Knit Data Together: Cleansing & Matching Techniques

Matthew Williams
LexisNexis

NYOUG – December 14, 2006

Cleansing & Matching Techniques

1. Data cleansing & matching techniques to help match up unrelated streams of data. Stacking equivalent dimensions and bundling matching dimensions, column-level thesauruses & data standardization, Match 'Code' phonetic + corporate hierachy confidence values.

2. Learning Systems-type techniques to improve Bad Data Tolerance over time. Iterations, Feedback of good matches and the Importance of Eyeballs when building and growing a thesaurus.

3. Physical & Application Design Choices to make best use of physical resources in a Decision Support-type matching program tasked to compare millions x million of unparsed strings for high-confidence match equivalence.

Who I Am

- Oracle DBA for 10 years – started on Oracle 6 on Macintosh ! Learned Oracle internals and the importance of DB tuning to make best use of physical resources. Learned danger of Cartesian Products !
- College Degree in Cognitive Science, comparing associative learning networks with Human Learning from Psychology and Linguistics.
- Interested in applying 'weak' AI techniques to augment learning and value of enterprise systems for large data stores.
- Oracle DW development work with specialty in scoring & weighting algorithms.
- Pleased to share my knowledge and experience with larger Oracle Community. Interesting current project touches on each part of my background...

Where I work

- Information Resource for Legal Industry
- Customer Feedback launched the project
- Not simply a Legal Directory anymore, but a Client Development Partner providing real-time Corporate and Legal tracking data- Showing which firms handle which Corporate Customer's business in which Practice Areas.
- Show Corporate Practice Trends over time.

What We Built

- Integration challenge- Non-key based 'confidence' matching needed between wildly different quality Data Inputs.
- Data Source 1 – MH data is backed by a team of editors, verified, updated daily, and kept clear of duplicates and outdated information. 2 million Law firms, ~1 million Attorneys, historical data dating back decades
- Data Source 2 – LexisNexis Courtlink data comes direct from the Federal Courts, a amalgamation of hundreds of separate typists making conflicting choices on data inputs... Names, Vanity Street addresses, out-dated info, etc. ~40 million Case filings added each year.
- Data Source 3 - LexisNexis Company Dossier – a conslidated Business profile for over 43 million worldwide companies, high quality non-duplicated data. Editorial Team continuously reviews/updates business data for accuracy.

