Querying

Building up our query technology

- Linear on-demand retrieval (aka grep)
- 0/1 Vector-Based Boolean Queries
- Posting-Based Boolean Queries

Building up our query technology

- Linear on-demand retrieval (aka grep)
- 0/1 Vector-Based Boolean Queries
- Posting-Based Boolean Queries

- How would it apply to
  - http://www.rhymezone.com/shakespeare/

Boolean Model vs. Ranked Retrieval Methods

* Only game for 30 years

* uses precise queries

* user decides relevance

* stayed current with proximity queries

* precise controlled queries

* transparent queries

* controlled queries

* Appeared with www

* uses “free-text” queries

* system decides relevance

* works with enormous corpora

* “no guarantees” in queries

Querying - Boolean Search Example

- Westlaw
  - Largest commercial (paying subscribers) legal search service (started in 1975, ranking added in 1992)
  - Tens of terabytes of data
  - 700,000 users
  - Majority of users still use boolean queries (default in 2005)
    - Example:
      - What is the status of limitations in cases involving federal tort claims act?
      - LIMIT! /3 STATUTE ACTION /S FEDERAL /2 TORT /3 CLAIM
      - /3 = within 3 words. /S same sentence

Querying - Boolean Search Example

- Westlaw
  - Example:
    - Requirements for disabled people to be able to access a workplace
    - disabl! /p access! /s work-site work-place employment /3 place
    - space is a disjunction not a conjunction
    - long precise queries, proximity operators, incrementally developed, not like web search
    - preferred by professionals, but not necessarily better

Building up our query technology

- “Matching” search
  - Linear on-demand retrieval (aka grep)
  - 0/1 Vector-Based Boolean Queries
  - Posting-Based Boolean Queries
- Ranked search
  - Parametric Search

Ranked Search

- Rather than saying
  - (query, document) matches or not (0,1)
    - (“Capulet”,”Romeo and Juliet) = 1
- Now we are going to assign rankings
  - (query, document) in {0,1}
    - (“capulet”,”Romeo and Juliet”) = 0.7

Querying

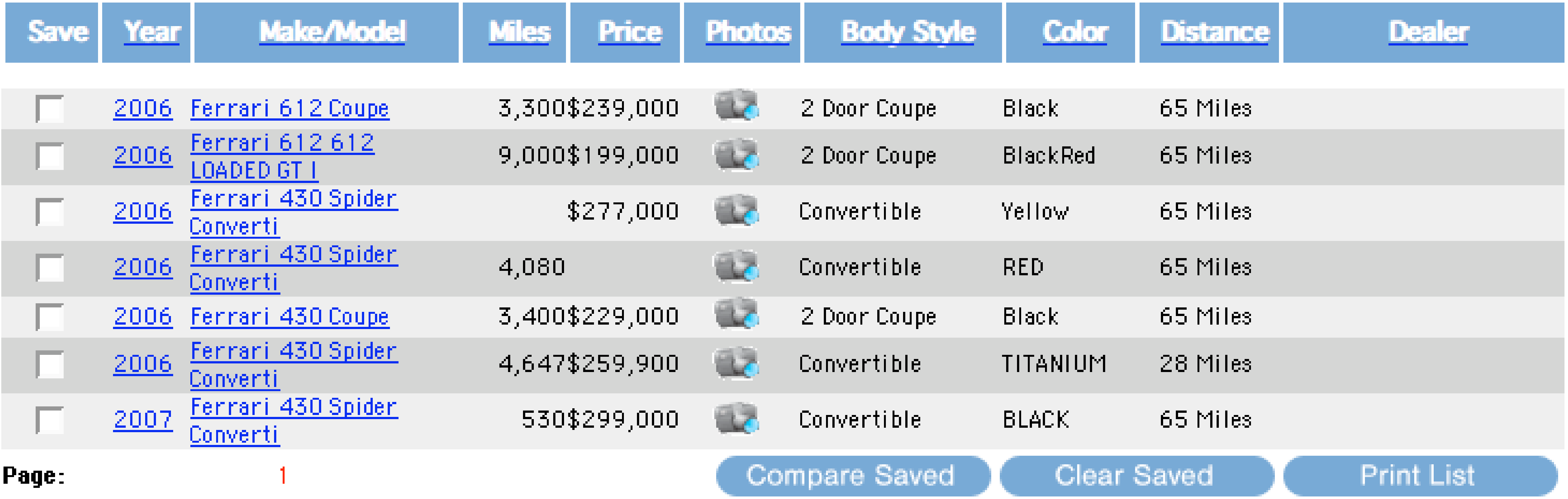
- Metadata = structured additional information about a document.
  - Examples:
    - The author of a document
    - The creation date of a document
    - The title of a document
    - The location where a document was created
  - author, creation date, title, location are fields
  - searching for “William Shakespeare” in a doc differs from
  - searching for “William Shakespeare” in the author of a doc

