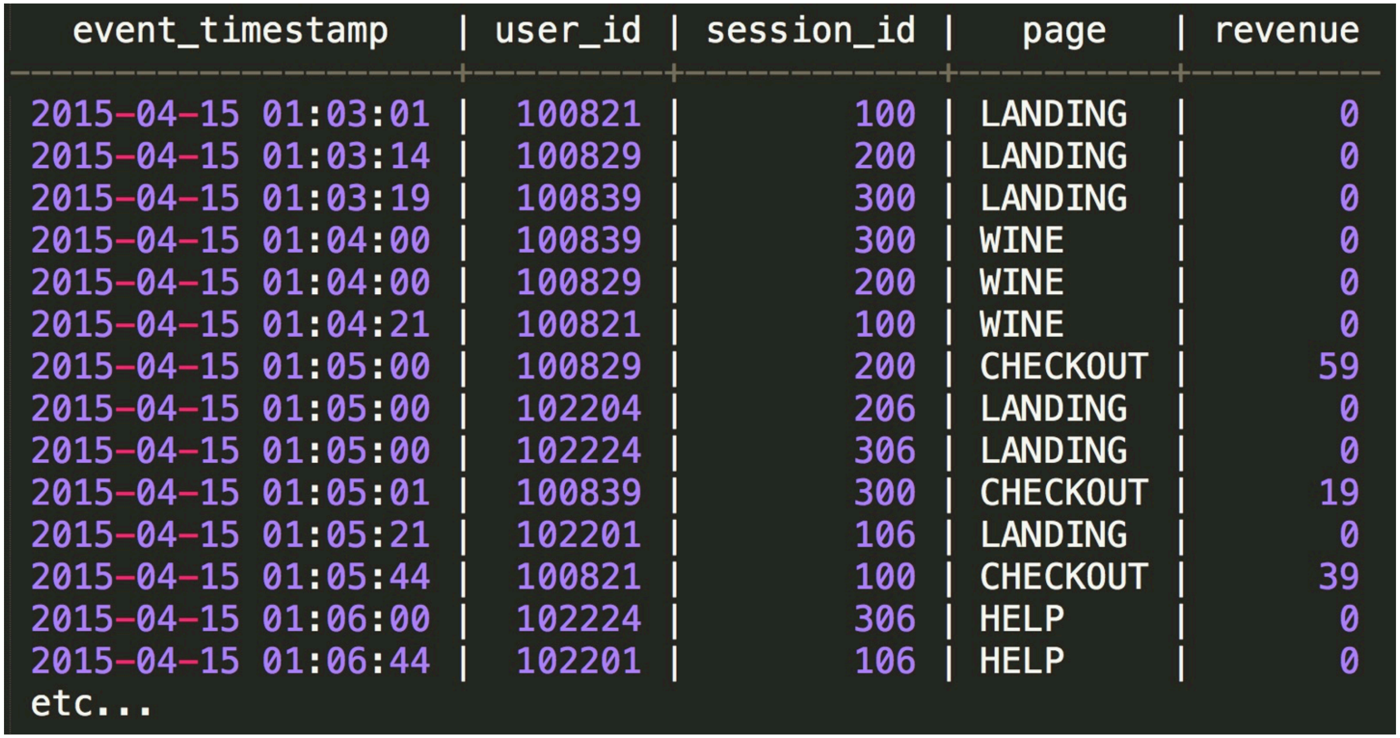
"New Tools To Shape Data In Apache MADlib", Frank McQuillan, Sept 2016, https://content.pivotal.io/blog/new-tools-to-shape-data-in-apache-madlib

Path Functions in E-commerce

?

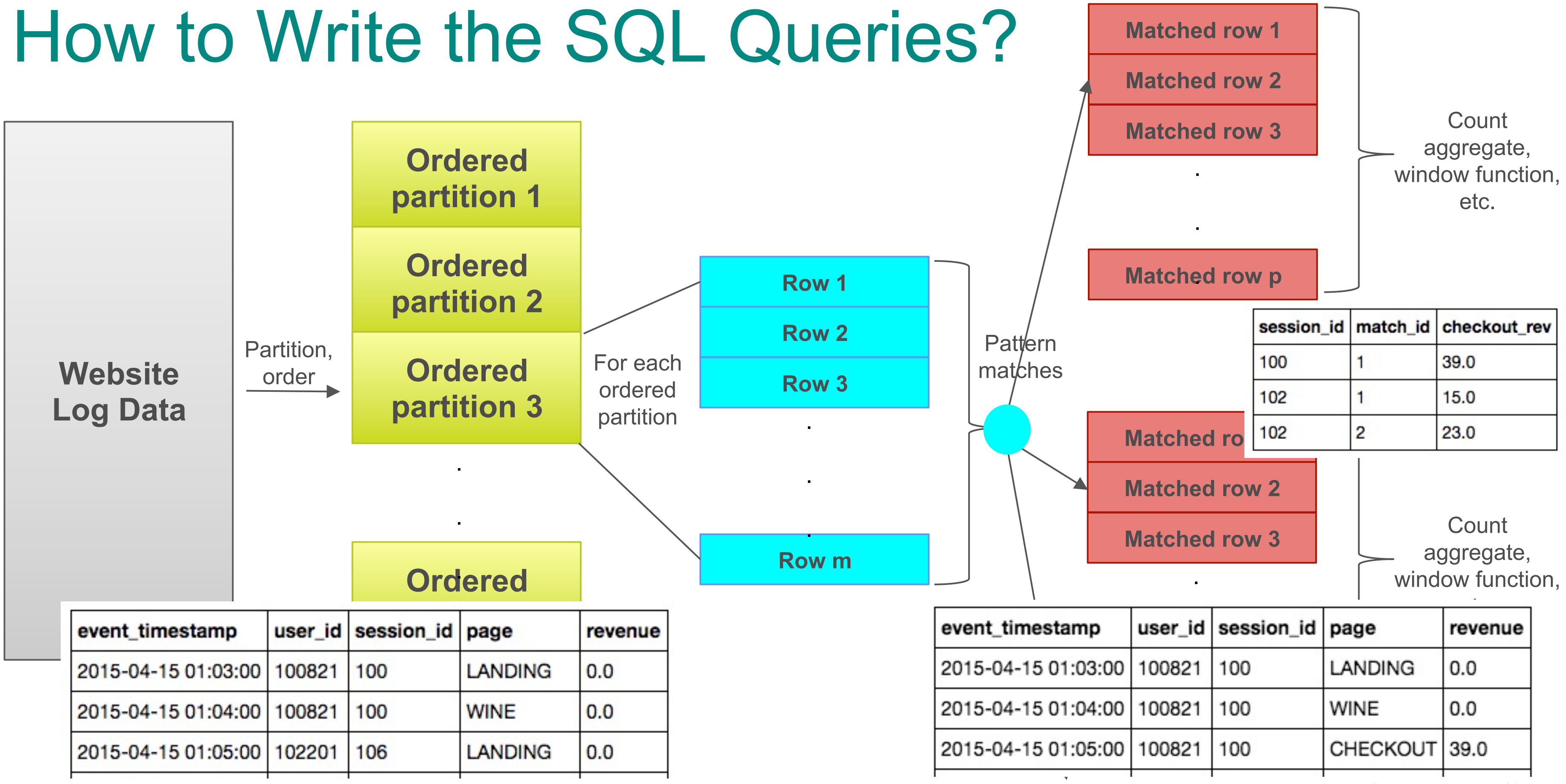
?

Raw Data from Website Logs



```
    event_timestamp     | user_id | session_id |    page    | revenue
------------------------+---------+------------+------------+---------
 2015-04-15 01:03:01 |  100821 |         100 | LANDING    |       0
 2015-04-15 01:03:14 |  100829 |         200 | LANDING    |       0
 2015-04-15 01:03:19 |  100839 |         300 | LANDING    |       0
 2015-04-15 01:04:00 |  100839 |         300 | WINE       |       0
 2015-04-15 01:04:00 |  100829 |         200 | WINE       |       0
 2015-04-15 01:04:21 |  100821 |         100 | WINE       |       0
 2015-04-15 01:05:00 |  100829 |         200 | CHECKOUT |      59
 2015-04-15 01:05:00 |  102204 |         206 | LANDING    |       0
 2015-04-15 01:05:00 |  102224 |         306 | LANDING    |       0
 2015-04-15 01:05:01 |  100839 |         300 | CHECKOUT |      19
 2015-04-15 01:05:21 |  102201 |         106 | LANDING    |       0
 2015-04-15 01:05:44 |  100821 |         100 | CHECKOUT |      39
 2015-04-15 01:06:00 |  102224 |         306 | HELP       |       0
 2015-04-15 01:06:44 |  102201 |         106 | HELP       |       0
 etc...
```

How to Write the SQL Queries?

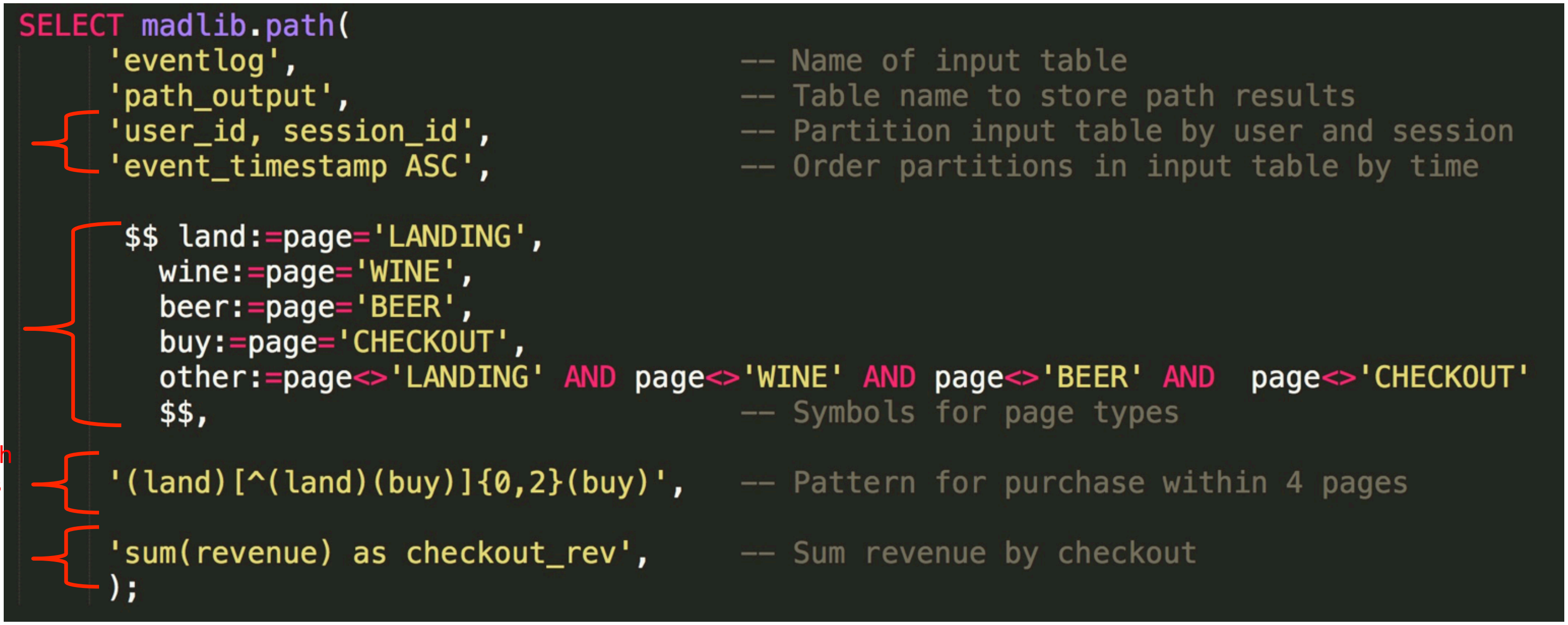


session_id	match_id	checkout_rev
100	1	39.0
102	1	15.0
102	2	23.0

event_timestamp	user_id	session_id	page	revenue
2015-04-15 01:03:00	100821	100	LANDING	0.0
2015-04-15 01:04:00	100821	100	WINE	0.0
2015-04-15 01:05:00	102201	106	LANDING	0.0

event_timestamp	user_id	session_id	page	revenue
2015-04-15 01:03:00	100821	100	LANDING	0.0
2015-04-15 01:04:00	100821	100	WINE	0.0
2015-04-15 01:05:00	100821	100	CHECKOUT	39.0