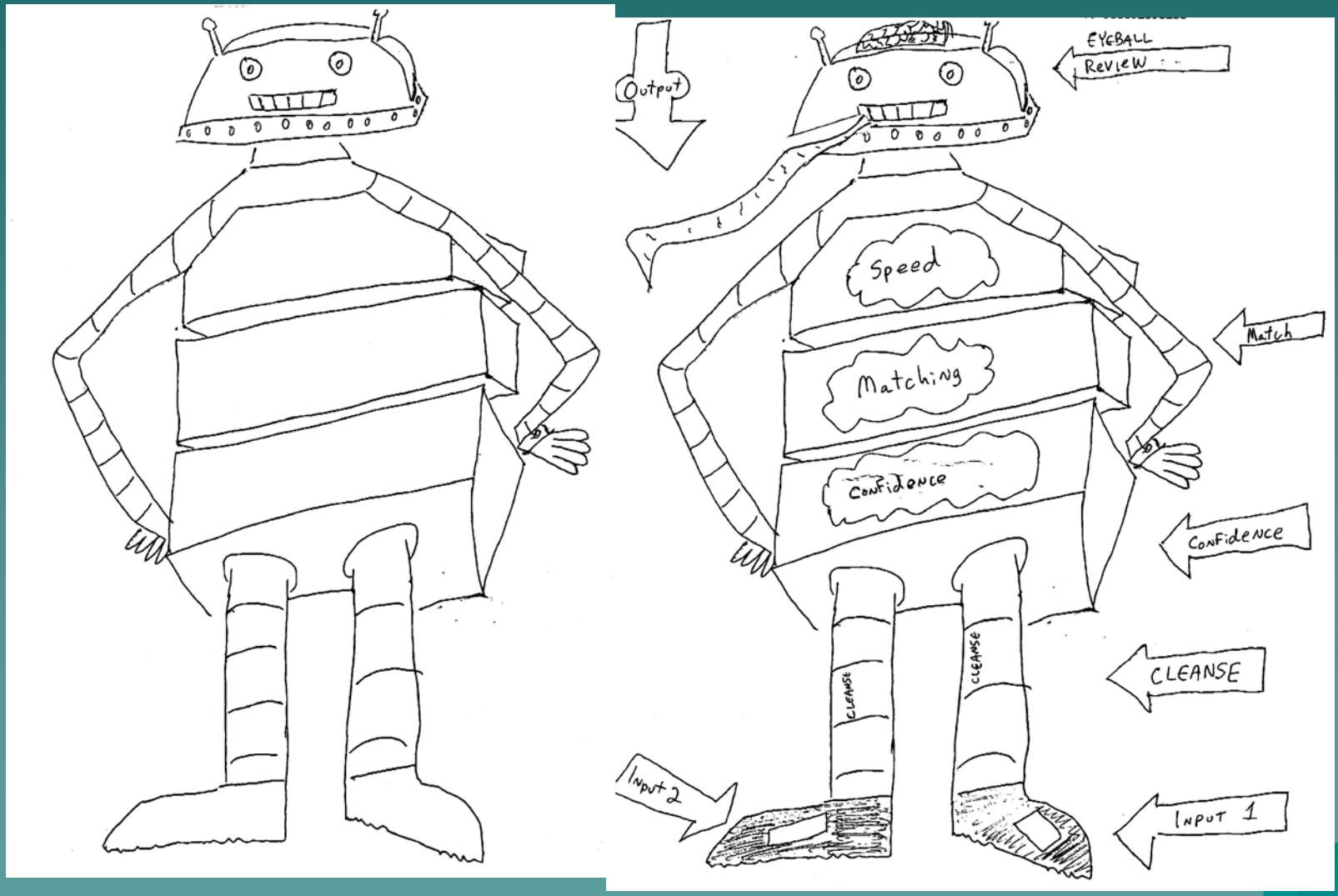
Knit Them Together

- Build a High-confidence Matching system that 'Tolerates' invalid and missing data in each possible Name/Address field.
- Missing Street, Incorrect Phone Number, Misspelled Attorney name.
- Each Match made to 'Bad' Data is fed back into the System for Perfect, Validated match next time around.
- System 'learns' the spelling/data entry habits of customers with each validated match

