Databases and IR: Perspective of a SQL Guy

Surajit Chaudhuri
Data Management & Exploration
Microsoft Research
http://research.microsoft.com/users/surajitc
Acknowledgements

- Data Exploration Project Home: http://research.microsoft.com/dmx/Data_Exploration/
- Data Exploration Project at DMX MSR
  - Sanjay Agrawal, Gautam Das, Venky Ganti
- External Collaborators
  - Aris Gionis, Luis Gravano, Rajeev Motwani, Raghu Ramakrishnan, Gerhard Weilum
Outline

- 3 interesting applications for DBIR
  - MylifeBits (some details), Community Support (barely), Enterprise KM (barely)
- Requirements for a platform
- Query Model: IR-like issues in core RDBMS
- Conclusion
#1 MyLifeBits –
Gordon Bell, BARC, MSR

(adopted from Gordon’s longer presentation available at: http://research.microsoft.com/barc/mediapresence/MyLifeBits.aspx)
Charter: Memex
As We May Think, Vannevar Bush, 1945

“A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility”
The guinea pig

- Gordon Bell is digitizing his life
- Has now scanned virtually all:
  - Books written (and read when possible)
  - Personal documents (correspondence including memos and email, bills, legal documents, papers written, ...)
  - Photos
  - Posters, paintings, photo of things (artifacts, ...medals, plaques)
  - Home movies and videos
  - CD collection
  - And, of course, all PC files
- Now recording: phone, radio, TV (movies), web pages... conversations?
- Paperless throughout 2002
MyLifeBits: Some Lives(t)

- Personal
  - Parents, children, grandkids
  - CGB himself
  - Close friends

- GB $s
  - Personal incl. several legal structures
  - Investments & boards

- Past companies/organiz’ns
  - DEC
  - Carnegie-Mellon U.
  - DEC, NSF, Encore, Ardent, GB_consulting,
  - CGB@ Microsoft
    - MLB
    - Clusters
    - Telepresence
    - WWW presence

- Computer History Museum
  - BOD member
  - Fund-raising
  - CyberMuseum

- Startups
  - Bell-Mason Director
  - Diamond, & Vanguard Brds.
I mean everything
Input: tools, time, and cost

- Photos: $1 or 0.5-5 min.
- Large posters: ~ 1-5 hr.
  Artifacts: ~ 10 min. including photo
- Scanning to TIF, PDF: <1 min/page or .10/page
  - OCR: for PDF: ~3-5 pages/min (old data)
  - OCR: to recreate an editable “original” 10 min/page!
- OCR (Volume paper files): 400 pages/hr. 7 ppm.
- Books: scanned at CMU ($10 - 100/book) in 1997
Storage trends

- Right now, it affordable to buy 100 GB/year
- In 5 years you can afford to buy 1TB/year!
  (assuming storage doubles every 18 months)
gbell wag: 67 yr, 25Kday life
Trying to fill a terabyte in a year

- Gordon’s lifetime collection < 30 GB (12 GB is CDs)

<table>
<thead>
<tr>
<th>Item</th>
<th>Items/TB</th>
<th>Items/day</th>
</tr>
</thead>
<tbody>
<tr>
<td>300 KB JPEG</td>
<td>3.6M</td>
<td>9800</td>
</tr>
<tr>
<td>1 MB Doc</td>
<td>1.0M</td>
<td>2900</td>
</tr>
<tr>
<td>1 hour 256 kb/s MP3 audio</td>
<td>9.3K</td>
<td>26</td>
</tr>
<tr>
<td>1 hour 1.5 Mbp/s MPEG video</td>
<td>290</td>
<td>4</td>
</tr>
</tbody>
</table>

Surajit Chaudhuri, NSF IDM Talk, Seattle, 2003
Memory Overload

As hard drives get bigger and cheaper, we're storing way too much.

By Jim Lewis

There's a famous allegory about a map of the world that grows in detail until every point in reality has its counterpoint on paper; the twist being that such a map is at once ideally accurate and entirely useless, since it's the same size as the thing it's meant to represent.
So you’ve got it – now what do you do with it?

- Can you organize that many objects?
- Can you find anything?
- Once you find it will you know what it is?
- Once you’ve found it once, could you find it again?
Guiding Principles

1. Context of information of great value
   - Capture them automatically
   - Keep the links when you author
   - Make manual annotations easy!