MADlib Path Functions



Partition and order

Define symbols

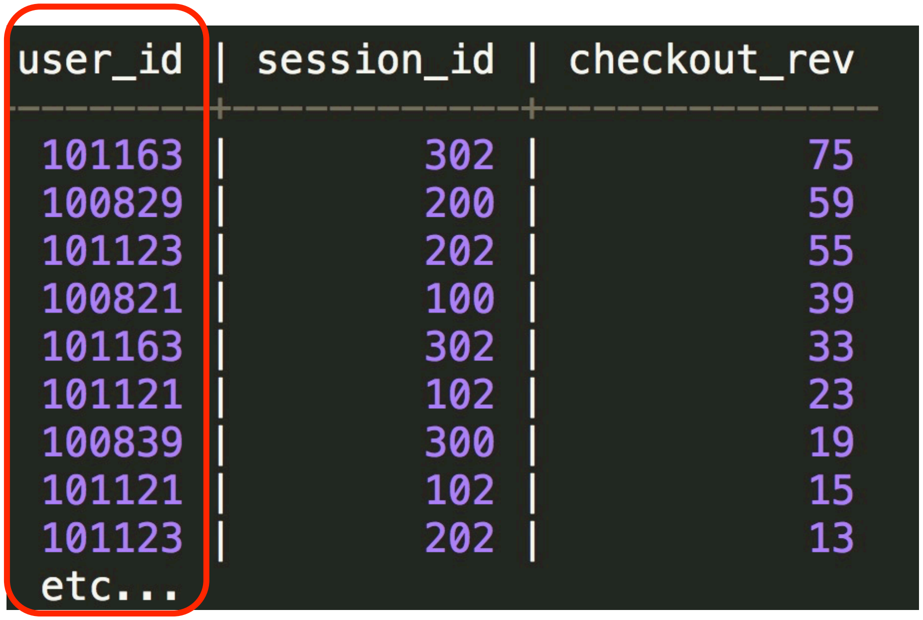
Pattern match across rows

Sum revenue

```
SELECT madlib.path(
     'eventlog',                         -- Name of input table
     'path_output',                      -- Table name to store path results
     'user_id, session_id',              -- Partition input table by user and session
     'event_timestamp ASC',              -- Order partitions in input table by time

      $$ land:=page='LANDING',
        wine:=page='WINE',
        beer:=page='BEER',
        buy:=page='CHECKOUT',
        other:=page<>'LANDING' AND page<>'WINE' AND page<>'BEER' AND  page<>'CHECKOUT'
        $$,                              -- Symbols for page types

     '(land)[^(land)(buy)]{0,2}(buy)',   -- Pattern for purchase within 4 pages

     'sum(revenue) as checkout_rev',     -- Sum revenue by checkout
     );
```

“Path Functions in Apache MADlib”, Frank McQuillan, May 2016, https://content.pivotal.io/blog/path-functions-in-apache-madlib

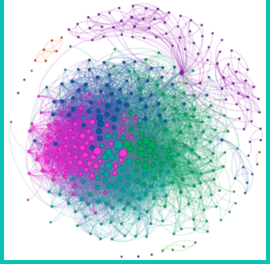
High Value Quick Shoppers



```
user_id | session_id | checkout_rev
--------+------------+--------------
 101163 |        302 |           75
 100829 |        200 |           59
 101123 |        202 |           55
 100821 |        100 |           39
 101163 |        302 |           33
 101121 |        102 |           23
 100839 |        300 |           19
 101121 |        102 |           15
 101123 |        202 |           13
 etc...
```

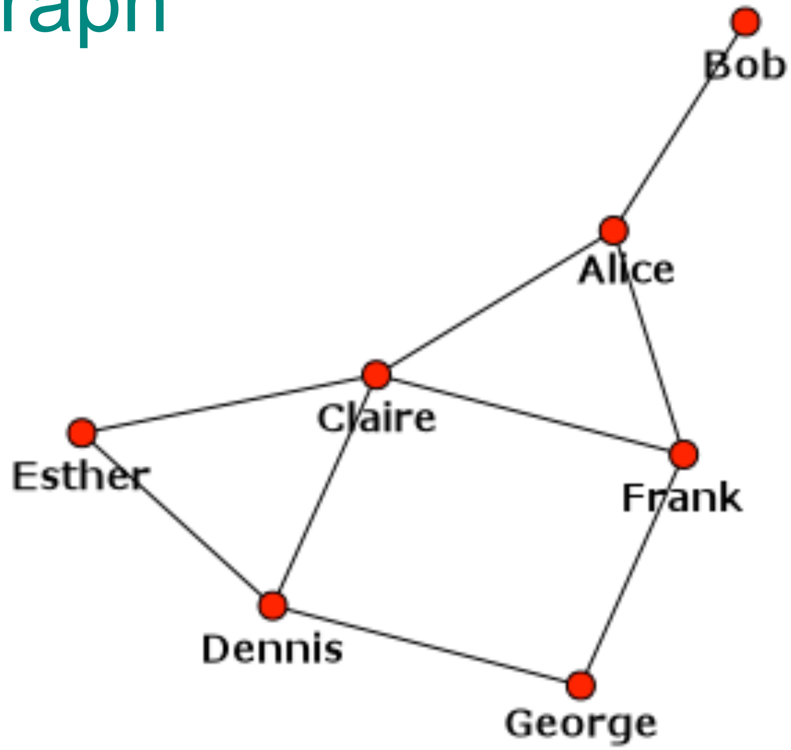
Sorted descending by revenue

3. Graph

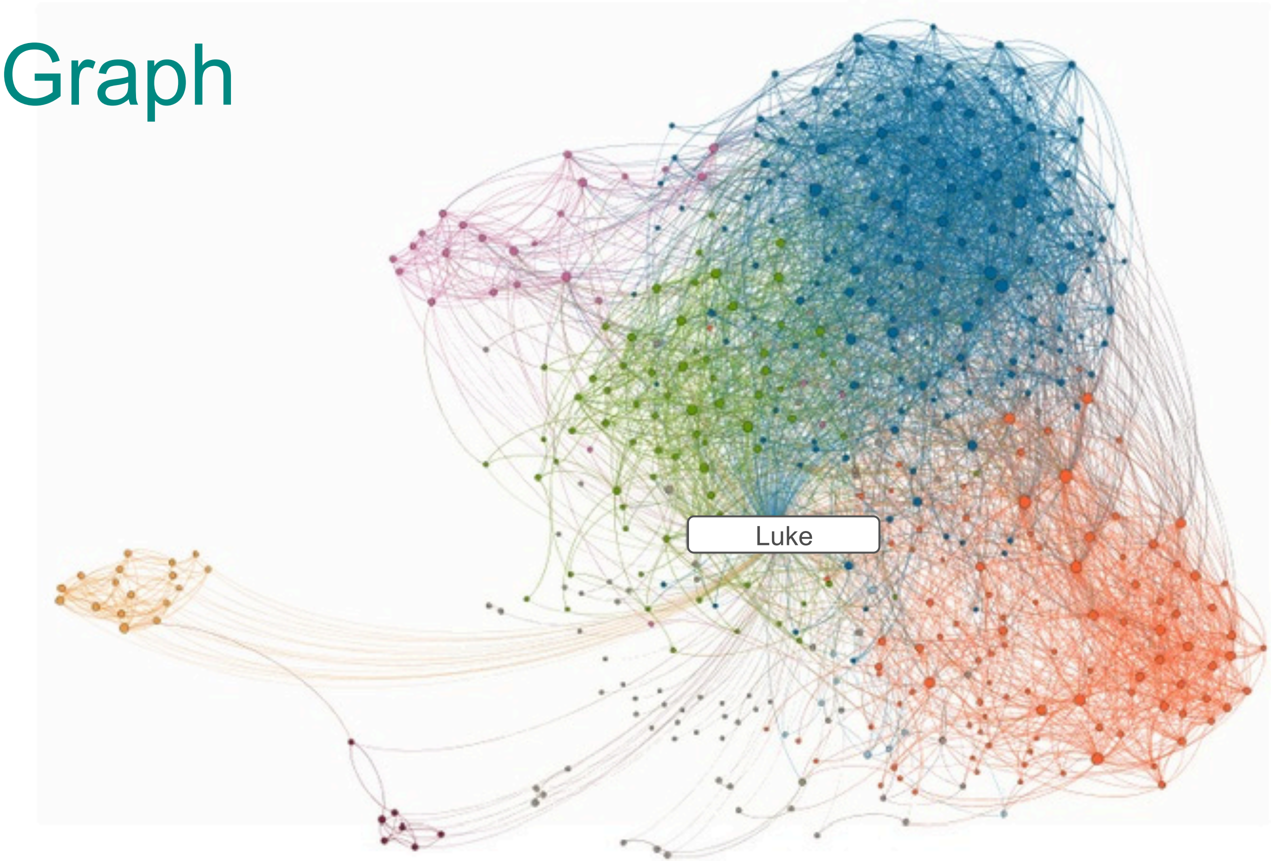


MPP databases are an effective tool for *graph analytics at scale in enterprise*

A Small Graph

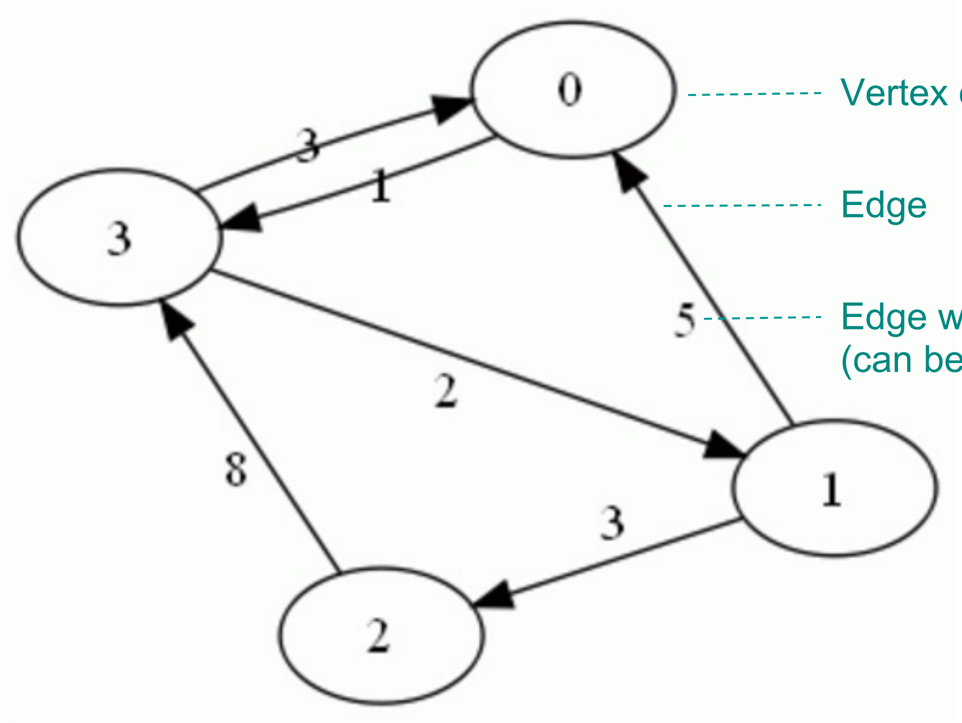


A Big Graph



Sample LinkedIn social graph

Directed Graph



Vertex or node

Edge

Edge weight
(can be negative)

Graph Representation in MADlib

Vertex Table

Vertex	Vertex Params
0	...
1	...
2	...
3	...

