**DSE 203 9.30.2017**

In database analytics -> don’t do in memory ie. in Python do within the server

* Do the feature extraction within the database (without performing expensive data transfer)

Tuple (mathematical definition) -> cross product of a set of attributes

NULL -> did not observe therefore did not recognize it vs. I looked for an observable but did not find a value there

Union (removes duplicates) vs. union all (doesn’t remove duplicates)

Group by (ie. clustering) -> Identification of grouping variable, create a list of grouped variables, may or may not aggregate (SQL it is required but modern is not necessarily required), it doesn’t necessarily imply aggregation

Projection -> column subset not row subset

Generalized projection -> not only will pick the column but will do something with it

Semijoin -> send sample of table to other table and determine what matches between sample to root table and then join completely, ie. if data is stored on multiple machines this is a more efficient means of doing a join, minimizes communication/data transfer, for distributed systems, happens behind the scenes

In relational cannot do ‘What rows have Johnny in it?’. Relational databases were built from first-order databases, schema databases are not completely queryable. But Johnny can be a tag in json

Tree structure -> can specify text search all over, tree depth

Semi-structured -> can be modeling as a tree or a graph

Ie. graph van be like html where @href is used or XML w/ID & IDREF

Schemaless models ie. MongoDB document model

Schematized models ie. XML DTDs, XML schema, JSON schema, AsterixDB Model

List child (red) vs. atomic value child (blue) they are different in semi-structured data structure

In AsterixDB -> if you know the datatype you specify the atomic type of the data in the json

-create type as closed (if you know the data structure) vs. open (if anticipate that the data will have more things), open is less efficient but more flexible

-use dataverse = create database -> create dataset (table creation)

-then read in the data by specifying path, type-name, format (ie. adm = stylized Json)

-AQL (ie. XQuery languages) & SQL++ created for querying AsterixDB, cannot express queries w/unbounded paths (ie. need to know to some degree the number of levels to the tree)

SQL++

Ie. SELECT VALUE user FROM GleambookUsers user; -> will get the subtree of root node

Built in functions ie. get\_date\_from\_datetime

-type mismatch: fxn field-access-by-name expects its 1st input parameter to be type object, but the actual input type is an array (need to know the datatype of the data which will be extracted)

-> use UNNEST (flatten the structure allows for the ability to access the data)

-equi-join (both matching columns to be shown in product), natural join (ie. equi-join but 1 column will be repressed in product), nested queries (theta join = not by equality by another condition ie. spatial)

-SOME/EVERY is a Boolean to add to ‘WHERE’ ie. for every that comes back needs (vs. in SQL would need to do negation, etc.)

-UNNEST has SQL’s inner join semantics – that is, if a message has no referred topics, no tuple corresponding to that message will be emitted in the result

-can group w/o aggregation

-want to create a new json object massaged in the way already wanted

Posting lists -> indexes of documents -> can do document intersecting

Solr is a typical engine used for text search; search engine database

-declare relational fields, want to handle term upset (must calculate the positions), ie. find nearest 2 words from 1 word

-there is a whole entire filter class

-returns rank order of documents

-has its own formula in creating a document store

-interacting with a GUI 🡪 then creates a rest URL for calling the data back; as if making a REST call

-fq (field query) -> gives a set of conditions ie. search within results; can be a drill down into even further

-facet -> produces a histogram; can be used to understand dominant terms ie. What are the important conversations going on?

-> facet.query ie. fq it is a search within the results; to do a conditional histogram

-q -> allows for function specification

-tvrf -> term vector, wait tf from the system ie. mul(tf(Text, ‘trump’), idf(Text, ‘trump’))

-internal scoring can be designed in several ways

Use each of the tools to their strength

Dependency graph -> how words POS are related to one another, used for dependency analysis

-information extraction ie. as well as vs. geo wells

-would get json returned w/word, lemma, pos, speaker, etc. & dependencies

Ie. Stanford CoreNLP can be used for POS tagging

Analysis plan -> database operations -> exploratory analysis

-get a random sample ie. SQL: ORDER BY random()

Ie.

SELECT \* from OrderTable

WHERE random() <= 0.01

ORDER BY random()

LIMIT 1000;

->this guarantees randomness, this is a lot faster b/c doesn’t go through all of the data, might not always have 1000 records back but them can relax the requirements in the ‘Where’ clause

-understand how to do data exploration using PostGres

Table sampling -> simpler to use db page vs. db sample, randomize db page/index on page would be a lot faster; this is possible in PostGres; database supported sampling