Querying

- Parametric Search
  - supports searching on meta-data explicitly
  - a parametric search interface allows a mix of full-text query and meta-data queries
  - Example:
    - www.carfinder.com

Querying

- Parametric Search
  - Example:
    - Result is a large table
    - Columns are fields
    - Searching for “2006” only applied to year field



Save	Year	Make/Model	Miles	Price	Photos	Body Style	Color	Distance	Dealer
□	2006	Ferrari 612 Coupe	3,300	$239,000	📷	2 Door Coupe	Black	65 Miles	
□	2006	Ferrari 612 612 LOADED GT I	9,000	$199,000	📷	2 Door Coupe	BlackRed	65 Miles	
□	2006	Ferrari 430 Spider Converti		$277,000	📷	Convertible	Yellow	65 Miles	
□	2006	Ferrari 430 Spider Converti	4,080		📷	Convertible	RED	65 Miles	
□	2006	Ferrari 430 Coupe	3,400	$229,000	📷	2 Door Coupe	Black	65 Miles	
□	2006	Ferrari 430 Spider Converti	4,647	$259,900	📷	Convertible	TITANIUM	28 Miles	
□	2007	Ferrari 430 Spider Converti	530	$299,000	📷	Convertible	BLACK	65 Miles	

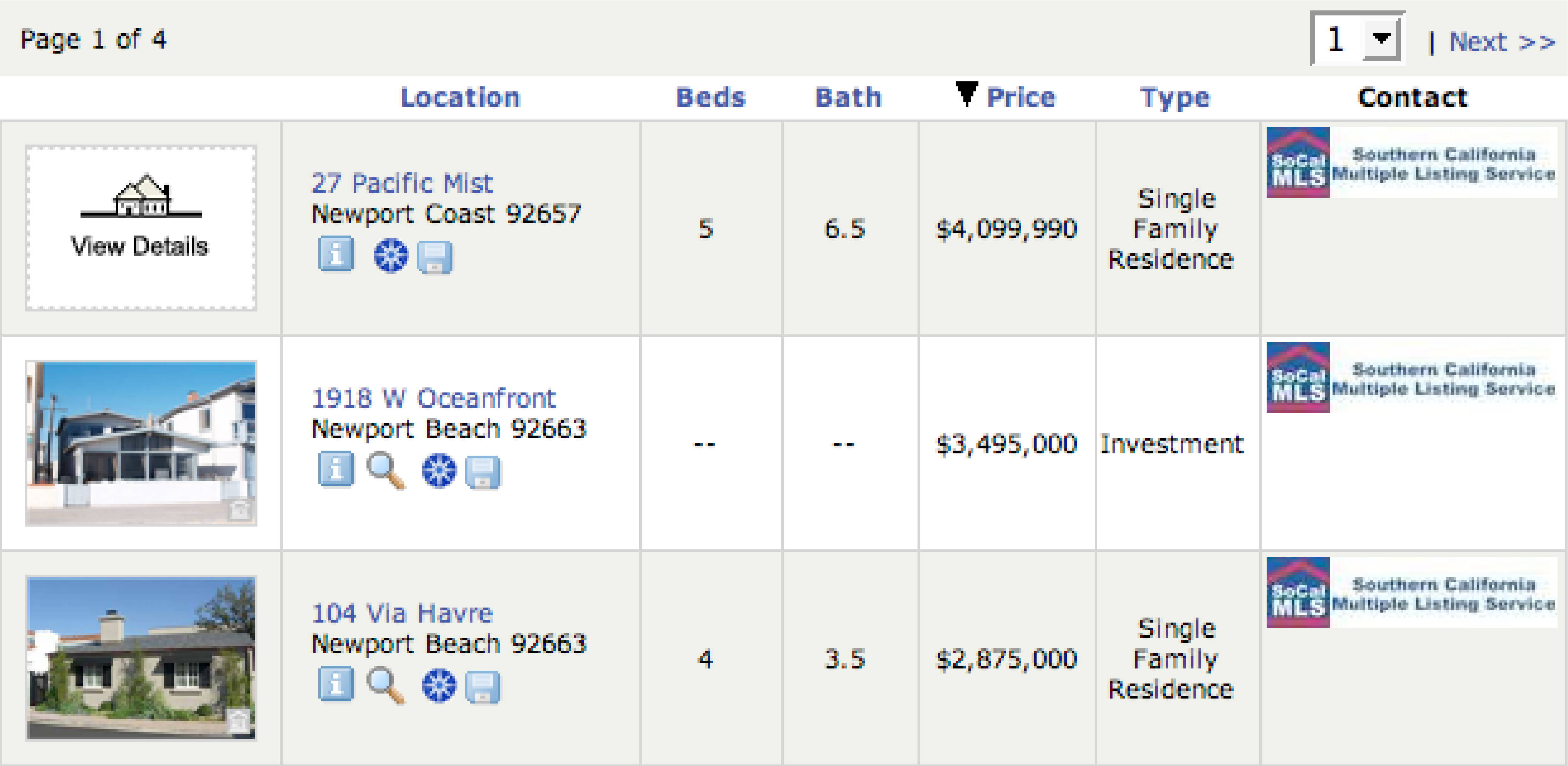
Page: 1

Querying

- Parametric Search
  - Example:
    - www.ocrealestatefinder.com

Querying

- Parametric Search
  - Example:
    - www.ocrealestatefinder.com

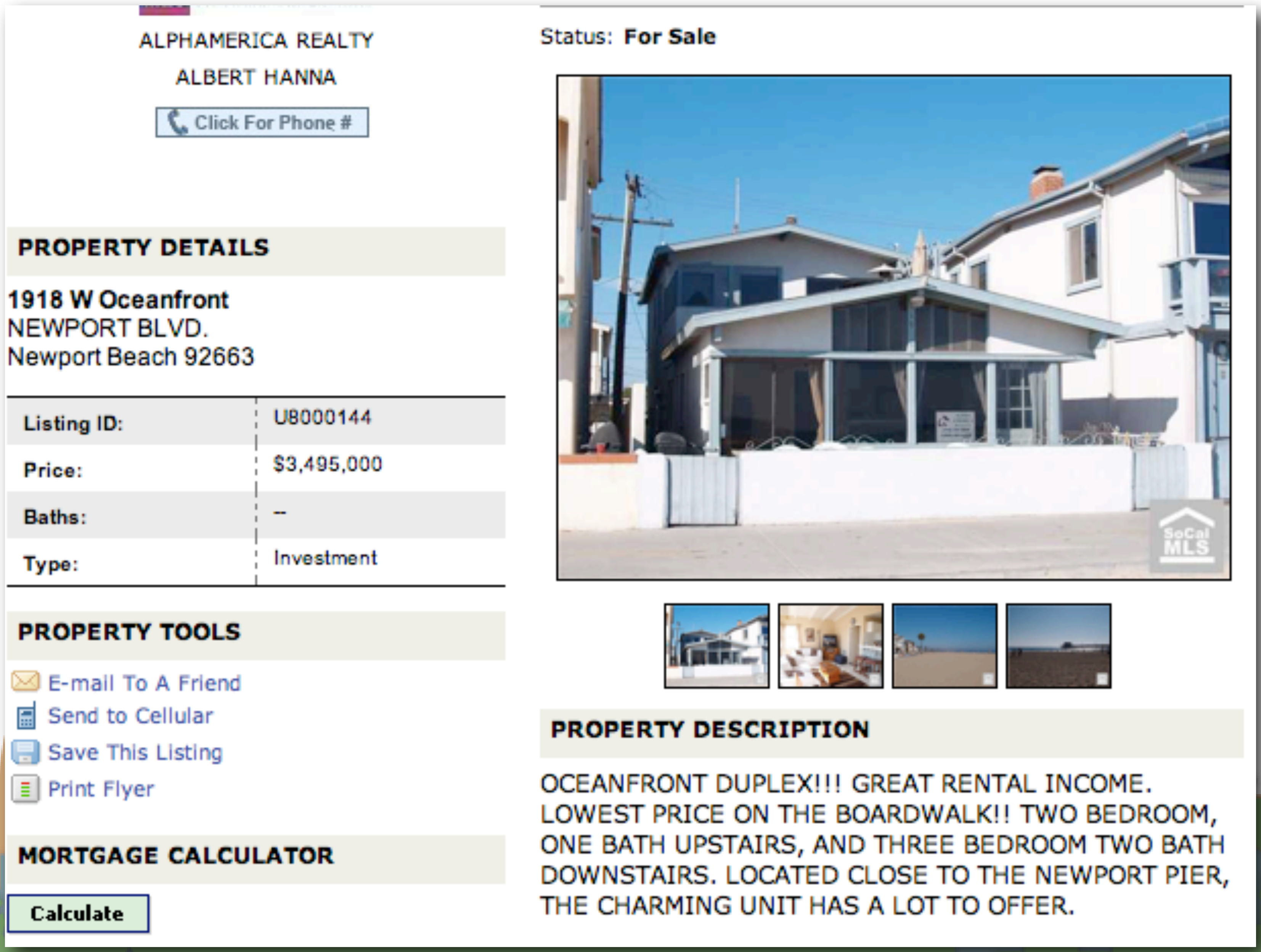


Page 1 of 4 | 1 | Next >>

	Location	Beds	Bath	▼ Price	Type	Contact
View Details	27 Pacific Mist Newport Coast 92657	5	6.5	$4,099,990	Single Family Residence	SoCal MLS Southern California Multiple Listing Service
	1918 W Oceanfront Newport Beach 92663	--	--	$3,495,000	Investment	SoCal MLS Southern California Multiple Listing Service
	104 Via Havre Newport Beach 92663	4	3.5	$2,875,000	Single Family Residence	SoCal MLS Southern California Multiple Listing Service

Querying

- Parametric Search
  - Example:
    - www.ocrealestatefinder.com
    - This one adds text search “charming”



ALPHAMERICA REALTY