. . .

⋮

Edge Table

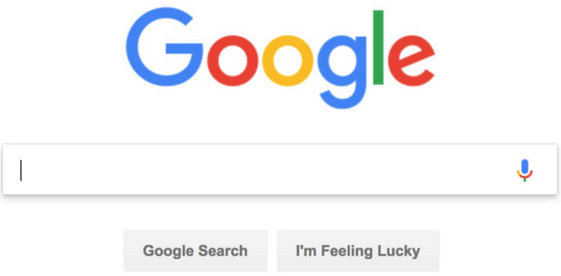
Source Vertex	Dest Vertex	Edge Weight	Edge Params
0	3	1.0	...
1	0	5.0	...
1	2	3.0	...
2	3	8.0	...
3	0	3.0	...
3	1	2.0	...

. . .

⋮

PageRank

- Web search
- Scientific impact of researchers
- Street and space usage
- Neuroscience



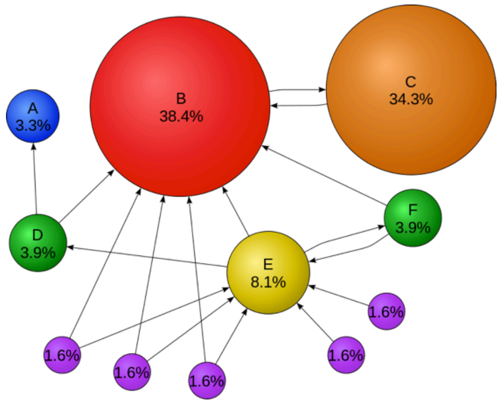
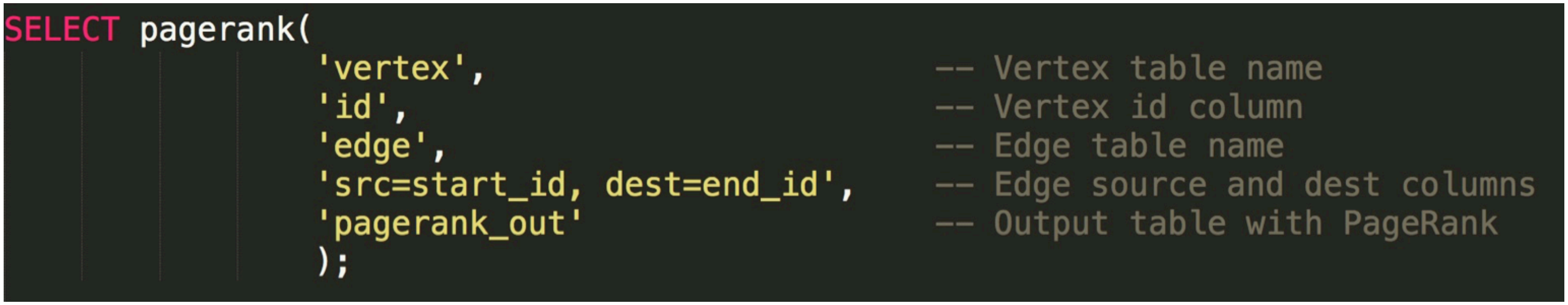
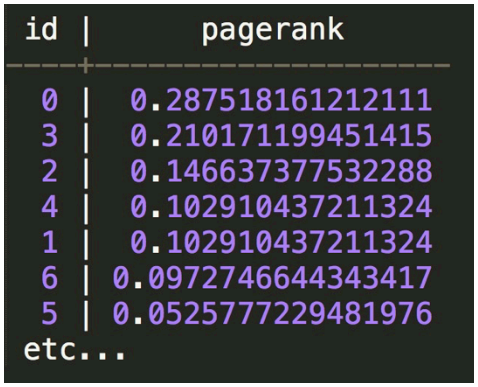


Image from
https://en.wikipedia.org/wiki/PageRank

PageRank in MADlib

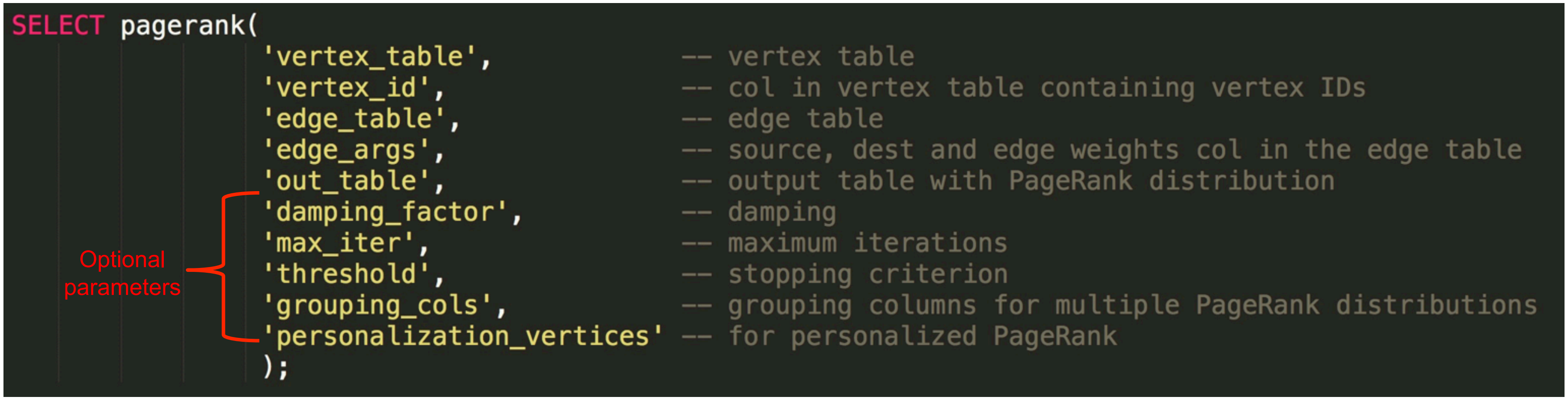


```
SELECT pagerank(
                'vertex',                           -- Vertex table name
                'id',                               -- Vertex id column
                'edge',                             -- Edge table name
                'src=start_id, dest=end_id',        -- Edge source and dest columns
                'pagerank_out'                      -- Output table with PageRank
                );
```



```
 id |      pagerank
----+--------------------
  0 |   0.287518161212111
  3 |   0.210171199451415
  2 |   0.146637377532288
  4 |   0.102910437211324
  1 |   0.102910437211324
  6 | 0.0972746644343417
  5 | 0.0525777229481976
etc...
```

PageRank in MADlib

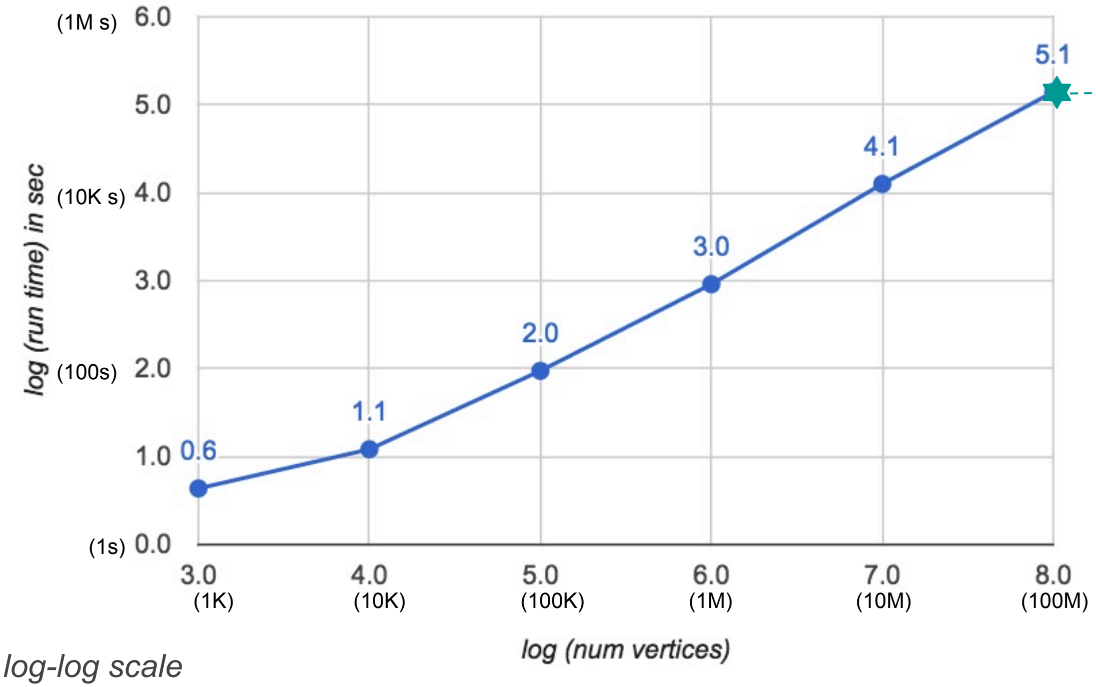


```
SELECT pagerank(
                'vertex_table',               -- vertex table
                'vertex_id',                  -- col in vertex table containing vertex IDs
                'edge_table',                 -- edge table
                'edge_args',                  -- source, dest and edge weights col in the edge table
                'out_table',                  -- output table with PageRank distribution
                'damping_factor',             -- damping
                'max_iter',                   -- maximum iterations
                'threshold',                  -- stopping criterion
                'grouping_cols',              -- grouping columns for multiple PageRank distributions
                'personalization_vertices'    -- for personalized PageRank
                );
```

“Graph Processing on Greenplum Database using Apache MADlib”, Frank McQuillan, Jan 2018, https://content.pivotal.io/blog/graph-processing-on-greenplum-database-using-apache-madlib

