CrowdDB: Answering Queries with Crowdsourcing

Franklin, Michael J., Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. "CrowdDB: answering queries with crowdsourcing." In *SIGMOD Conference*, pp. 61-72. 2011.

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?

samasource

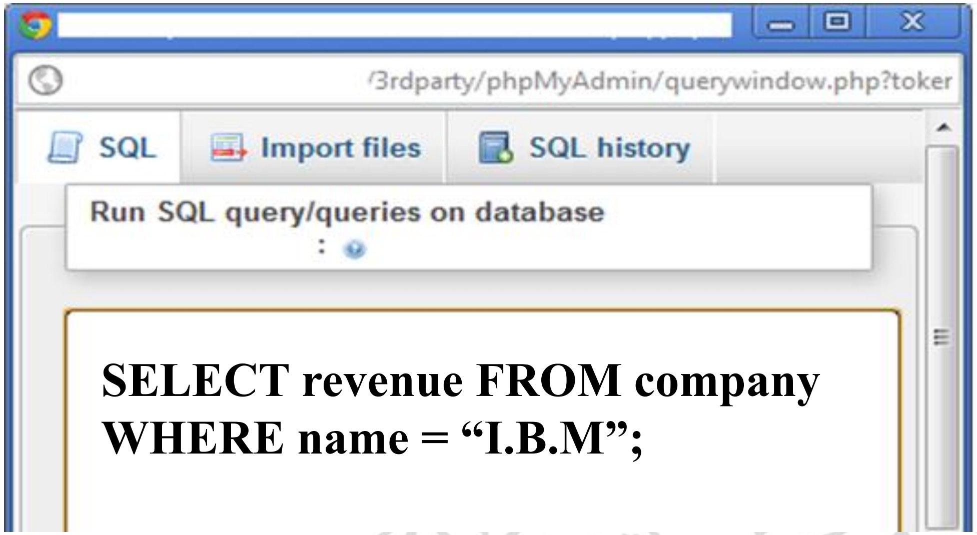
amazon mechanical turk beta

twitter

flickr

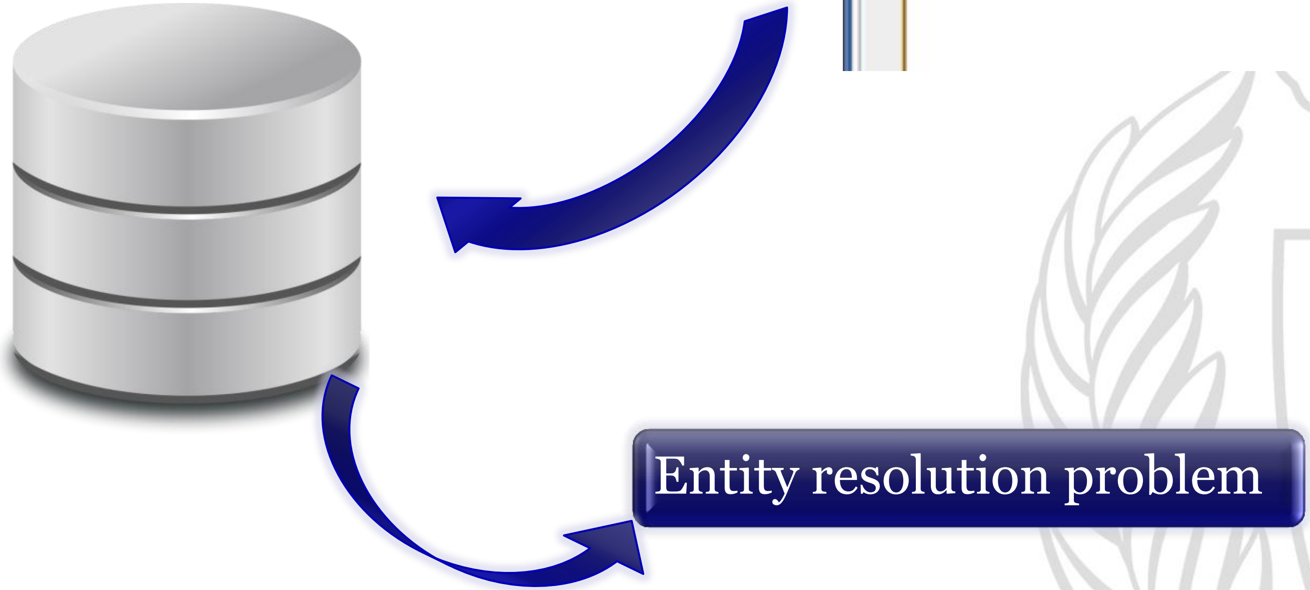
Problem definition

company_name	logo	revenue
Apple		US$ 156.508 billion
Intelligence Bussines Machine	IBM	US$ 106.916 billion
Microsoft	Microsoft	US$ 73.72 billion