2. Full text search & Collections
   - Freedom from strict hierarchy
     - May want more than a single parent, or may not want be bothered
   - Search in one place
   - Saved Queries in addition to fixed collections (find it again)

3. Many visualizations
   - For browsing, histograms and other aids,..
Value of media depends on annotations

- “It’s just bits until it is annotated”
System annotations provide base level of value

- Date 7/7/2000
Tracking usage – even better

- Date 7/7/2000. Opened 30 times, emailed to 10 people (it's valued by the user!)
Get the user to say a little something is a big jump

- Date 7/7/2000. Opened 30 times, emailed to 10 people. “BARC dim sum intern farewell Lunch”
Getting the user to tell a story is the ultimate in media value

- A story is a “layout” in time and space
- Most valuable content (by selection, and by being well annotated)
- Stories must include links to any media they use (for future navigation/search).
- Cf: MovieMaker; Creative Memories PhotoAlbums

Dapeng was an intern at BARC for the summer of 2000

We took him out to lunch at our favorite Dim Sum place to say farewell at the end of his internship

At table L-R: Dapeng, Gordon, Tom, Jim, Don, Vicky, Patrick, Jim
Requirements: MylifeBits

- Annotations/context capture is crucial
  - Money (transactions, payees, etc.)
  - Attributes for photos - Location, time, settings
  - Trips to cross-index to all docs
  - Presentations as a report or trail. Each slide an object!

- Search is a crucial component. But..
  - You may not know what you are looking for
  - Even with our best efforts, media will not be sufficiently annotated
    - Intelligent Browsing

- Database features are essential
  - Durability, Backup/Replication - guarantee that data will live forever!
  - Rich usage of schema
  - Indexing (multimedia), Queries, Scalability
  - Information control: privacy, security, delete

(End of Gordon’s great slides. Back to my boring slides...)
#2 Using Community for Product Support
Key Steps in using the Community Site

- Exploit hierarchy to isolate part of the relevant product information
- Use Search and review questions
- Ranked answer based on
  - Degree of match of content, Reputation rank of answer provider, timeliness, user profile
- Notification/subscription services for standing queries
- Integrated with CRM workflow
Requirements: Community for Product Support

- Structured attributes influence rank
  - Reputation in community
  - Classification of posting relative to query
  - Content
- “Posting” interface automatically captures structured attributes
#3 Enterprise Knowledge Management

- Example: Verity K2
- Taxonomy construction and maintenance
  - Assisted using automated tools: query/rule language, learning techniques,..  
- Search Millions of documents with ~10 structured attributes
  - Derived from text classification or context
  - Free-format search (not your rigid SQL)
- Personalization
  - Exploit past transactions/activities
  - Search + Recommend
- Crawls multiple sources

Surajit Chaudhuri, NSF IDM Talk, Seattle, 2003
Outline

- 3 interesting applications for DBIR
  - MylifeBits, Kanisha Compaq Community Support (barely), Enterprise KM (barely)
- Requirements for a platform
- Query Model: IR-like issues core RDBMS
- Conclusion
Observations

- Rich mixture of structured, text and media information
- Every usage required a custom-engine and custom set of APIs
  - Storage, Query layer and tools are all custom-built
- Current solution has a high TCO
  - Administration cost
  - Developer cost: Divergences in query model
Our Core Challenge

- Reducing total cost of ownership via consolidation of components
  - Identifying clear interfaces between tools/middleware, querying and storage layers
  - Storage and query layers should support multiple scenarios
Squashing two storage components?

- Reduce TCO
- Examine the stack of storage and query layers - different costs of tweaking
  - Lazy index updates (interesting similarity with QUIQ architecture)
- Tied to modularization of relational architectures
  - Hard to isolate modules
  - Chaudhuri and Weikum (VLDB 2000 vision paper):
    - RISC Architecture
Rest of the talk

- Identifying novel elements of the querying layer
  - Query functionalities
  - Query Execution Engine
Outline

- 3 interesting applications for DBIR
- Requirements for a platform
- **Query Model: IR-like issues in core RDBMS**
  - 2 Key issues (Just adopt the IR techniques?)
  - Implications for query execution
- Conclusion
Query Model: Two key differences

- Results are auto-ranked
  - ORDER BY AUTO!
- Schema-oriented vs. keyword queries
- Are these useful for database queries?
  - Auto-Ranking
    - Empty answers and many answers problem
    - Data cleaning
  - Keyword search (queries)
    - “Object” locator (table and column locator)
A vanilla example of today’s DB-IR integration