PageRank Performance on Greenplum

Mean edge degrees per node = 50



5B edges

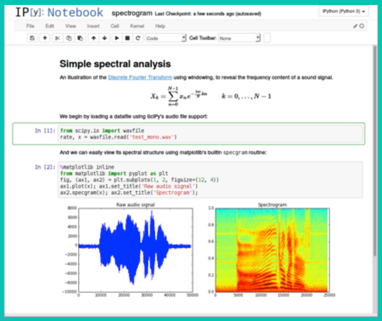
Normal random graphs with
mean degrees 50 edges per vertex
(i.e., 5B edges in the largest case)

Greenplum cluster:

- 1 master
- 4 segment hosts with 6 segments per host

Note: log-log scale

4. Data Science Productivity Tools



Simple spectral analysis

An illustration of the Discrete Fourier Transform using windowing, to reveal the frequency content of a sound signal.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

We begin by loading a datafile using SciPy's audio file support:

```
In [1]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin specgram routine:

```
In [2]: %matplotlib inline
        from matplotlib import pyplot as plt
        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```

PivotalR

- Familiar R interface + performance/scalability of in-database analytics

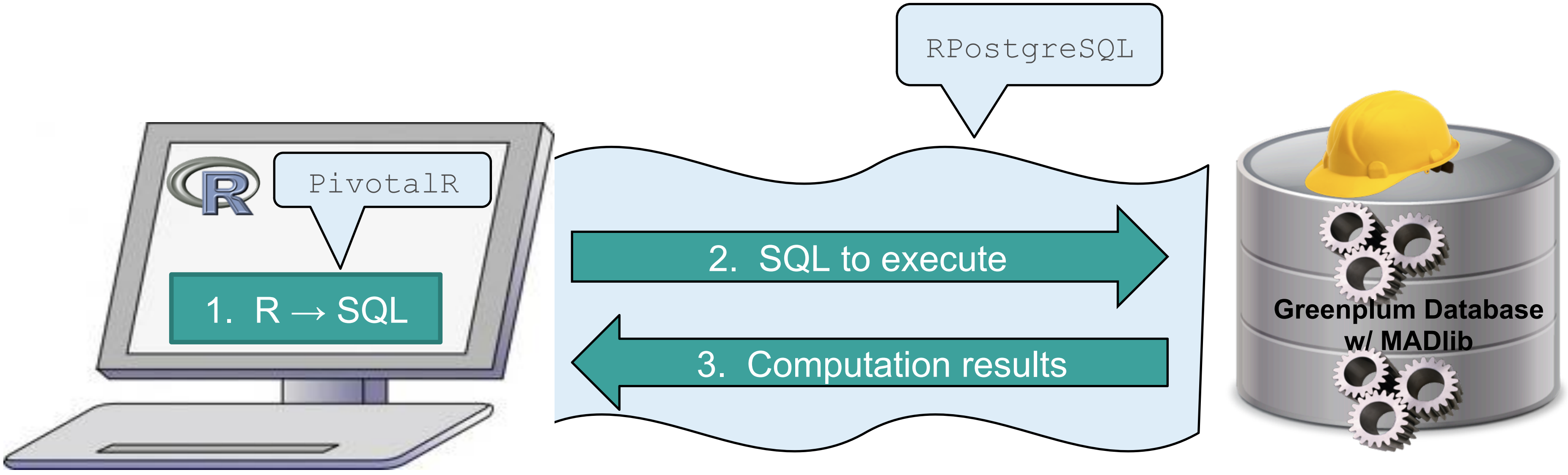
PivotalR

```
d <- db.data.frame(”houses")
houses_linregr <-
          madlib.lm(price ~ tax
                     + bath
                     + size
                    , data=d)
```

SQL Code

```
SELECT madlib.linregr_train( 'houses’,
                       'houses_linregr’,
                                'price’,
            'ARRAY[1, tax, bath, size]’);
```

PivotalR Workflow



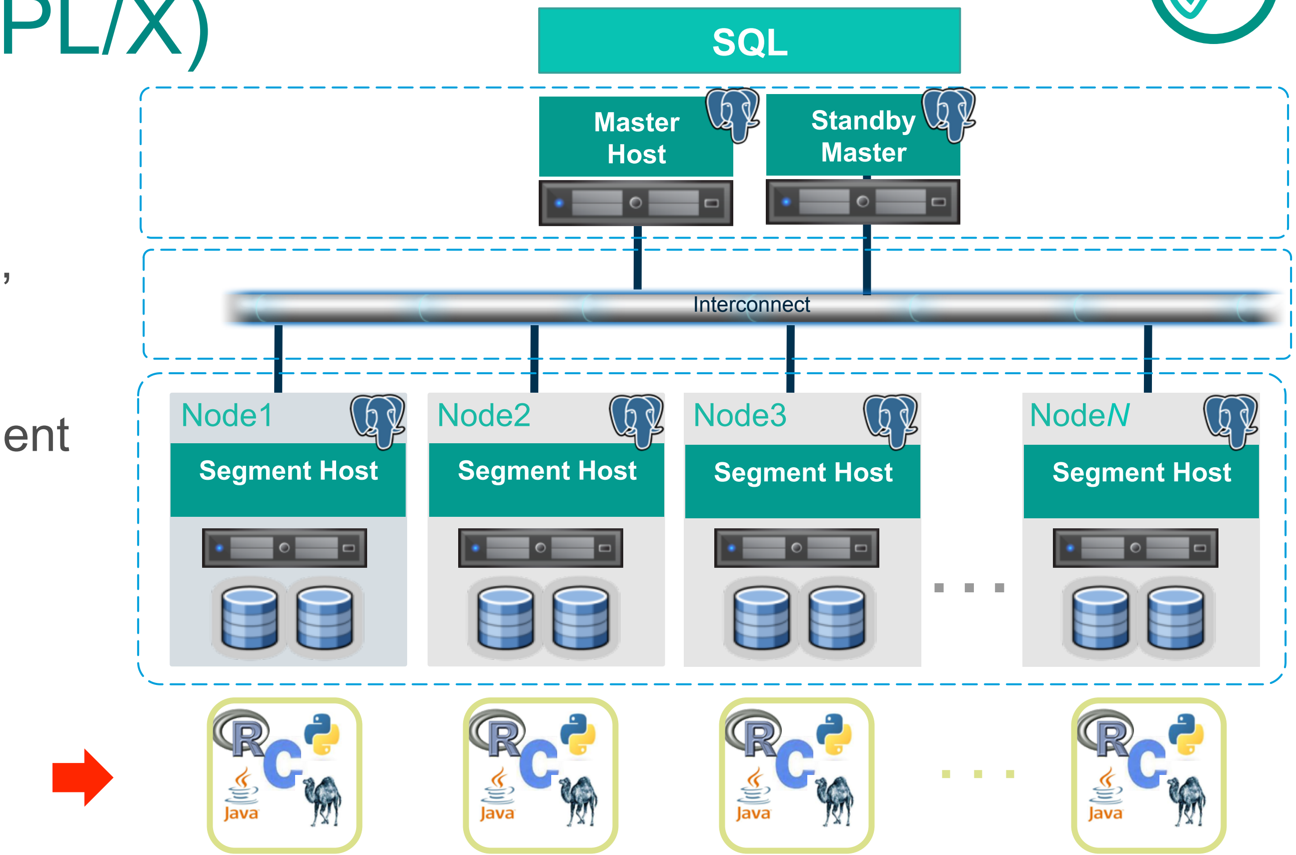
No data here

Data Never Leaves DB

Data lives here

Procedural Language Extensions (PL/X)

- Write functions in Python, R, C, Java, pgsql, Perl
- Run on each segment in data parallel manner



Data Science Bundle

100+ Libraries in Python & R

gensim

NumPy

MCMCpack

TensorFlow

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

scikit learn

pyLDAvis

LIFELINES

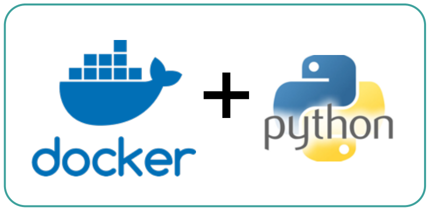
spaCy

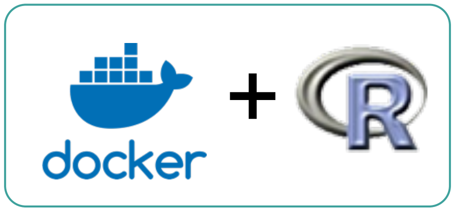
SM

XGBoost

BeautifulSoup

PL/Container



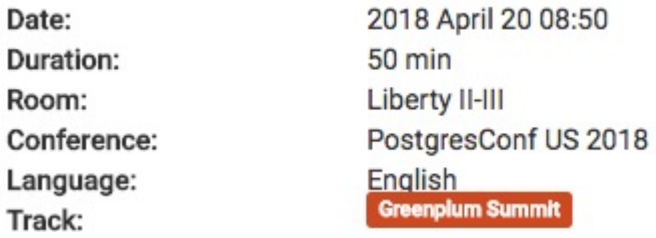


- Execute functions in isolated secure containers
- Deploy code and functions as non super-user

"Customize and Secure the Runtime and Dependencies of Your Procedural Languages Using PL/Container"

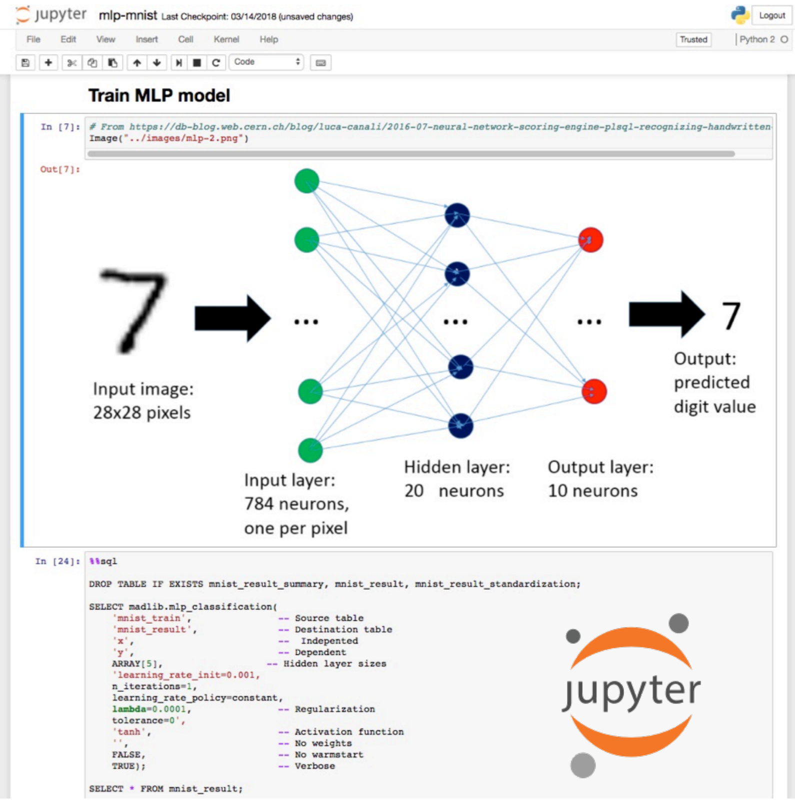
Hubert Zhang

Jack Wu



Date:	2018 April 20 08:50
Duration:	50 min
Room:	Liberty II-III
Conference:	PostgresConf US 2018
Language:	English
Track:	Greenplum Summit

