CrowdDB: Answering Queries with Crowdsourcing


Presented by Patricia Ortega

February/2013
Outline

• Introduction
• Problem definition
• Crowdsourcing
• CrowdDB
• User Interface Generation
• Query Processing
• Experiment and Results
• Conclusion
Introduction

What do they have in common?
### Problem definition

<table>
<thead>
<tr>
<th>company_name</th>
<th>logo</th>
<th>revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>🍎 Apple</td>
<td>US$ 156.508 billion</td>
</tr>
<tr>
<td>Intelligence Bussines Machine</td>
<td>IBM</td>
<td>US$ 106.916 billion</td>
</tr>
<tr>
<td>Microsoft</td>
<td>Microsoft</td>
<td>US$ 73.72 billion</td>
</tr>
</tbody>
</table>

SELECT revenue FROM company WHERE name = “I.B.M”;

**Entity resolution problem**
## Problem definition

<table>
<thead>
<tr>
<th>company_name</th>
<th>logo</th>
<th>revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td><img src="image" alt="Apple Logo" /></td>
<td>US$ 156.508 billion</td>
</tr>
<tr>
<td>Intelligence Bussines Machine</td>
<td><img src="image" alt="IBM Logo" /></td>
<td>US$ 106.916 billion</td>
</tr>
<tr>
<td>Microsoft</td>
<td><img src="image" alt="Microsoft Logo" /></td>
<td>US$ 73.72 billion</td>
</tr>
</tbody>
</table>

SELECT revenue FROM company WHERE name = “Google”;

**Closed world assumption**
Problem definition

<table>
<thead>
<tr>
<th>image</th>
<th>relevance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SELECT image FROM picture WHERE topic = "Business Success" ORDER BY relevance LIMIT 1;
Problem definition

<table>
<thead>
<tr>
<th>company_name</th>
<th>logo</th>
<th>revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td><img src="image" alt="Apple Logo" /></td>
<td>US$ 156.508 billion</td>
</tr>
<tr>
<td>Intelligence Bussines Machine</td>
<td><img src="image" alt="IBM Logo" /></td>
<td>US$ 106.916 billion</td>
</tr>
<tr>
<td>Microsoft</td>
<td><img src="image" alt="Microsoft Logo" /></td>
<td>US$ 73.72 billion</td>
</tr>
</tbody>
</table>

```sql
SELECT revenue FROM company
WHERE name = "The best software company to work at";
```

I guess your answer was "Google". Is that the answer from the crowd?
Crowdsourcing

Two main human capabilities that allow correct answers:

- Finding new data
- Comparing data
Crowdsourcing

A requester has work to be done. → The problem is broadcast online → Crowd is asked for a solution

Crowd reply their solutions → Requesters approve or reject → Requesters pay the pre-defined reward
Crowdsourcing – Mechanical Turk Basics

**Microtasks:** No requires special training, typically less than a minute.

**HIT (Human Intelligent Task):** The smallest entity of work that could be accepted by a worker.

**Assignment:** HIT can be replicated into multiple assignments. A worker can process at most a single assignment per HIT.

**HIT Group:** AMT automatically groups HIT’s by requester, title, description and reward.
AMT Workflow

- Requesters post HITs.
- AMT post them into compatible HIT groups.
- Worker search, accept and process the assignment.
- Requesters approve or reject.
- For each task completed requesters pay the predefined reward, bonus and commission to Amazon.
Mechanical Turk APIs

Create new HIT:
- `createHit(tittle, description, question, keywords, reward, duration, maxAssignments, lifetime): HitId`

List of assignments of a HIT
- `getAssignmentsForHIT(HitId): list(ansId, workerId, Answer)`

Approve/Reject
- `approveAssignment(ansId)/rejectAssignment(ansId)`
CrowdDB – Design Considerations

- Performance and variability
  - Work speed
  - Work cost
  - Work quality
- Task design and ambiguity
  - Natural language ambiguity
  - UI Design
CrowdDB – Design Considerations

• Affinity and learning
  • Workers develop skills, and relationships with requesters.
• Relatively small worker Pool
  • Impact in parallelism and throughput
• Open vs. closed world
  • Possible return unlimited number of answers.
    (Query planning, execution cost, answer quality)
CrowdDB - Architecture
Crowd Components

Turker Relationship Manager:
• Handles: approving/rejecting assignments, paying, etc.