MS SQL Server FTS

- Core SQL Engine only supports LIKE (indexing support for prefix only)
  - Description LIKE "%XP%"
- Full Text Engine
Crawling, Index structures, Querying

- Indexes are stored in a compressed form (sacrifices update cost)
  - Uses stack indexes for efficiency
  - Indexer builds an inverted keyword list and persists
  - Sends notification back to SQL Server process
- Full/incremental/change tracking crawls
- Keyword match with options
  - Prefix/phrase/exact, Linguistic variations, Weighting of terms, proximity, Boolean composition, Request for ranks
Example of FTS Query (1)

- SELECT FT_TBL.CategoryName, FT_TBL.Description, KEY_TBL.RANK
  FROM Categories AS FT_TBL
  INNER JOIN FREETEXTTABLE(Categories, Description, 'sweetest candy bread and dry meat') AS KEY_TBL
  ON FT_TBL.CategoryID = KEY_TBL.[KEY]
  ORDER BY KEY_TBL.rank DESC
Query Model: Auto Ranking
Gordon’s Examples

- Find Gordon’s memos with title IN [Vax, VMS] and Year IN [1960, 1978]
  - What if Gord got the dates or titles or the combination wrong?
  - Or, if he had too many memos?
  - ..Empty and many answer problems
- Find person IN Gordon’s notes with meeting BETWEEN [1/1/01 2/1/01] AND organization = [Boeig Corporation]
  - Misspelling
  - data cleaning

Surajit Chaudhuri, NSF IDM Talk, Seattle, 2003
Other Examples

- Browsing for a home in homeadvisor/realtor database. Got no hits. How about returning nearest k results

- Business has a registry of customer names. A customer walks in. Is he a returning customer?
Next few slides are from..

- Chaudhuri, Ganjam, Ganti, Motwani, Robust and Efficient Fuzzy Match for Online Data Cleaning. ACM SIGMOD 2003, San Diego.
Empty Answers Problem

SELECT * FROM Homes WHERE Price = 325000 AND HasFence = true AND ExtColor = 'Purple'

May be preferable to return a few "partially matching" tuples
Leverage IR: Why not use TF-IDF?

- View tuples and queries as small documents and define similarity function between tuple and query

- TF-IDF Similarity:
  - IDF: Give less importance to frequently occurring query values
    - E.g. Bellevue less important than purple
  - TF: irrelevant in our case
Limitation 1: Inadequacy of IDF Weights

A data value may be important for ranking irrespective of its data frequency

- More homes built in Bellevue compared to Carnation; thus Bellevue has smaller IDF
- Yet demand for Bellevue homes is usually more than that for Carnation homes
Limitation 2: Binary Similarity between Data Values

- Need to have a non-binary gradation in similarity
  - $\text{SIM}_\text{City}(\text{Bellevue}, \text{Bellevue}) > \text{SIM}_\text{City}(\text{Bellevue}, \text{Redmond}) > \text{SIM}_\text{City}(\text{Bellevue}, \text{Seattle})$
Limitation 3: Numeric Data

- Binary similarity between numeric values is inappropriate
  - $\text{SIM}_{\text{Price}}(300000, 350000) > \text{SIM}_{\text{Price}}(300000, 400000)$

- Exact frequency (hence IDF and QF) for numeric data is meaningless
  - E.g., $\text{IDF}_{\text{Price}}(300000)$ should be small if there are many houses in the database whose prices are close to $300k$
Leveraging Workloads

- Gathering queries only is relatively easy using standard DBMS profiling tools.
- Recording ranked results from users is expensive.
- What can queries tell us?
**Query Frequency of Data Values**

- **Assumption:** The frequency of data values referenced in workloads, QF(value), is likely to indicate their importance in ranking.
Addressing Limitation 1: Improving IDF Weights

- Use the product $QF \times IDF$
  \[ W_{t_{\text{City}}}(\text{Bellevue}) = QF(\text{Bellevue}) \times IDF(\text{Bellevue}) \]

- This is similar to term frequency of original TF-IDF algorithm in IR
  - $QF(\text{Bellevue})$ is similar to TF of Bellevue in “query”
Addressing Limitation 2: Deriving Non-Binary Similarity

Assumption: If certain values often occur together in IN clauses, they are likely to be similar.
Deriving Non-Binary Similarity: Fuzzy Lookup