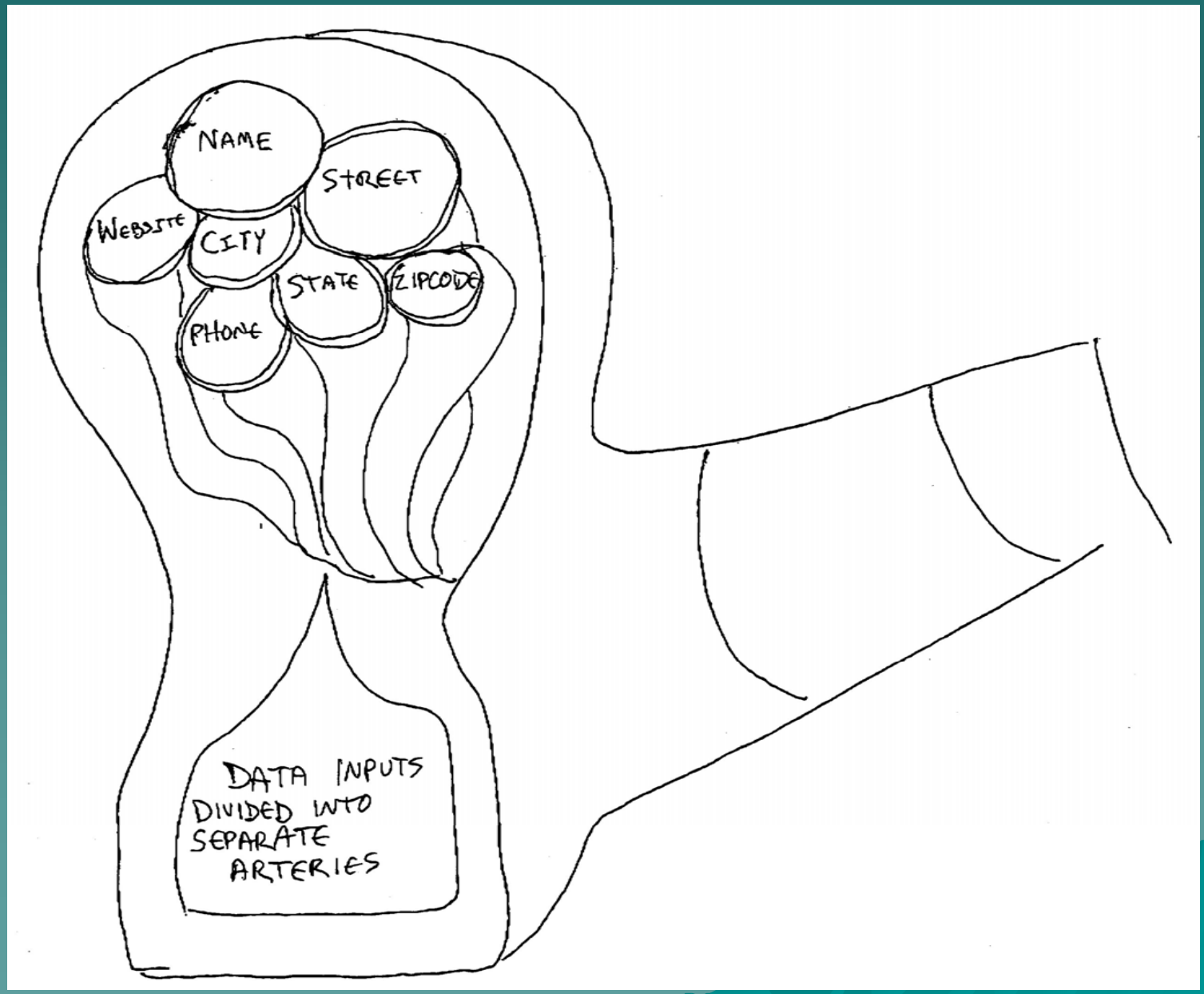
Matching Robot

- Build Matching Robot in discrete pieces
- Data Analysis (FEET),
- Separate Pipes for Data Cleansing & Translation Name/Address parts in (ARTERIES)
- Phonetic Re-spelling & Match Code application in multiple Confidence lengths (LEGS)
- Cross data streams in a cascading series of bundled Matching Dimensions (HEART).
- Process millions x millions of string comparisons in varying sets of matching dimension values. (SPEED)
- Review Output matches to verify accuracy and soundness of choices (EYEBALLS)
- Feedback Validated Matches to teach system known data entry tendencies of data inputs (BRAIN)

The Matching Robot



Step 1 – Data Analysis (Feet & Arteries)



Expert System = Dataflux/SAS (Legs)

- In the business of IT, the main question was always whether to "build, buy or rent.“ Cost, maintenance, and time-to-market are the key driving factors to decide.
- In this case, we chose to 'buy' a pre-loaded dimensions-level Thesaurus product from SAS called Dataflux.
- Built using data from a wide variety of sources, (Clients include Retail, Pharma/Healthcare, Data Services, Financial Industry)
  - Wide variety of industries/ data inputs helped them develop a pre-loaded ‘Expert System’ Thesaurus for each Name/Address part
- Noise Word Removal & Data Standardization
- Phonetic respelling of all input values
- 12-character 'bitmask value' of Phoetic string. Breadth of value to wildcard defines the confidence level of the matching strings- Food for the PL/SQL Matching Package, broken out by ‘Match Rules’

Available Products

- **Data Profiling** - Inspect data for errors, inconsistencies, redundancies and incomplete information
- **Data Quality** - Correct, standardize and verify data
- **Data Integration** - Match, merge or link data from a variety of disparate sources
- **Data Enrichment** - Enhance data using information from internal and external data sources
- **Data Monitoring** - Check and control data integrity over time