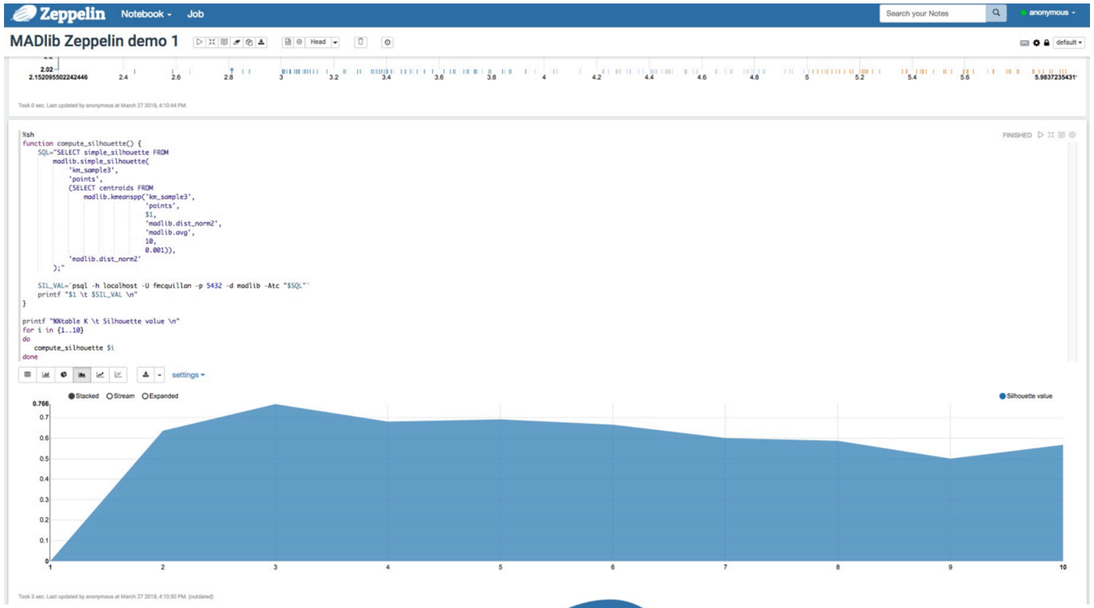
Data Science Notebooks



Train MLP model

```
In [7]: # From https://db-blog.web.cern.ch/blog/luca-canali/2016-07-neural-network-scoring-engine-plsql-recognizing-handwritten
        Image("../images/mlp-2.png")
Out[7]:
```

```
In [24]: %%sql

         DROP TABLE IF EXISTS mnist_result_summary, mnist_result, mnist_result_standardization;

         SELECT madlib.mlp_classification(
             'mnist_train',                  -- Source table
             'mnist_result',                 -- Destination table
             'x',                            --  Indepented
             'y',                            -- Dependent
             ARRAY[5],                     -- Hidden layer sizes
             'learning_rate_init=0.001,
             n_iterations=1,
             learning_rate_policy=constant,
             lambda=0.0001,                  -- Regularization
             tolerance=0',
             'tanh',                         -- Activation function
             '',                             -- No weights
             FALSE,                          -- No warmstart
             TRUE);                          -- Verbose

         SELECT * FROM mnist_result;
```



MADlib Zeppelin demo 1

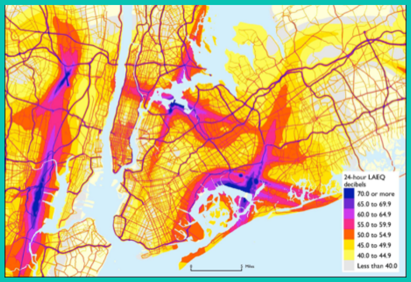
Took 0 sec. Last updated by anonymous at March 27 2018, 4:10:44 PM.

```
%sh
function compute_silhouette() {
    SQL="SELECT simple_silhouette FROM
        madlib.simple_silhouette(
            'km_sample3',
            'points',
            (SELECT centroids FROM
                madlib.kmeanspp('km_sample3',
                                'points',
                                $1,
                                'madlib.dist_norm2',
                                'madlib.avg',
                                10,
                                0.001)),
            'madlib.dist_norm2'
        );"

    SIL_VAL=`psql -h localhost -U fmcquillan -p 5432 -d madlib -Atc "$SQL"`
    printf "$1 \t $SIL_VAL \n"
}

printf "%%table K \t Silhouette value \n"
for i in {1..10}
do
  compute_silhouette $i
done
```

Took 3 sec. Last updated by anonymous at March 27 2018, 4:10:50 PM. (outdated)

Apache Zeppelin

5. Geospatial

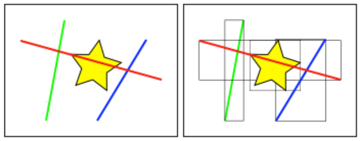


Geospatial Analytics with PostGIS

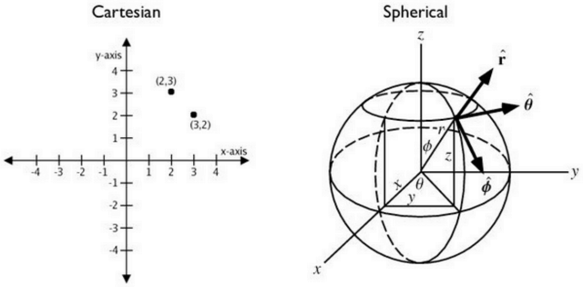
Geospatial Objects for PostgreSQL

- PostGIS is a spatial database extension which allows for analysis and processing of GIS objects

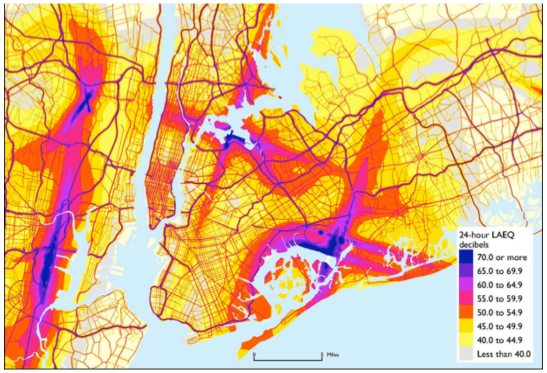
Spatial Indexes & Bounding Boxes



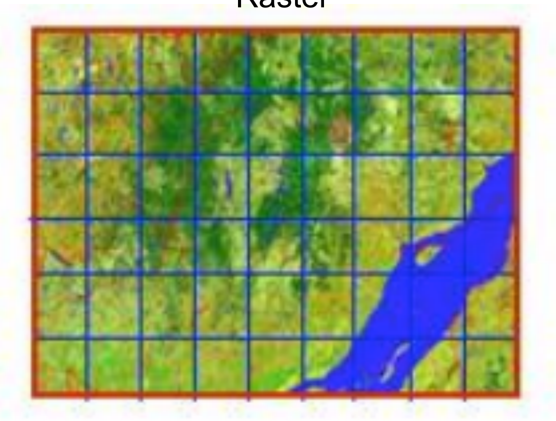
Round earth calculations



Vector



Raster

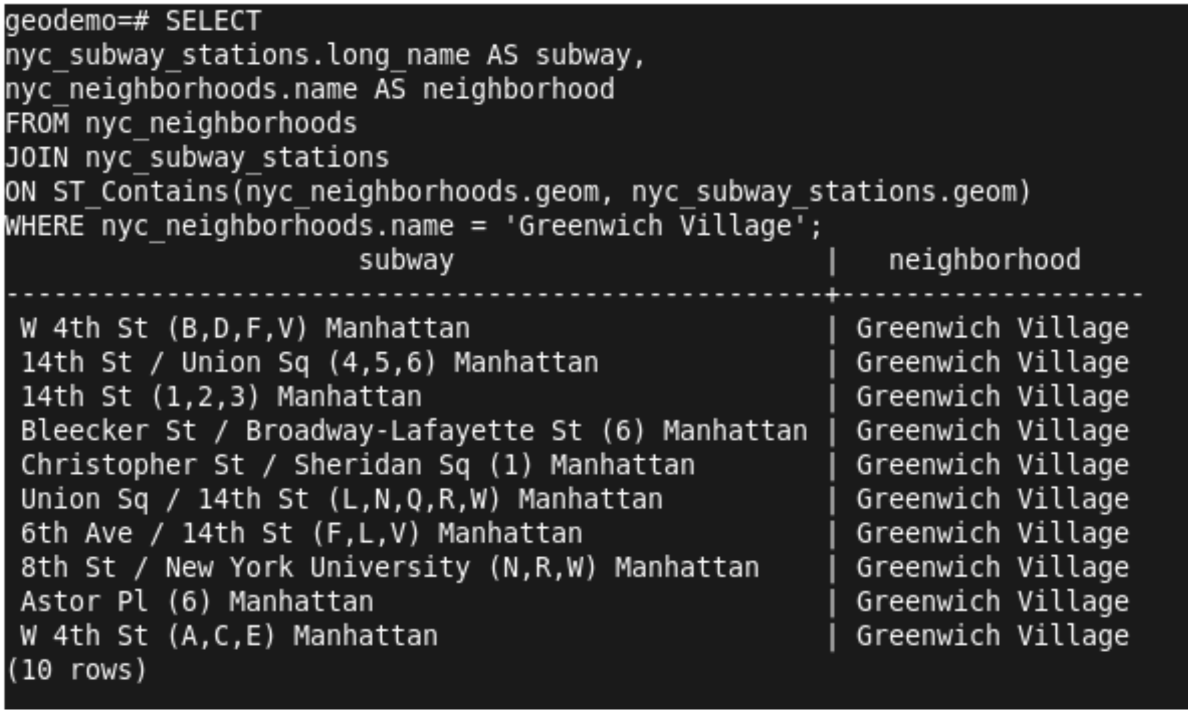


Spatial Relationships & Joins

- Relationships
  - ST_Equals
  - ST_Intersects
  - ST_Crosses
  - ST_Disjoint
  - ST_Overlaps
  - ST_Touches
  - ST_Within
  - ST_Contains