ALBERT HANNA

Status: **For Sale**

PROPERTY DETAILS

1918 W Oceanfront
NEWPORT BLVD.
Newport Beach 92663

Listing ID:	U8000144
Price:	$3,495,000
Baths:	--
Type:	Investment

PROPERTY TOOLS

MORTGAGE CALCULATOR

PROPERTY DESCRIPTION

OCEANFRONT DUPLEX!!! GREAT RENTAL INCOME. LOWEST PRICE ON THE BOARDWALK!! TWO BEDROOM, ONE BATH UPSTAIRS, AND THREE BEDROOM TWO BATH DOWNSTAIRS. LOCATED CLOSE TO THE NEWPORT PIER, THE CHARMING UNIT HAS A LOT TO OFFER.

Parametric Search

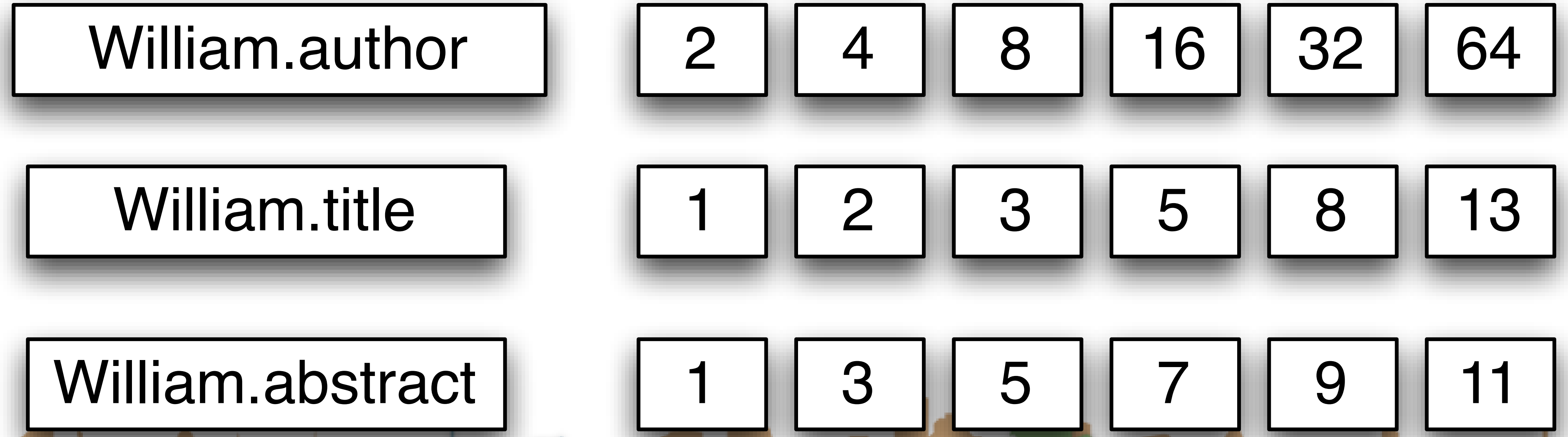
- In these examples we select field values
  - Values could be hierarchical
    - USA -> California -> Orange County -> Newport Beach
- It is a paradigm for navigating through a corpus
  - e.g, “Aerospace companies in Brazil” can be found by combining “Geography” and “Industry”
    - (“Capulet”,”Romeo and Juliet) = 1
- Approach:
  - Filter for relevant documents
  - Run text searches on subset