User Interface Management:
• CrowdSQL extends data definition language to annotate tables, information used later to create UI.

HIT Manager:
  Manages interaction CrowdDB and crowdsourcing platform
CrowdSQL

Is a SQL extension that support crowdsourcing.

- Minimal extension
- Support use case with missing data and subjective comparisons.
CrowdSQL - Considerations

SQL DDL extensions
Keyword CROWD:

- Incomplete data can occurs:
  - Specific attributes of tuples
  - Entire tuple

**Crowdsourced column**
CREATE TABLE Department (  
  university STRING,  
  name STRING,  
  url CROWD STRING, phone STRING,  
  PRIMARY KEY (university, name) );

**Crowdsourced Table**
CREATE CROWD TABLE Professor (  
  name STRING PRIMARY KEY,  
  email STRING UNIQUE,  
  university STRING, department STRING,  
  FOREIGN KEY (university, department)  
  REF Department(university, name) );
CrowdSQL - Considerations

SQL MDL semantics

Keyword CNULL:

• Equivalent to NULL
• Means that value should be crowd sourced at its first use.
• Default value of CROWD column

```
INSERT INTO Department(university, name)
VALUES ("UC Berkeley", "EECS");
```

```
INSERT INTO Department(university, name, url)
VALUES ("ETH Zurich", "CS", "inf.ethz.ch");
```
CrowdSQL - Considerations

Query semantics

- Support any kind of query on CROWD tables and columns.
- Incorporates crowdsourced data as part of processing SQL queries.

```
SELECT url FROM Department WHERE name = "Math";
SELECT * FROM Professor WHERE email LIKE "%berkeley%" AND dept = "Math";
```
CrowdSQL – Subjective comparisons

To support subjective comparisons has to be built in functions (CROWDEQUAL and CROWDORDER):

- CROWDEQUAL : ~=(takes 2 parameters lvalue, rvalue, ask the crowd to decide if values are equals)

```sql
SELECT profile FROM department WHERE name ~= "CS";
```
CrowdSQL – Subjective comparisons

- **CROWORDER**: Used to ask crowd rank the result.

```sql
CREATE TABLE picture (
  p IMAGE,
  subject STRING);

SELECT p FROM picture
WHERE subject = "Golden Gate Bridge"
ORDER BY CROWORDER(p,
  "Which picture visualizes better %subject")
```
User Interface Generation

Key: Provide effective user interfaces.

(a) Crowd Column & Crowd Tables w/o Foreign Keys
(b) CROWDEQUAL
User Interface Generation

UI key to success in crowdsourcing:

• At compile time, creates templates to crowdsourcing missing information (HTML5, JavaScript)
• These templates are instantiated at runtime providing a UI for a concrete tuple or set of tuples.
User Interface Generation

Key: Provide effective user interfaces.

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

(b) CROWDEQUAL
User Interface Generation

Key: Provide effective user interfaces.
User Interface Generation

Key: Provide effective user interfaces.

(e) Foreign Key (denormalized)
Current version of CrowDB has three crowd operators:

- CrowdProbe:
  Crowd missing information about CROWD columns and new tuples. (Uses generated UI)
Query Processing – Crowd Operators

• CrowdJoin:
  • Implement an index nested-loop-join over two tables (at least one crowdsourced).
  • Creates HIT’s for each tuple in the inner relation.
Query Processing – Crowd Operators

- CrowdComprare:
  - Implements CROWDEAQUAL and CROWDORDER.
  - Instantiate UI.
  - Typically used inside another traditional operator (sorting or predicate evaluation).
Query Processing – Plan Generation

(a) PeopleSQL query

SELECT *
FROM professor p,
    department d
WHERE p.department = d.name
    AND p.university = d.university
    AND p.name = "Karp"

(b) Logical plan before optimization

(c) Logical plan after optimization

(d) Physical plan

Figure 3: CrowdSQL Query Plan Generation
Experiments and Results

Experiments run with CrowdDB and AMT. Ran over 25,000 HITs on AMT during October 2010

- Parameters:
  - Price
  - Jobs per HIT and
  - Time of day.

