CrowdDB: Answering Queries with Crowdsourcing

Michael Franklin et al., SIGMOD’11

Lianghong Xu
CMU 15-799 course presentation
Power to the People

• Some queries cannot be easily answered by machines
  – Which picture shows the golden bridge better?

• Human labor can be better and cheaper
  – Sometimes humans are smarter than computers
  – Ability to find new data in the open world
CrowdDB: Intuition

• Complement traditional database systems with human knowledge, whenever needed

• Leverage the best from both sides
  – Human power for comparing and finding data
  – Machine power for heavy-lifting computation

• Automate task assignment
• Crowdsourcing platform
• CrowdSQL
• User interface
• Query processing
• Evaluation
Crowdsourcing platform: AMT

• Amazon Mechanical Turk

• Basic concepts
  – HIT: Human Intelligent Task. Smallest entity of work
  – Assignment: 1 HIT replicated to N assignments
  – HIT group: similar HITs

• Mechanical Turk APIs
  – Requester: createHIT(), approve/rejectAssignment()
  – Worker: getAssignmentForHIT()
Design Considerations

• Performance and variability
  – Difference in worker productivity

• Task design and ambiguity
  – Need user-friendly interface

• Affinity and learning
  – Workers learn and become more experienced

• Relatively small worker pool

• Open vs. closed world
• Crowdsourcing platform
• CrowdSQL
• User interface
• Query processing
• Evaluation
CrowdSQL

• Superset of SQL
  – With minimal extension to support crowdsourcing
  – Expressive semantics

• Incomplete data
  – Crowdsourse new column/record if not exist

• Subjective comparisons
  – Compare/order values
Example: Crowdsourced Column

```sql
CREATE TABLE Department (  
    university STRING,  
    name STRING,  
    url CROWD STRING,  
    phone STRING,  
    PRIMARY KEY (university, name) );

SELECT url FROM Department WHERE name = "Math";
```
Example: CROWDORDER

CREATE TABLE picture ( 
  p IMAGE, 
  subject STRING); 

SELECT p FROM picture 
WHERE subject = "Golden Gate Bridge" 
ORDER BY CROWDORDER(p, 
"Which picture visualizes better %subject");
Potential issues in CrowdSQL

• Unbounded cost and latency
  – #items to be crowdsourced is unclear
  – Can be mitigated by setting query “budget”

• Lineage
  – Track source of data to take actions

• Cleansing of crowdsourced data
  – Reduce data redundancy due to human input
• Crowdsourcing platform
• CrowdSQL
• User interface
• Query processing
• Evaluation
User interface

• Create templates in compile-time

• Automatically instantiate user interface templates in the runtime

• Can be edited for customized instructions
Basic Interface

(a) Crowd Column & Crowd Tables w/o Foreign Keys

(b) CROWDEQUAL

(c) CROWDORDER
Multi-Relational Interfaces

(e) Foreign Key (denormalized)
Possible Optimizations

• Batch tuples
  – The same person for many similar tasks

• Prefetch attributes
  – Crowdsources more than needed for future use
• Crowdsourcing platform
• CrowdSQL
• User interface
• **Query processing**
• Evaluation
Query Processing

• Extended operators to support CrowdSQL
  – CrowdProbe, CrowdJoin, CrowdCompare
• Create HIT (group) using AMT APIs
• Parse crowdsourced results
  – Perform majority-based quality control
• Rule-based optimizer
  – Basic parameter setting (e.g., price, batching size)
  – Better candidate: cost-based optimizer
• Crowdsourcing platform
• CrowdSQL
• User interface
• Query processing
• Evaluation
Response Time

Figure 4: Response Time (min): Vary Hit Group (1 Asgn/HIT, 1 cent Reward)
Completion with Varied Reward

Figure 6: Completion (%): Vary Reward (100 HITs/Group, 5 Asgn/HIT)
HITs/Quality by Worker

Figure 8: HITs/Quality by Worker (Any HITs/Group, 5 Asgn/HIT, Any Reward)
Interesting Observations

• Crowdsourcing involves long-term relationship
  – Keep the workers happy

• Interface design matters
  – A good interface improves result quality and worker efficiency

• Need of data independence
  – Free application writers from worrying about changing environment