Parametric Search

- Index support for parametric search
  - Must be able to support queries of the form:
    - Find pdf documents that contain “UCI”
    - Field selection and text query
- Field selection approach
  - Use inverted index of field values
    - (field value, docID)
    - organized by field name
    - Using same compression and sorting techniques

Parametric Search

- Now, we crawl the corpus
- We parse the document keeping track of terms, fields and docIDs
- Instead of building just a (term, docID) pair
- We build (term, field, docID) triples
- These can then be combined into postings like this:



William.author	2	4	8	16	32	64
William.title	1	2	3	5	8	13
William.abstract	1	3	5	7	9	11

Parametric Search

- So are we just creating a database?
  - Not really.
  - Databases have more functionality
    - Transactions
    - Recovery
      - Our index can be recreated. Not so with database.
    - Text is never stored outside of indices
- We are focusing on optimized indices for text-oriented queries not a full SQL engine

Building up our query technology

- “Matching” search
  - Linear on-demand retrieval (aka grep)
  - 0/1 Vector-Based Boolean Queries
  - Posting-Based Boolean Queries
- Ranked search
  - Parametric Search
  - Zones

Zones

- A zone is an extension of a field
- A zone is an identified region of a document
  - e.g., title, abstract, bibliography
  - Generally identified by mark-up in a document
    - <title>Romeo and Juliet</title>
- Contents of zone are free text
  - Not a finite vocabulary
- Indices required for each zone to enable queries like:
  - (instant in TITLE) AND (oatmeal in BODY)
- Doesn’t cover “all papers whose authors cite themselves”
  - Why?

Querying

Building up our query technology

- “Matching” search
  - Linear on-demand retrieval (aka grep)
  - 0/1 Vector-Based Boolean Queries
  - Posting-Based Boolean Queries
- Ranked search
  - Parametric Search
  - Zones
  - Scoring

Scoring

- Boolean queries “match” or “don’t match”
- Good for experts with needs for precision and coverage
  - knowledge of corpus
  - need 1000’s of results
- Not good with non-expert users
  - who don’t understand boolean operators
  - or how they apply to search
  - or who don’t want 1000’s of results

Scoring

- Boolean queries require careful crafting to get the right number of results (Ferrari example)
- Ranked lists eliminate this concern
  - Doesn't matter how big the list is
- Scoring is the basis for ranking or sorting document that are returned from a query.
  - Ideally the score is high when the document is relevant
  - WLOG we will assume scores are between 0 and 1 for each doc.

Scoring

- First generation of scoring used a linear combination of Booleans

$$\begin{aligned} Score &= 0.6(instant \in TITLE) + \\ &\quad 0.3(oatmeal \in BODY) + \\ &\quad 0.1(health \in ABSTRACT) \end{aligned}$$

- Explicit decision about importance of zone
- Each subquery is 0 or 1
- This example has a finite number of possible values
  - What are they?

