Qunits: Queried Units in Database Search

Arnaud Nandi\(^1\) and H. V. Jagadish\(^2\)

\(^1\)Computer Science and Engineering, \(^2\)Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI

---

**Motivation**

- Keyword Search in databases
  - Simplicity is important
  - Keyword Search works
  - SQL / Query does not

- Consider search “PWP1”
  - Ambiguous information need
  - User may not how to search
  - May not know what DB has

- There is a need to define exactly what we expect as a result

---

**Qunits**

- The user has a “mental model” of how a database is organized
  - It does not need to correspond with the internal schema

- Our goal is to guide the user to their information need

- Qunit: Queried Unit
  - Basic, independent semantic unit of information in a database
  - Atomic piece of information to be returned for a query

- A small number of Qunit Definitions exist
  - e.g. “interactions”

- When applied to the database, they generate Qunit Instances
  - e.g. “list of interactions with CKM…FN1L, GAMT, MRCC…”

- Qunits are ranked based on their “Qunit Utility”
  - The utility of a qunit is the importance of a qunit to a user query, in the context of the overall intuitive organization of the database.

---

**The Qunit Paradigm**

- Derive Qunits
  - Index qunits (just like documents)

- Process Query
  - Search over collection
  - return best qunit(s)

- Example, for MIMI:
  - Analyze MIMI database
  - Derive Qunit definitions
    - Interactions
      - Genomic Info
      - Pathways
      - Protein info
  - For search “CK”
    - Look for closest Qunit
      - [gene.name] usually matches
        - Genomic Info qunit
      - return “CK” Genomic Info qunit that closest matches the query string “CK”

- Example Qunit-based result for query “CK”

- Why are Qunits better?
  - Search Quality
    - Predefined qunits = meaningful results
    - Comparing 2 results makes sense
  - Data Integration
    - Search across multiple databases = solving multiple search problems at the same time
    - Each database outputs qunits, which are put into a unified search pool

---

**Qunit Derivation**

- Human generated
  - Ideal, but not tractable

- Schema & Data
  - Look at Schema
  - Query rollup
    - Look at existing query logs
  - External Evidence
    - Look at published instances of results

- Qunit Derivation: Query Rollup
  - Look outside of the database
  - Great when data + schema are not sufficient

  - “The result of a query is the union of all specialized(i.e. stricter) versions of that query”

  - \[gene\] :=
    - SELECT gene FROM gene X X gene as G WHERE G.name = "$"
    - SELECT location FROM gene WHERE gene.name = "$"
    - [gene.name] isoenzymes
    - SELECT organism.location FROM gene X organism

---

**Evaluation**

- IMDB Dataset
  - 14 tables, 54M rows.

- Query workload
  - 26 keyword queries, 13 types
  - Most popular types in AOL query log

- Perform search using each query algorithm
  - Results “normalized” to plaintext

- Classify each result: does this satisfy the query
  - 5 possible classes

---

**User Consensus**

- Relevant Classes
  - Highest frequency of results in all classes
  - Highest frequency of results in all classes

---

**Acknowledgements**

This work was supported by National Institutes of Health: Grant #K08DA032177 and a Yahoo Research Fellowship.

---

Further details

Qunits are part of the Database Usability project: Further details are available at [http://www.eecs.umich.edu/db/usable](http://www.eecs.umich.edu/db/usable)