Edit distance not sufficient

- Reference set R1: [Boeing Company, Seattle, WA, 98004]
  - R2: [Bon Corporation, Seattle, WA, 98014]
  - R3: [XYZ Corporation, Seattle, WA, 98004]
- Input: I1: [Boeing Corporation, Seattle, WA, 98004]
  - I2: [Beoing Corporation, Seattle, WA, 98004]
- Edit distance: I1 mapped R2!
  - *Token importance*: some tokens are more important (IDF weights)
- Cosine Similarity with IDF weighting: I2 mapped to R2!
  - *Closeness* between tokens to be tolerant to input errors
- **Challenge**: Putting edit distance and IDF weights together
Addressing Limitation 3: Handling Numeric Data

- Need **smoothened** versions [CDG03] of
  - Similarity, frequency, IDF, QF, Weight

```sql
SELECT * FROM Homes WHERE Price = 300000
AND Bedrms = 4
```
Summary: Using IR Concepts for Auto Ranking

- IR metaphors need adaptations
- TF-IDF approach falls short
- Workload Analysis: Cheap and efficient way of capturing user preferences
  - Unified treatment of categorical and numeric data
- Not a replacement for domain knowledge
Query Model: Keyword Search in Structured World
Next few slides are from..

Keyword search on databases

- Why bother?
  - Object-locator
  - Give me information related to “Gray” and “Computer”
- How is it different from search over text documents?
An Example

Database Publications

Authors \( \xrightarrow{\text{Writes}} \) Titles \( \xrightarrow{} \) SoldIn \( \xrightarrow{} \) Stores

Publishers
Search string

Assume that in Publications
gray occurs in name of authors
computer occurs in title of titles

What is the expected answer?
Some Answer Sets

titles by authors with name gray and title computer
Some Answer Sets

titles by authors with name gray and title computer

stores that sell titles with title computer by authors name gray
Some Answer Sets

- **titles by authors** with name gray and title computer
- **stores that sell titles** with title computer by *authors* name gray
- **publishers of titles** with title computer by *authors* with name gray

Surajit Chaudhuri, NSF IDM Talk, Seattle, 2003
Some Answer Sets

- titles by 63 authors with name gray and title computer
- stores that sell titles with title computer by authors name gray
- publishers of titles with title computer by authors with name gray
- stores and publishers of titles with title computer by authors with name gray
Our answer

- Titles by authors with name gray and title computer
- Stores that sell titles with title computer by authors name gray
- Publishers of titles with title computer by authors with name gray
- Stores and publishers of titles with title computer by authors with name gray

Surajit Chaudhuri, NSF IDM Talk, Seattle, 2003
Differences from Text Docs

- Information resides in different tables in databases
  - Naïve approach of treating each row as a document does not work as such
  - Results need to be constructed on the fly
- Ranking needs to be done on results constructed dynamically as well
Outline

- 3 interesting applications for DBIR
- Requirements for a platform
- **Query Model: IR-like issues in core RDBMS**
  - 2 Key issues - Just adopt the IR techniques?
  - Implications for query execution
- Conclusion
Query Execution Challenges

- Keyword Search queries
  - Structure dereferencing
- Top-K matches for each attribute
  - Maintain auxiliary information (workload, link)
  - Need for efficient “error-tolerant” indexing for substring matching
- Top-K matches for a record
  - Ranking engine should be customizable
  - Leverage Monotonicity of combination function
Structure Dereferencing: Symbol Table

- For a keyword, tells the locations where it occurs in the database
  - Critical - Must provide fast lookup
  - Easy to build and maintain

- Design Decisions
  - What structure to use - relational table, custom?
  - What does location mean in context of databases? How does it affect search performance?
## Location Options: Pub-Col vs. Pub-Cell

| Symbol Table Size | Pub-Col an order of magnitude smaller  
|                  | Only distinct values in some column stored |
| Building Time    | Pub-Col takes much less time to build  
|                  | Much less data (only distinct values in column) brought into application |
| Maintenance      | Pub-Cell maintenance costlier  
|                  | Pub-Col updated only if new values get added to some column or a value gets deleted |
| Search performance | If indexes present on base table  
|                 | Pub-Col is faster otherwise Pub-Cell is faster |
Compress Each Clique
Search: 3 steps

1. Identify matching Table.Column for each keyword - Symbol Table Look up

2. Join tree enumeration
   - Ear removal on schema graph
   - Breadth first enumeration of join trees
     - Select keyword matching fewest number of Tables
     - Anchor search on Tables that match this keyword