Our Data Analysis

- Each Data Stream in each project presents unique Strengths and Weaknesses…
- Name-Only data (web crawlers) requires matching weighted keyword co-occurence, focused on removing common 'noise' words, (“The”, “And”, “Web”)
- Each Project requires agreement on Business Rules before Analyzing
- Name, Address data allows for compound confidence Scores, and was used for this project… Name, Street, City/State, Zip, Phone, Fax, Website, Ticker, each with variable Confidence Degrees bundled into Collections… called “Match Rules”.
- Apply 'Dimension Level Thesaurus' to each address part… Distinct ‘Noise’ for each type

Data Cleansing (Legs)

Data Cleansing (Legs)

- Choose Capitalization to apply
  - InitCap () UPPER()
- Remove Whitespace, punctuation, special characters like # % $
- Standardize all Address Parts
- Always Context Sensitive to Address Part
  - St. = Street BUT St. Louis <> Street Louis
  - Ste. = Suite = Suite #
  - Jr. = Junior
  - Varies by Country (Zip Codes, Phone Number Parsing, etc)

Remove ‘Noise’ by Dimension Type

- Each Country has different ‘Noise’ in their Address data… SPA in Italy, Alpha-Zipcodes
- Name = Incorporated, LLC, The, LTD, Group
  - Inherently ‘Dirty’ and impossible to perfect. Experience is crucial to building right list, and tuning as you go along
- Website = WWW, HTTP://, .
- Phone/Fax = -, x, ext.
- Address = Street, Avenue,
    - Vanity Address Parts (Element and Phrase level parsing applied

Phonetic Respelling & Match Strings

- The LexisNexis Data Warehouse Group, Inc.
- LexisNexis Data Warehouse Group
- LXSNXS DT WRHS GRP
- The following values written to columns for run-time comparison of match procedures
- Match Code String Value
  - LJKHGTKSDG = 100% Match Confidence
  - LJKHGT**** = 60 % Match Confidence
  - LJKH****** = 40 % Match Confidence