Spatial joins use spatial relationships as the join key

Example:
Subway stations: POINT
Neighborhoods: MULTIPOLYGON

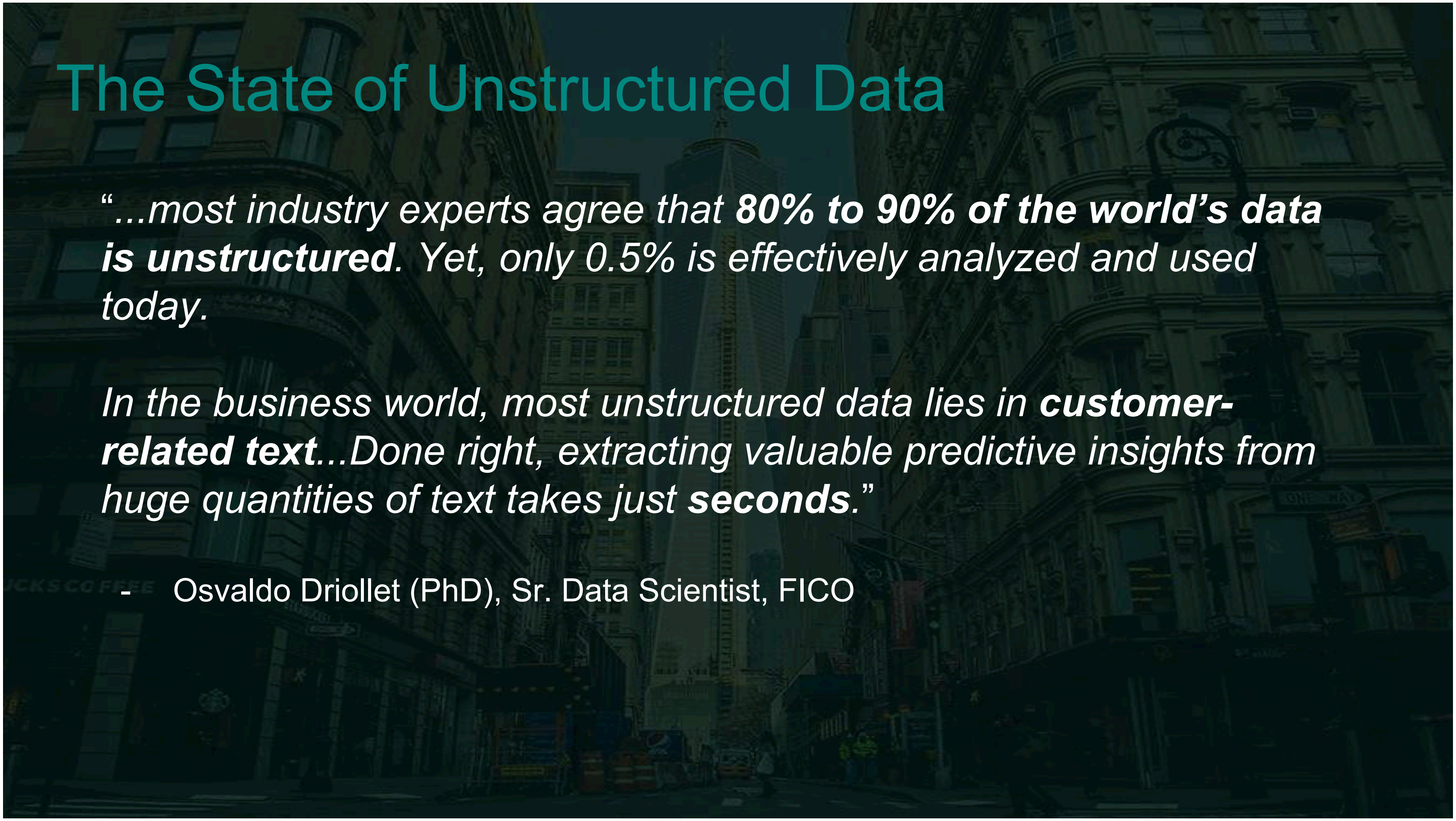


```
geodemo=# SELECT
nyc_subway_stations.long_name AS subway,
nyc_neighborhoods.name AS neighborhood
FROM nyc_neighborhoods
JOIN nyc_subway_stations
ON ST_Contains(nyc_neighborhoods.geom, nyc_subway_stations.geom)
WHERE nyc_neighborhoods.name = 'Greenwich Village';
                      subway                      |   neighborhood
--------------------------------------------------+-------------------
 W 4th St (B,D,F,V) Manhattan                     | Greenwich Village
 14th St / Union Sq (4,5,6) Manhattan             | Greenwich Village
 14th St (1,2,3) Manhattan                        | Greenwich Village
 Bleecker St / Broadway-Lafayette St (6) Manhattan | Greenwich Village
 Christopher St / Sheridan Sq (1) Manhattan       | Greenwich Village
 Union Sq / 14th St (L,N,Q,R,W) Manhattan         | Greenwich Village
 6th Ave / 14th St (F,L,V) Manhattan              | Greenwich Village
 8th St / New York University (N,R,W) Manhattan   | Greenwich Village
 Astor Pl (6) Manhattan                           | Greenwich Village
 W 4th St (A,C,E) Manhattan                       | Greenwich Village
(10 rows)
```

From Introduction to PostGIS, http://workshops.boundlessgeo.com/postgis-intro/

6. Text

INTRODUCTION
QUALITATIVE
SOFTWARE
BROWSE
ANALYSIS
WELCOME
TOOL
CONCORDANCE
TEXT
DATA
ANALYZER
FREE
CHECKOUT
CONTENT
WRITING
ACADEMIC
CENTER
ENCYCLOPEDIA
BLOG
OPINION
RESEARCH
PORTAL

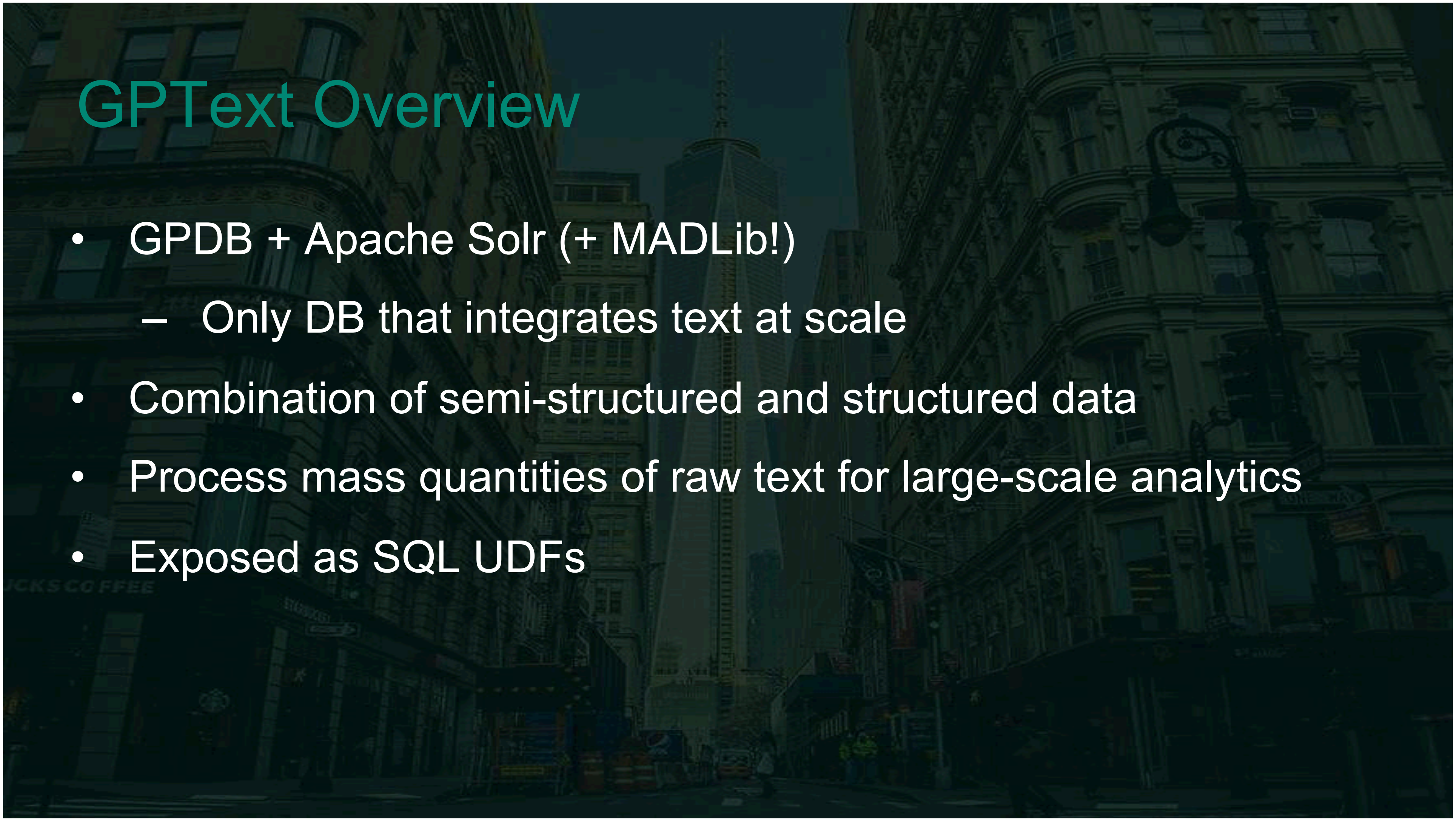


The State of Unstructured Data

*“...most industry experts agree that **80% to 90% of the world’s data is unstructured**. Yet, only 0.5% is effectively analyzed and used today.*

*In the business world, most unstructured data lies in **customer-related text**...Done right, extracting valuable predictive insights from huge quantities of text takes just **seconds**.”*

- Osvaldo Driollet (PhD), Sr. Data Scientist, FICO

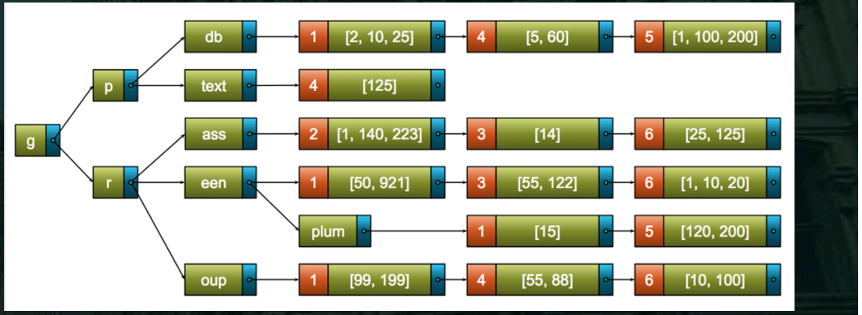


GPText Overview

- GPDB + Apache Solr (+ MADLib!)
  - Only DB that integrates text at scale
- Combination of semi-structured and structured data
- Process mass quantities of raw text for large-scale analytics
- Exposed as SQL UDFs

GPText Index

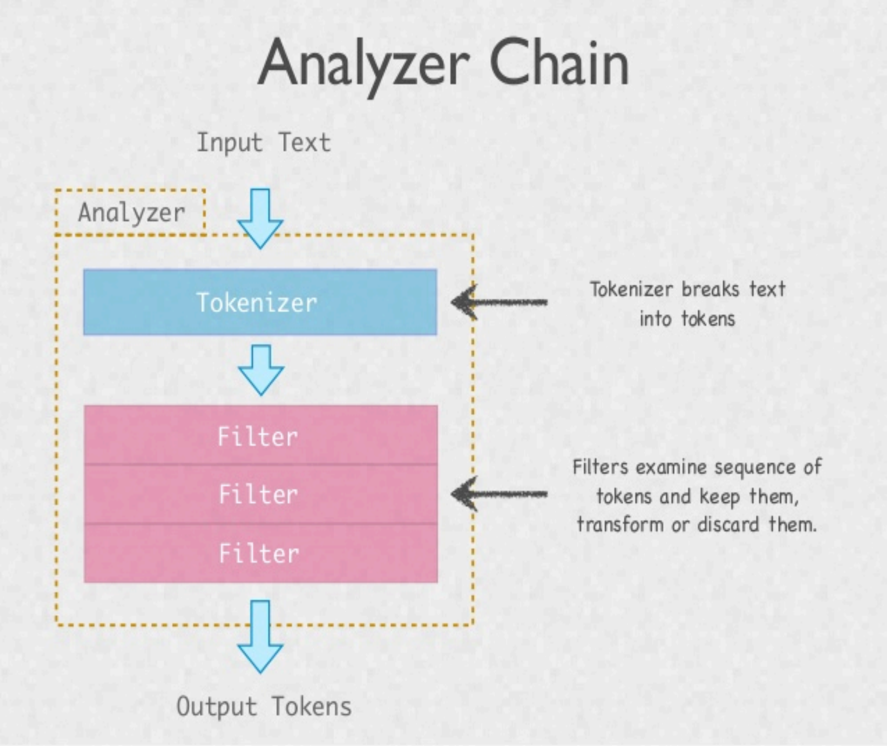
- Efficient Storage
  - Word, Position, Synonyms, Stem, Relevancy, Emoticons
- Fast Search
  - Indexed, not Scanning
- Relevant Results



$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \times \mathrm{idf}(t,D)$$

Analyzer Chains

- Document formats are NOT standard
  - International, Social Media, Logs, etc.
- Parse and Extract without losing meaning!