3. Map Join Trees to SQL and Execute SQL to get results
Ranking System Architecture

Ranking System Architecture Diagram:
- **Database**
- **Tables**
- **Ranking function**
- **Workload**
  - **Offline (Preprocessing)**
    - **Extract ranking function**
  - **Online (Query Processing)**
    - **Incoming query**
    - **Execute Top-K algorithm**
    - **Top-K tuples**
Results of Ranking

Execute Top-K Algorithm

```
SELECT * FROM Homes WHERE City = 'Bellevue'
AND ExtColor = 'Purple'
```

<table>
<thead>
<tr>
<th>HomeID</th>
<th>Price</th>
<th>Bedrms</th>
<th>City</th>
<th>ExtColor</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td>274,000</td>
<td>3</td>
<td>Redmond</td>
<td>Purple</td>
</tr>
<tr>
<td>12</td>
<td>512,500</td>
<td>4</td>
<td>Seattle</td>
<td>Purple</td>
</tr>
<tr>
<td>811</td>
<td>375,300</td>
<td>3</td>
<td>Bellevue</td>
<td>White</td>
</tr>
<tr>
<td>311</td>
<td>280,000</td>
<td>4</td>
<td>Bellevue</td>
<td>Yellow</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Ranked Top-K answers
Error Tolerant Indexing for Fuzzy Match Similarity

- \textit{tokens} \rightarrow \{\textit{sub-tokens}\} to approximate closeness
  - Similar tokens share sub-tokens (e.g., q-grams)
  - Set of sub-tokens \rightarrow Min-hash signature vectors (efficiency)
  - Index tuples on min-hash signatures

- Example: [Microsoft Corp, Redmond, WA, 98052]
  \rightarrow \{\text{Micr,icro,cros,roso,osof,soft,corp}\}, \{\text{redm,edmo,dmon,mond}\}, \{\text{wa}\}, \{\text{[9805, 8052]}\}
  \rightarrow \{\{\text{Micr, osof}\}, \{\text{corp}\}\}, \{\{\text{redm, mond}\}\}, \{\{\text{WA}\}\}, \{\{\text{[9805, 8052]}\}\}
Combined Ranking Functions

- Objective: Use traditional SQL DBMS with minimal changes
  - No new access method
- Traditional SQL Top-K approach results in linear scans
  - Evaluate ranking function for each tuple and then sort
- Can we use index lookups?
  - No benefit if we must look at > 10% of tuples
  - Question: How can we avoid evaluating ranking function for the rest?
Exploiting Monotonicity

- Our ranking functions are “monotonic”
- Fagin’s **Threshold Algorithm** may avoid looking at all tuples
  - For a given query, get tuples that are top-ranked for each attribute
    - Closest prices
    - Closest cities
  - Winner can be found from such “sorted streams”
  - Indexes and Materialized view can implement such sorted streams
Example using Threshold Algorithm

SELECT * FROM Homes WHERE Price = 300000 AND City = 'Bellevue'

<table>
<thead>
<tr>
<th>HomeID</th>
<th>Price</th>
<th>Bedrms</th>
<th>City</th>
<th>…</th>
<th>ExtColor</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td>274,000</td>
<td>3</td>
<td>Redmond</td>
<td>…</td>
<td>Purple</td>
</tr>
<tr>
<td>12</td>
<td>512,500</td>
<td>4</td>
<td>Seattle</td>
<td>…</td>
<td>Purple</td>
</tr>
<tr>
<td>811</td>
<td>375,300</td>
<td>3</td>
<td>Bellevue</td>
<td>…</td>
<td>White</td>
</tr>
<tr>
<td>311</td>
<td>280,000</td>
<td>4</td>
<td>Bellevue</td>
<td>…</td>
<td>Yellow</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

- Closest Price: Return 311, 46, 811, 12, ...
- Closest City: Return 811, 311, 46, 12, ...

Surajit Chaudhuri, NSF IDM Talk, Seattle, 2003
Outline

- 3 interesting applications for DBIR
- Requirements for a platform
- Query Model: IR-like issues in core RDBMS
  - 2 key issues - Just adopt the IR techniques?
  - Implications for query execution
- Conclusion
Final Thoughts

- Look for horizontal layering of functionality as generic as possible
  - Cannot have too many engines due to TCO

- Querying: Enable Top-K matches in relational world
  - Auto-ranked from multiple sources
  - Exploit structure dereferencing for keyword search queries, error-tolerant indexes, and monotonicity of ranking functions in the Engine

- Discover from freetext
  - Shallow information extraction from text documents driven by a schema (analogy with DM)
  - Isolate link properties among documents

- A bit closer to Gordon’s needs but not quite there..