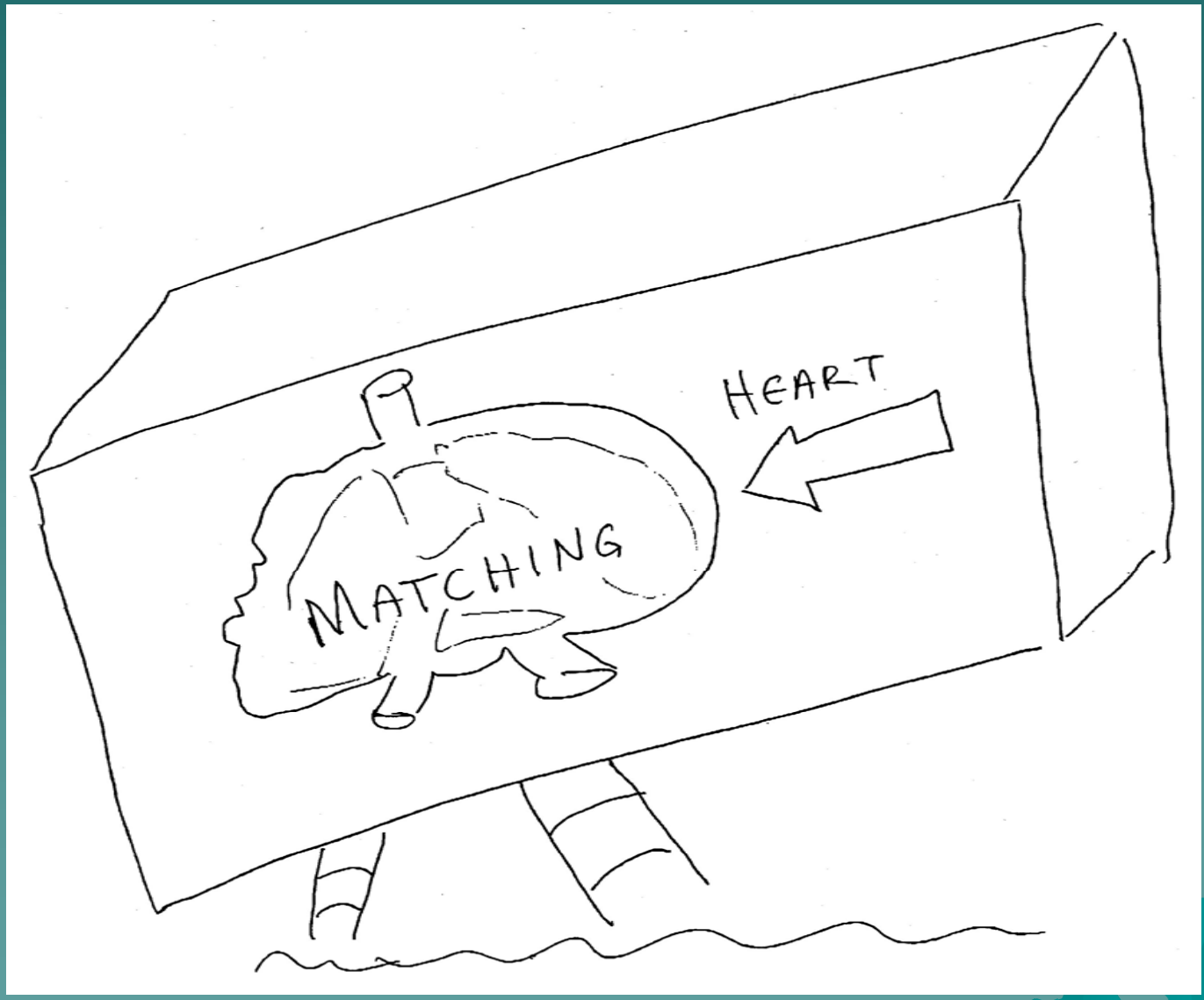
Dataflux Expert System

- Surprise ! Corporate Hierarchy Translations Also Built In...
- PEOPLESOFT CORPORATION = PPLSFT
  - DGTYUDGTYU
- ORACLE CORPORATION = ORCL
  - DGTYUDGTYU
  - “Hard-Coded Equivalence”

Problems with All Expert Systems

- Need periodic updates to stay up to date with changing language and habits... i.e. new area codes, new streets.
- 'Brittle' knowledge... does same things right, and never improves.
- Doesn't 'customize' to input for each specific project.
- Therefore, need to add a 'Learning Module' piped from a 'Feedback' Loop of validated matches to 'teach' the robot to recognize new match patterns of real-world data translations and data sources over time.
- Match feedback is a Key ROI driver to build value of product with each new matching job.
- Converts low-confidence, expensive and slow “Eyeball” matches to high-confidence, automatic “Exact matches” based on previous work.

MATCHING ROUTINES (Heart)



COMPARE DATA IN MATCHING ROUTINES (Heart)

- Create Collections of Match Dimensions at varying Confidence Levels… Called “Match Rules”
- Need to Define “Confidence Hierarchy” and Run the Matching Routine from Highest to Lowest Confidence Matching
- Requires Extensive Iterative Testing with Slow, Business Aware “Eyeballs” to Define Hierarchy
- Time-Consuming, but Critical for each Project

Stack Equivalent Dimensions for Bad Data Tolerance

Ticker		
Website	Phone Number	
DBA	Zip	Area Code + Zip
Organization	City + State	State + Zip

Robot must Tolerate Bad and Incomplete Data

- Teach the robot to tolerate bad data… missing data, misspelled data, out of date data, using bundled collections of Match Dimensions in varying confidence levels… i.e. Match Rules.
- Match Rules developed from Stack of equivalent dimensions
- Observations: Street contains lowest quality input. Names are often misleading and require use of Corporate alias and DBA values if available.
- Important to cast a Wide Net… even Low-Quality Match Rules provide good matches… with a lower ‘Kill’ rate

Eyeballs Needed : Define Confidence Levels for Match Rules

- Create a test file with 10 examples for each Match Grouping.
- Expensive and time-consuming, but must test different collections of matching dimensions at varying degrees of compound confidence to define Order of Match Rules to process. Use stacked dimensions as roadmap for Match Rules.
- Need Eyeballs & Business knowledge here to validate a 'correct' match. Some prefer Exact match, others ok with Match to a Corporate Parent. Depends on Project Needs.
- Don’t disparage Low Confidence Rules- they help cast a 'wide net' of tolerance for missing/incomplete data.