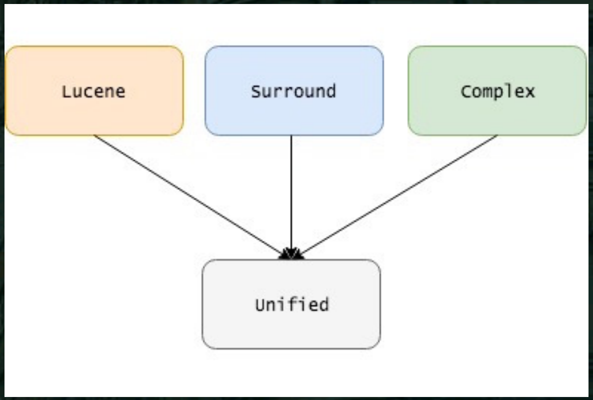


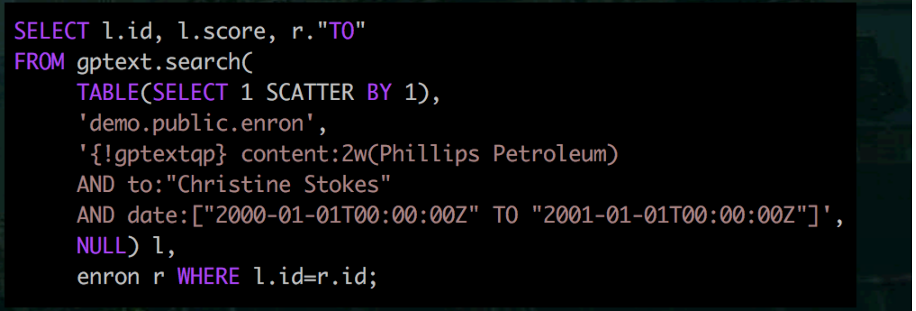
Tokenizer breaks text into tokens

Filters examine sequence of tokens and keep them, transform or discard them.

Unified Query Parser

- Designed to support multi-faceted queries
  - Boolean
  - Proximity
  - Wildcard
- No need to write multiple individual queries with joins

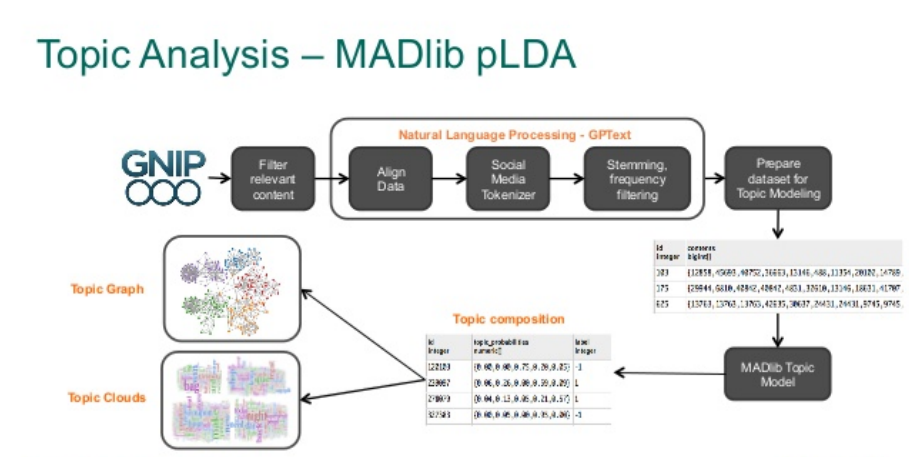




```
SELECT l.id, l.score, r."TO"
FROM gptext.search(
     TABLE(SELECT 1 SCATTER BY 1),
     'demo.public.enron',
     '{!gptextqp} content:2w(Phillips Petroleum)
     AND to:"Christine Stokes"
     AND date:["2000-01-01T00:00:00Z" TO "2001-01-01T00:00:00Z"]',
     NULL) l,
     enron r WHERE l.id=r.id;
```

GPText + MADLib

- Integrated with MADLib
  - Topic Modeling
  - Clustering
  - Sentiment Analysis
  - Sequence Pattern Mining



Topic Analysis – MADlib pLDA

id integer	contents bigint[]
383	{12858,45089,48752,36663,13146,488,11354,20182,14789,
375	{29944,6810,42842,48842,4831,32610,13146,18631,41787,
625	{13763,13763,13763,42635,30637,24431,24431,9745,9745,

id integer	topic_probabilities numeric[]	label integer
122183	{0.02,0.00,0.75,0.20,0.25}	-1
230067	{0.06,0.26,0.00,0.39,0.29}	1
270079	{0.04,0.13,0.05,0.21,0.57}	1
327583	{0.02,0.05,0.06,0.35,0.20}	-1

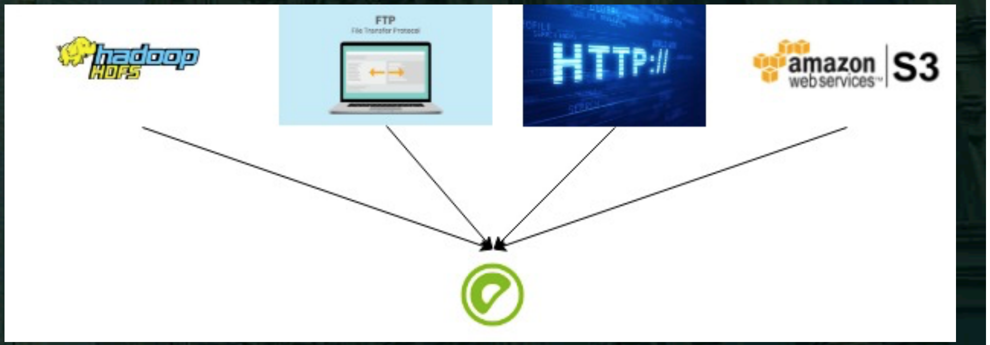
External Indexing

- Ability to connect to external data sources
  - Currently: HTTP, HDFS
  - Planned: FTP, S3

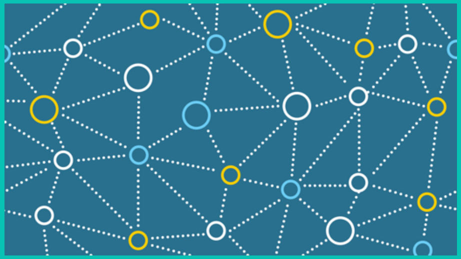
- Index and Store raw files (PDF, Word, Mail, etc.)

- Access and search your data, no matter where, no matter what.

Apache™ Tika



7. Connectivity

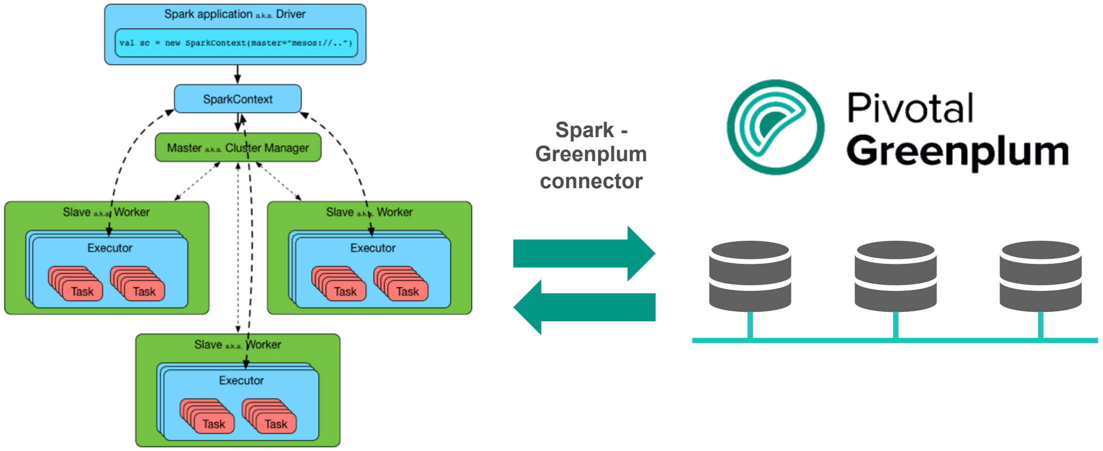


Pivotal

Greenplum - Spark Connector

APACHE Spark™

In-memory processing

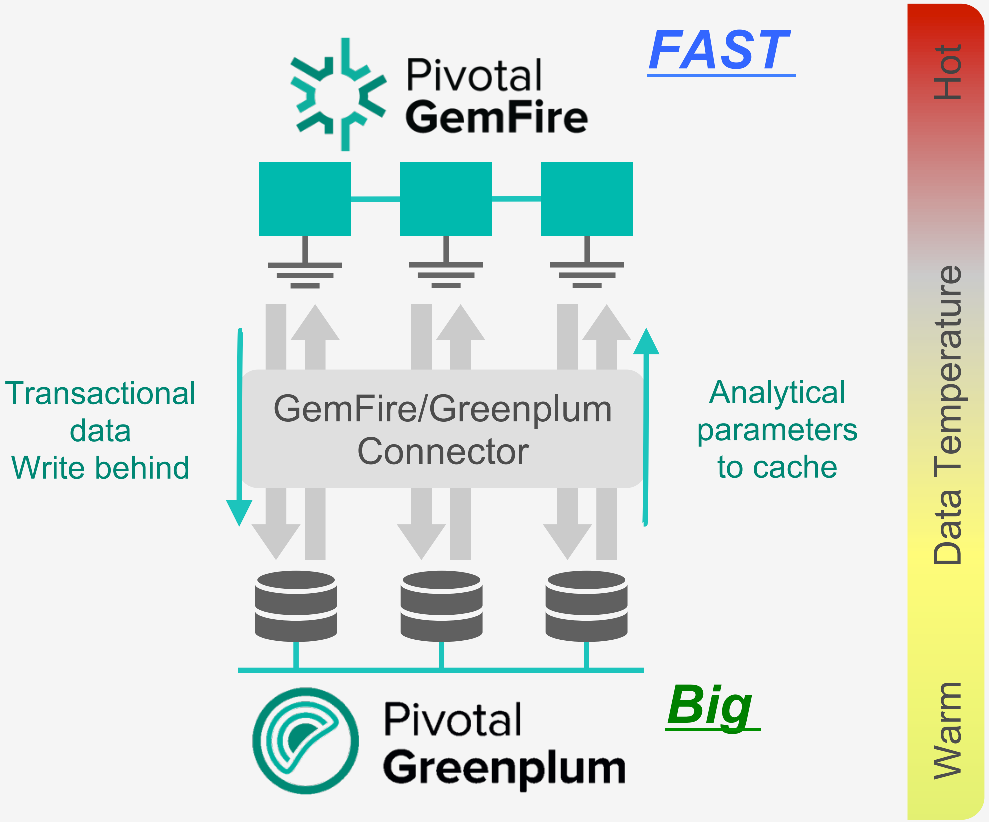


- Provide Data Access to Greenplum Data
- Leverage SPARK Skill Set of Data Scientists
- Use off-cluster resources to do computations
- Push result sets back into Greenplum for storage