- Measured the response time and quality of the answers provided by the workers.
Experiments and Results

Micro-benchmarks:

- Simple jobs involving finding new data or making subjective comparisons.
- Goal:
  - Observe the behavior of workers for the types of tasks required.
  - Obtain insight to develop costs models for query optimization.
Experiments and Results - Micro Benchmarks

- Description: Simple tasks requiring workers to find and fill in missing data for a table with two crowdsourced columns:

```sql
CREATE TABLE businesses (  
    name VARCHAR PRIMARY KEY,
    phone_number CROWD VARCHAR(32),
    address CROWD VARCHAR(256));
```
Experiments and Results - Micro Benchmarks

- Table was populated with names of 3607 businesses (restaurants, hotels, and shopping malls) in 40 USA cities.
- Study the sourcing of the phone_number and address columns using the following query:

```
SELECT phone_number, address FROM businesses;
```
Experiment 1: Response Time, Vary HIT Groups

Time of completion of 1,10,25 group HIT size.

Response time decrease dramatically as size of group is increased.

Figure 4: Response Time (min): Vary Hit Group (1 Asgn/HIT, 1 cent Reward)
Experiment 2: Responsiveness, Vary Reward

How response time varies as a function of the reward.

Figure 6: Completion (%): Vary Reward (100 HITs/Group, 5 Asgn/HIT)
Experiment 2: Responsiveness, Vary Reward

Fraction of HITs that received at least one assignment as a function of time and reward

Figure 7: Completion (%): Vary Reward (100 HITs/Group, 5 Asgn/HIT)
Experiment 3: Worker Affinity and Quality

Number of HITs computed for a particular worker and the number of errors made for the worker.

Figure 8: HITs/Quality by Worker (Any HITs/Group, 5 Asgn/HIT, Any Reward)
Complex Queries: Entity Resolution on Companies

<table>
<thead>
<tr>
<th>Non Uniform Name</th>
<th>Query Result</th>
<th>Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayerische Motoren Werke</td>
<td>BMW</td>
<td>3</td>
</tr>
<tr>
<td>International Business Machines</td>
<td>IBM</td>
<td>2</td>
</tr>
<tr>
<td>Company of Gillette</td>
<td>P&amp;G</td>
<td>2</td>
</tr>
<tr>
<td>Big Blue</td>
<td>IBM</td>
<td>2</td>
</tr>
</tbody>
</table>

SELECT name FROM company WHERE name~="[a non-uniform name of the company]"

Figure 9: Entity Resolution on Company Names
Complex Queries: Ordering Pictures

Figure 10: Pictures of the Golden Gate Bridge [1] ordered by workers. The tuples in the sub-captions is in the following format: \{the number of votes by the workers for this picture, rank of the picture ordered by the workers (based on votes), rank of the picture ordered by experts\}. 
Conclusion

- CrowdDB is a relational query processing system that uses microtask-based crowdsourcing to answer queries that cannot otherwise be answered.
- Combination of human input with high-powered database processing:
  - Extends the range of existing database systems.
  - Enables completely new applications and capabilities
References

REFERENCES
[1] Pictures of the Golden Gate Bridge retrieved from Flickr by akaporn, Dawn Endico, devinleedrew, di_the_huntress, Geoff Livingston, kevincole, Marc_Smith, and superstriker two under the Creative Commons Attribution 2.0 Generic license.
References

References

Questions...?

Thank you.