Summary

- Data is cleansed and translated within each unique name/address dimensions expert system. Noise words are removed. Match Codes are applied at multiple confidence levels for each part.
- A System of ‘Match Rule’ collections have been defined, tested, and ordered from Highest to Lowest Compound Confidence Score. IT and Business stakeholders have reviewed the system to validate Matches, and resolve ‘Possible’ Matches effectively.
- We’re ready to match- Start the Robot !!

SPEED - (Bones)

- Data Warehouse-type resource profile – String-by-string matching in flexible series of dimensions with VLDB sizes makes it impossible to do all comparisons in memory- must make create fastest possible I/O setup to keep CPUs fed with data for comparison
- Matching 43 Million rows of 12 different Address Parts (Name, DBA, Street, City, State, Zip, Phone, Fax, Ticker, Website etc).
- Matching against 3+ million LN/MH Legal Industry Data Records, (Firms, Individuals, Governments, Corporate Counsel, etc)
- Matching against 40+ million Court Filings to determine Litigants, Law Firms, and Lawyers
- 43 x 3 x 40 million in each successive pass of 'match rules' from order of precedence. At least 60 passes during each match process. Trillions of Comparisons each Run.

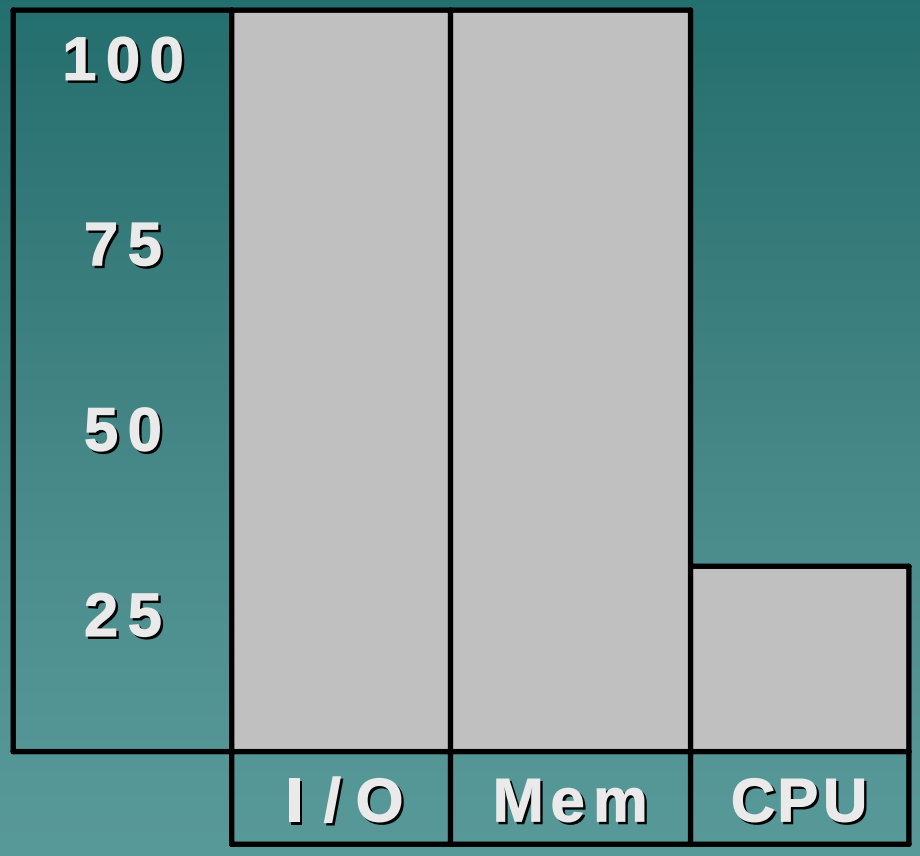
Oracle Database & Server

- Oracle 10.2.0.2.0
- Total SGA = 10 Gig of 16G available,
- 16k Block Size
- 10.2.6 on Linux 2.6.9-34.ELsmp on 4 dual-core x86_64 processors.
- Attached to a High-performance RAID-5 SAN using 2 dual-channel multi-path 1 Gbgt I/O channel cards.
- Paired 16k block size with high file-multiblock read count- using higher block size reduced performance due to narrow indexes doing Fast Full-type Scans

