**DSE 203 10.13.2017**

Consider the data integration architecture -> design axes (one vs. multiple data sources – global schema vs point-to-point – materialized vs. virtual)

Materialized vs. Virtual

Prof wants to try a virtual data integration

* Heterogeneous data sources and need to query them effectively

Centralized vs. Distributed

Ie. medical documents ie. HL7 need to translate data accordingly to become compatible with the protocol

* Data exchange -> suppliers data is different from the acquirer, need to have a middleman to transform the data accordingly

Need to provide source description (ie. schema, access methods, auxiliary info.)

* Relation description
* Queries as Datalog, simple enough and easy to be manipulated
* Constraints in a separate block can be specified
* Notice query/data access restrictions ie. cannot access books under $15 on Amazon although the data is there

Schema mapping (take different columns in combination to create a value) vs. matching (need to put a value in there what is the value)

Data aggregation = vertical integration

Data fusion = heterogeneous data fragments from ie. internet, hard to assimilate into integration system; data is incomplete, inaccurate, and dirty

**Data Matching**

Can do this in Solr

A match is a correspondence/association btw. Individual structures in diff. data structures (value, tuple, and schema matching), goal is to find correspondences, not to ‘clean’/rectify

Band join -> equivalence range between value and associated values ie. within a delta of n do a band join

Fuzzy or similarity join ie. Dave Smith join with David D. Smith

Use:

* Similarity (the higher s(x,y) the more likely that x & y match), but not scalable -> sol’n = apply s(x,y) to only most promising pairs, using FindCands
* Edit distance (Levenshtein distance) -> d(x,y) computes min. cost of transforming x into y, using a sequence of operators (ie. delete/insert/change character), each w/cost 1; normalize using length (of words) normalization
  + Use dynamic programming via matrix logic to iterate and calculate the cost of the transformation
* Needleman-Wunsch Measure -> generalized Levenshtein edit distance (computes similarity scores vs. distances values, edit costs in score matrix are generalized, insertions/deletions are generalized into gaps), defines notion of alignment btw. x,y , allows for gaps, use a score matrix & a gap penalty
  + Alignment score = score of scores of all correspondences – sum of penalties of all mismatches & gaps
* Affine Gap Measure -> extension of Needleman-Wunch handles longer gap more gracefully (would need to subtract the gaps out more), define cost of opening a gap vs. cost of continuing the gap
* Smith-Waterman Measure -> find 2 substrings of x & y that are most similar, use local alignment; find the best local alignment btw. x & y, & return its score as the score btw. x & y, diagonal is always the preference
  + Ie. looking at genetic sequences
* Jaro Measure -> little spelling errors, compares strings does not consider gaps
* Jaro-Winkler Measure -> captures cases where x & y have a low Jaro score, but share a prefix, still likely to match
* Jaccard Measure
* TF/IDF Measure

Cannot check all string combinations, because at some point it is not scalable, how does one determine which strings to measure -> FindCands

* Create indexing for words
* Size filtering -> if I have a short string I shouldn’t match it with a long string
* Prefix filtering -> if 2 sets share many terms -> large subsets of them also share items