Greenplum - Gemfire Connector

Seamlessly share data between GemFire and Greenplum

Bi-Directional
Direct Connection
--
GemFire and Greenplum Segment Servers


Pivotal GemFire
FAST
Transactional data Write behind
GemFire/Greenplum Connector
Analytical parameters to cache
Big
Pivotal Greenplum
Hot
Data Temperature
Warm

8. Example Use Cases

Pivotal

NICT

BIG DATA ANALYTICS LABORATORY

Event Data Warehouse (EvDW) System Architecture

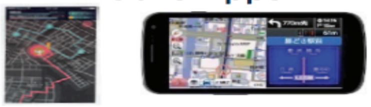
Applications

GIS tools

QGIS

Navigation

Mobile Apps



Machine Learning/Data Processing Infrastructure

PostGIS MADlib

Pivotal Greenplum

- Risk map creation
- Route analysis
- Event Prediction
- Event Data Archives
- Event Association analysis
- Spatial/ temporal analysis

Sources

Raster data

- *Weather radar*
- *Air pollution*
- *Floating population*

Vector data

- *Road network*
- *Traffic data*
- *Atmospheric observation*
- *Mobile/automotive sensing*

Social Network Service data

- *Geo-tagged Twitter*

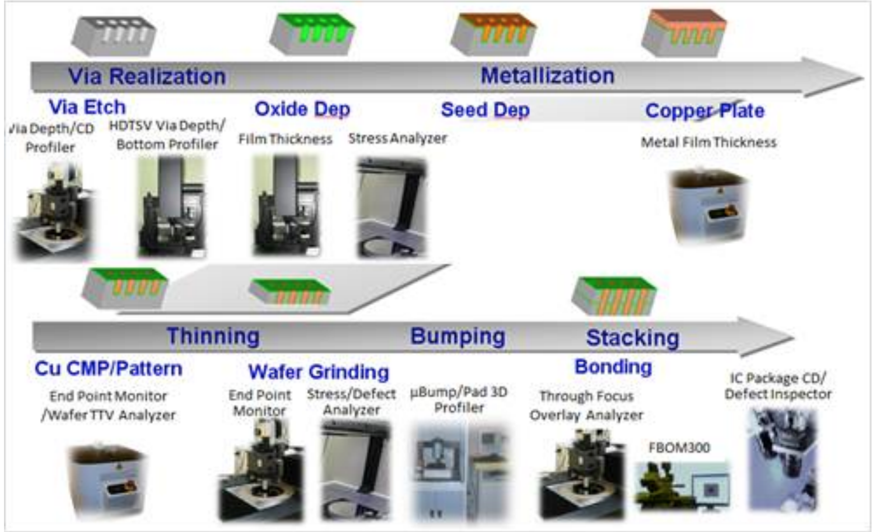
Other

- *Origin-destination (people) flows*
- *Epidemiological surveillance data*

The QGIS logo is a registered trademark of the QGIS project (qgis.org).

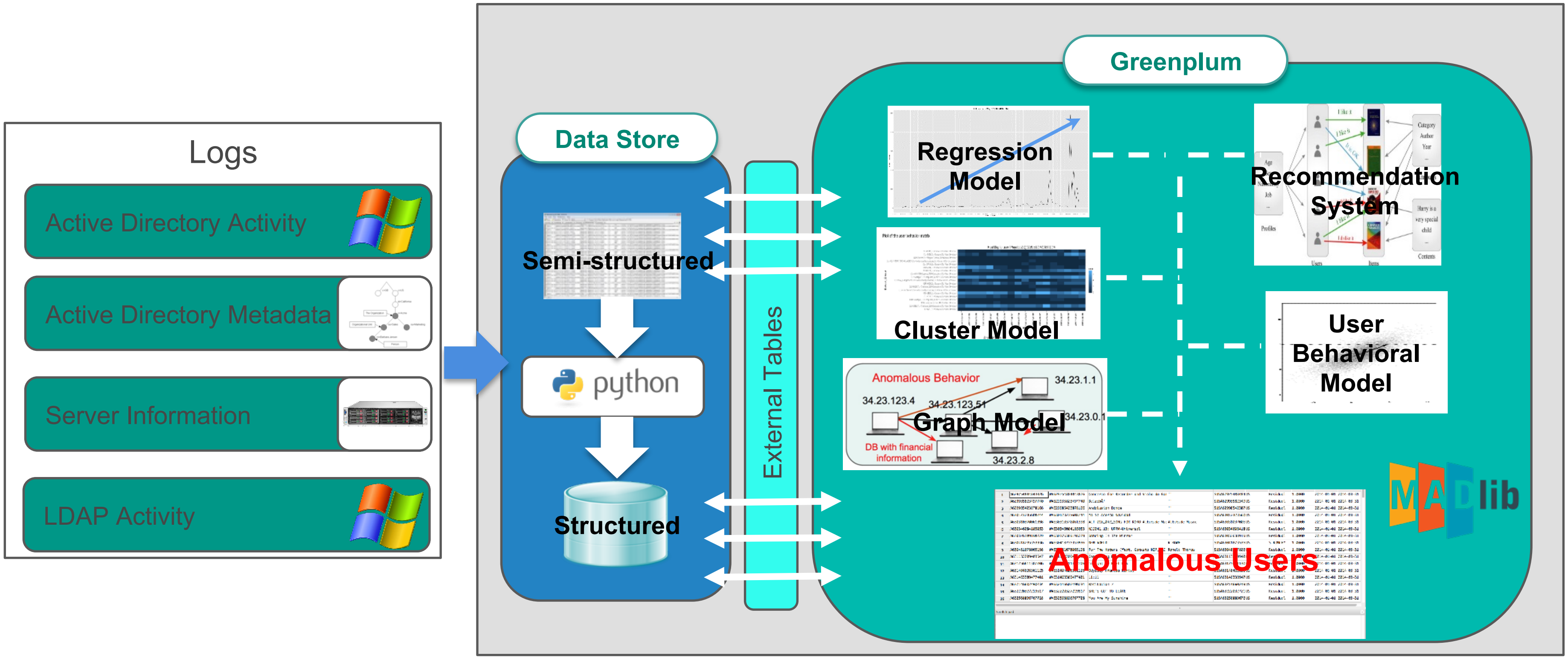
Operations - Parts Monitoring

- Monitoring 100s of different models + parts
- Structured Data + Operator Notes
- Minimized Recall Risk and Improved Reliability



PostGIS MADlib Solr

Cyber Security - Lateral Movement Detection



Logs

Active Directory Activity

Active Directory Metadata

Server Information

LDAP Activity

Data Store

Semi-structured

python

Structured

External Tables

Greenplum

Regression Model

Cluster Model

Graph Model

Recommendation System

User Behavioral Model

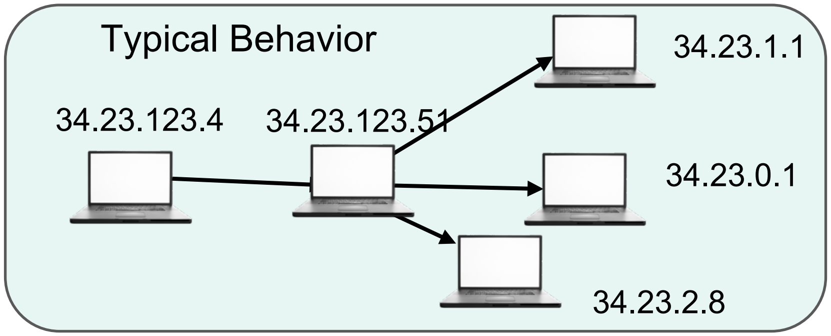
Anomalous Users

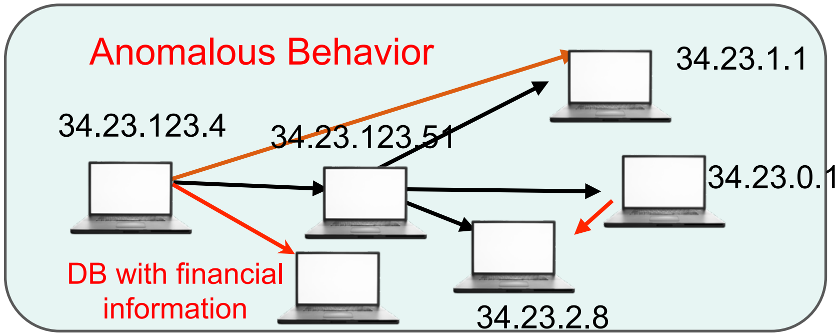
MADlib

https://content.pivotal.io/blog/better-threat-detection-and-response-with-analytics-for-lateral-movement

Cyber Security (continued)

- Using historical window events data to build graphs of typical user behavior*
- Is this behavior typical?
- Graph models are sensitive to direction, order, and frequency.





*Reference: Alexander D. Kenta, Lorie M. Liebrock, Joshua C. Neila. *Authentication graphs: Analyzing user behavior within an enterprise network.*

9. Looking Ahead

Pivotal

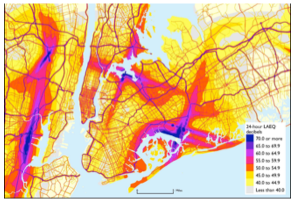
Greenplum Integrated Analytics

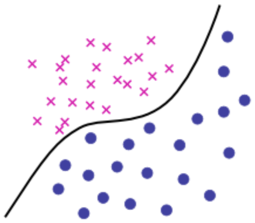
Data Transformation

Text

Traditional BI

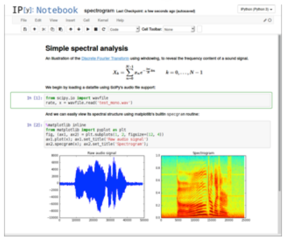
Geospatial





Machine Learning

Data Science Productivity Tools



Simple spectral analysis

An illustration of the Discrete Fourier Transform using windowing, to reveal the frequency content of a sound signal.

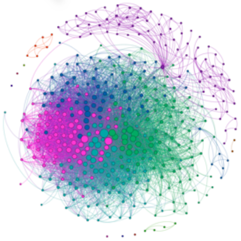
$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

We begin by loading a datafile using SciPy's audio file support:

```
In [1]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin specgram routine:

```
In [2]: %matplotlib inline
        from matplotlib import pyplot as plt
        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```



Graph

Thank you!

Pivotal