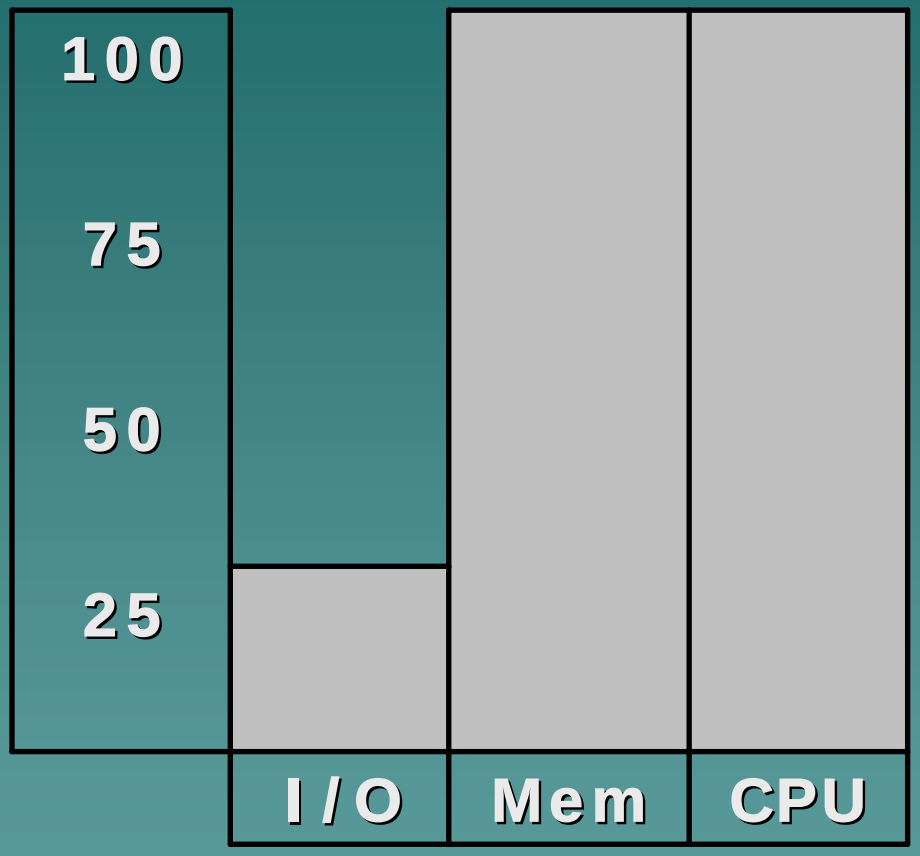
Application Choices in Oracle

- Single indexes, not composite, on each unique matching dimensions....
- Cardinality is low, but bad use for Bitmap Indexes due to high # of inputs to system each day, and need for OLTP-type quick inputs/response... found a great alternative in function-based SUBSTR() indexes on first few chars of match code strings- low cardinality means less index key space to search/ reduces IO dependence.