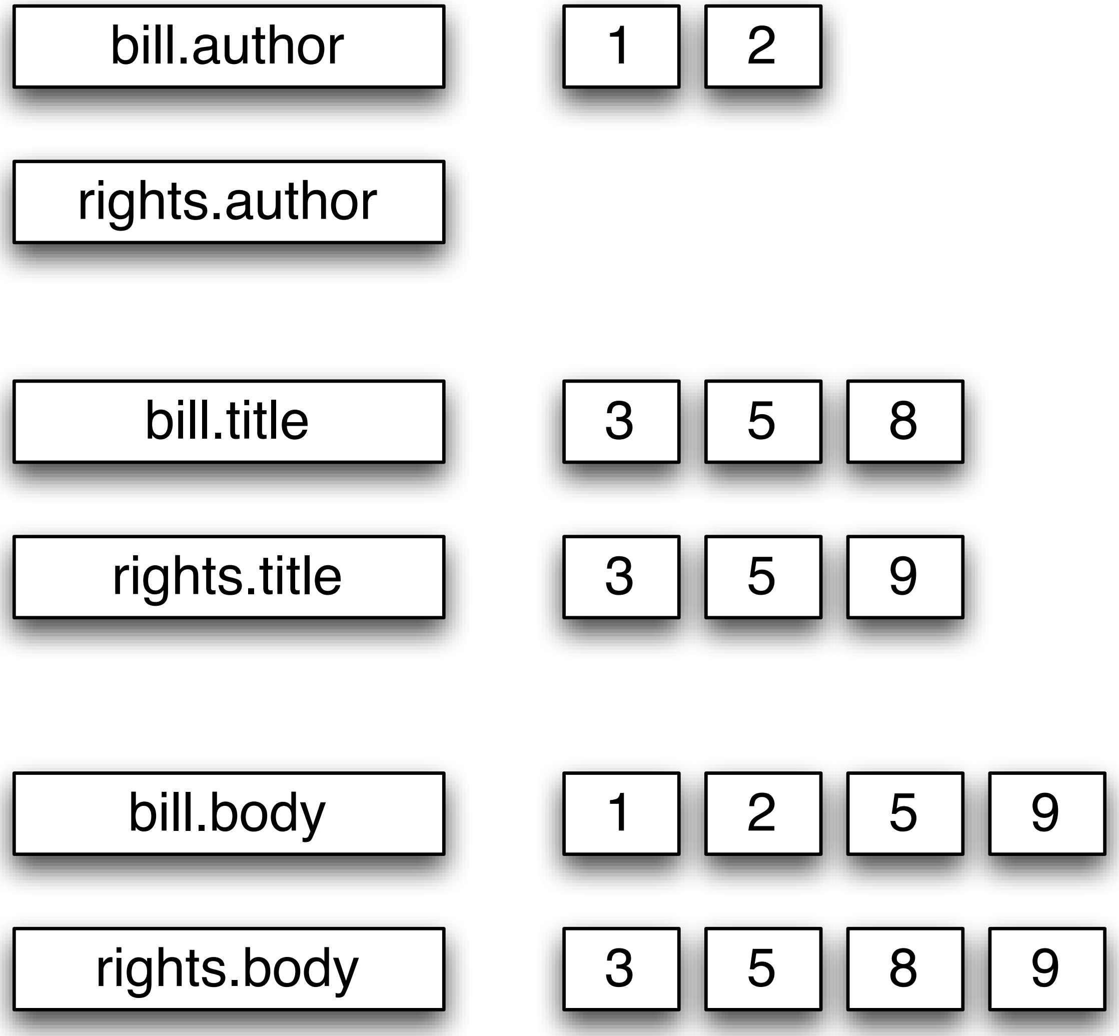
Scoring

$$\begin{aligned} Score \quad = \quad & 0.6(instant \in TITLE) + \\ & 0.3(oatmeal \in BODY) + \\ & 0.1(health \in ABSTRACT) \end{aligned}$$

- Subqueries could be *any* Boolean query
- Where do we get the weights? (e.g., 0.6,0.3,0.1)
    - Rarely from the user
    - Usually built into the query engine
        - Where does the query engine get them from?
            - Machine learning

Scoring Exercise

- Calculate the score for each document based on the weightings 0.6, 0.3, 0.1
- For the query
  - “bill” or “rights”



bill.author	1	2		
rights.author				
bill.title	3	5	8	
rights.title	3	5	9	
bill.body	1	2	5	9
rights.body	3	5	8	9