Performance Problems on Starting



Sad Machine

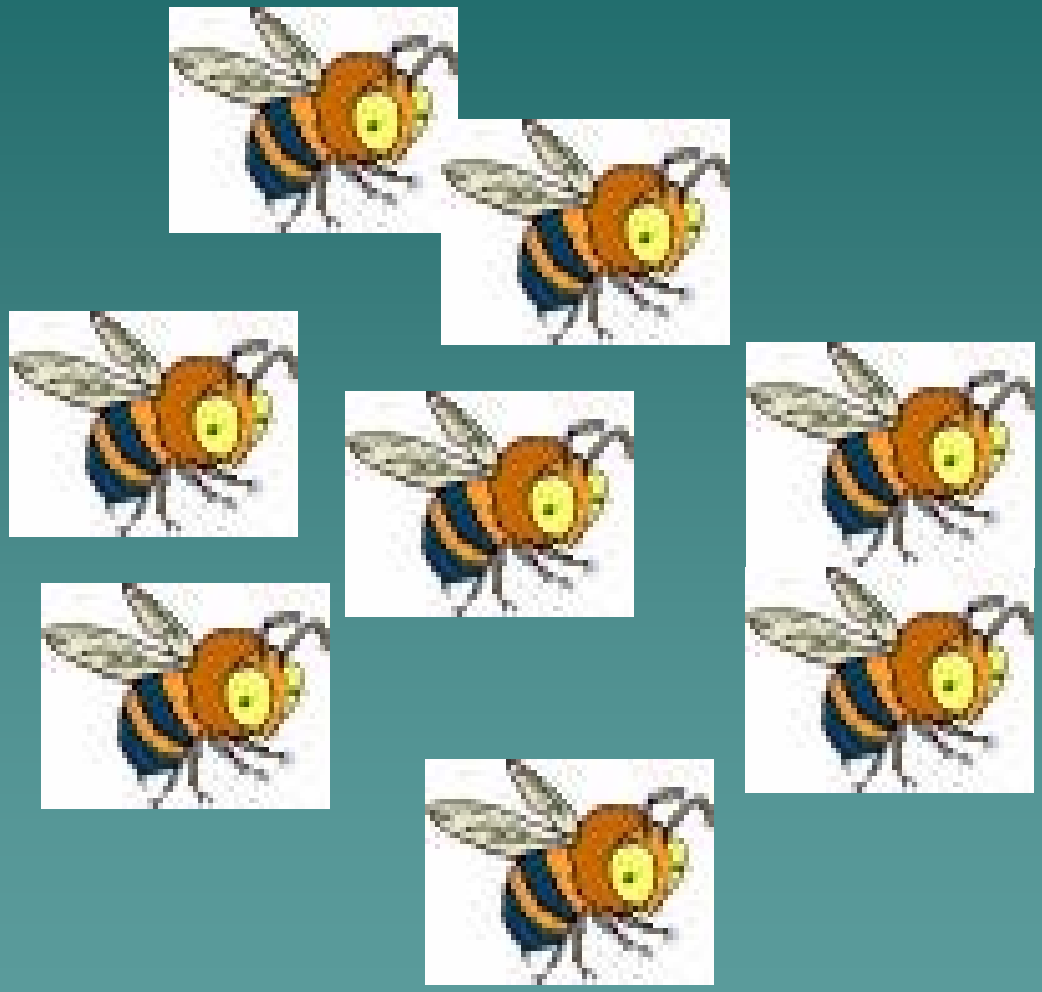


Happy Machine

Slow Output – What's wrong ?

- Performance was slow- just 5,000 matches/hour.
- Explain Plans were clean / all tables analyzed for CBO…
- Sad Machine… I/O slow, pushing no load to CPUs.
  - Dropped Block-Size, increased file-multiblock read count.
  - Reviewed LUNS-level breakouts of system for Hotspots and constraints
  - OFA lessons still apply- Huge, few LUNS were slowing throughput, and OS was rebuilt with many smaller size LUNS – blazing I/O gains.
- Pinning the CPUs means IO is meeting needs, and machine working as quickly as possible.
- Performance moves from 5,000 matches/ hour to 20,000 matches per hour

Send in the Swarm



Bring in the Swarm

- Still struggled to increase performance.
- Performance metric is measured by Inputs * Comparison Data. Noticed ‘Tipping Points’ of performance when testing at 1,000, 5,000 and 10,000 Inputs per Matching Run.
- CPU usage not consistent across all CPUs.
- Send in the Swarm- Divide Load across distinct OS-level processes, share data in SGA by running Concurrently
- Break up jobs of 100,000 x 36 million into 10 jobs of 10,000 each ' create a concurrent 'swarm'... reduces TEMP sorting collision, allows SGA re-use of same Candidate-side indexes at same time.
- Major increase in speed again... from 20,000 matches/hour to 100,000.