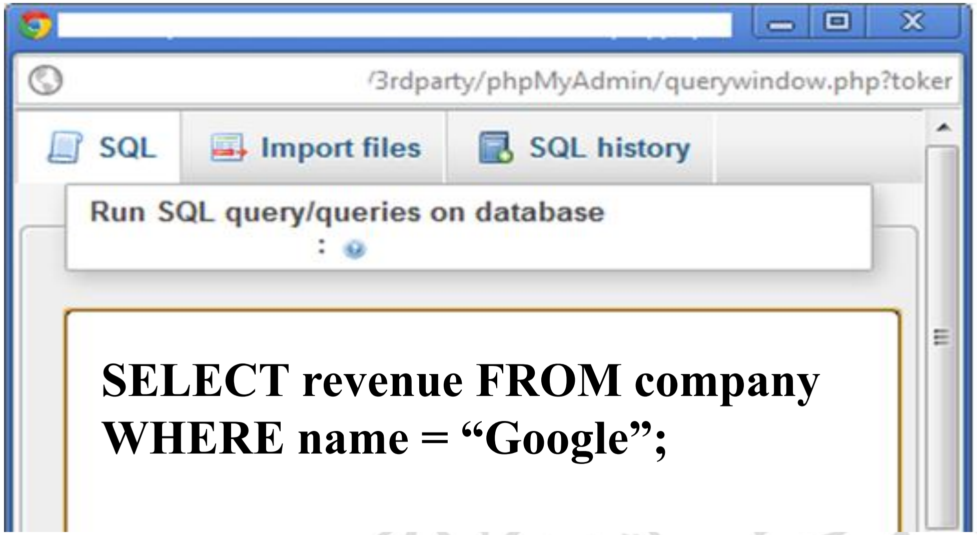
Run SQL query/queries on database :

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



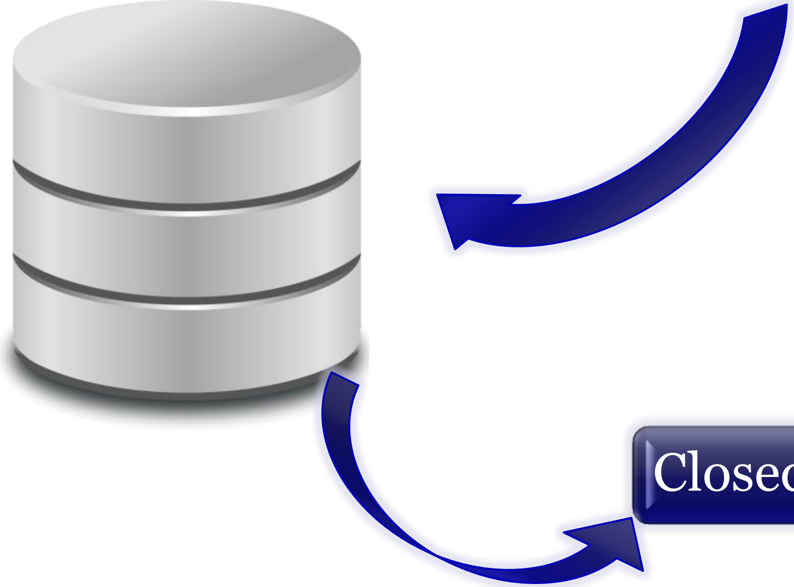
Problem definition

company_name	logo	revenue
Apple		US$ 156.508 billion
Intelligence Bussines Machine	IBM	US$ 106.916 billion
Microsoft	Microsoft	US$ 73.72 billion



Run SQL query/queries on database
:

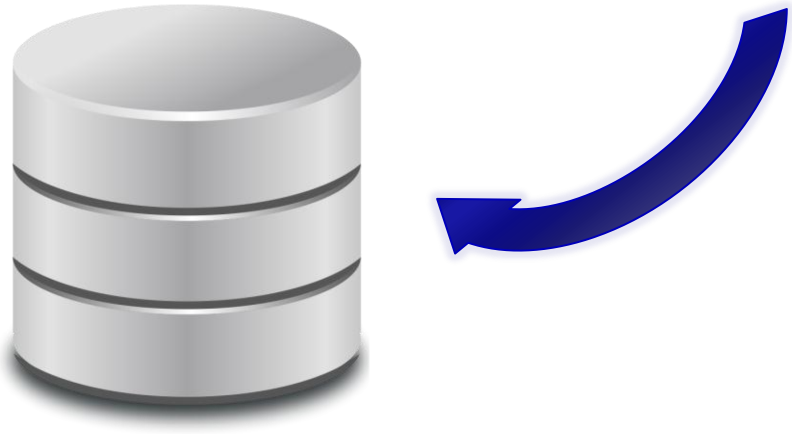
```
SELECT revenue FROM company
WHERE name = “Google”;
```

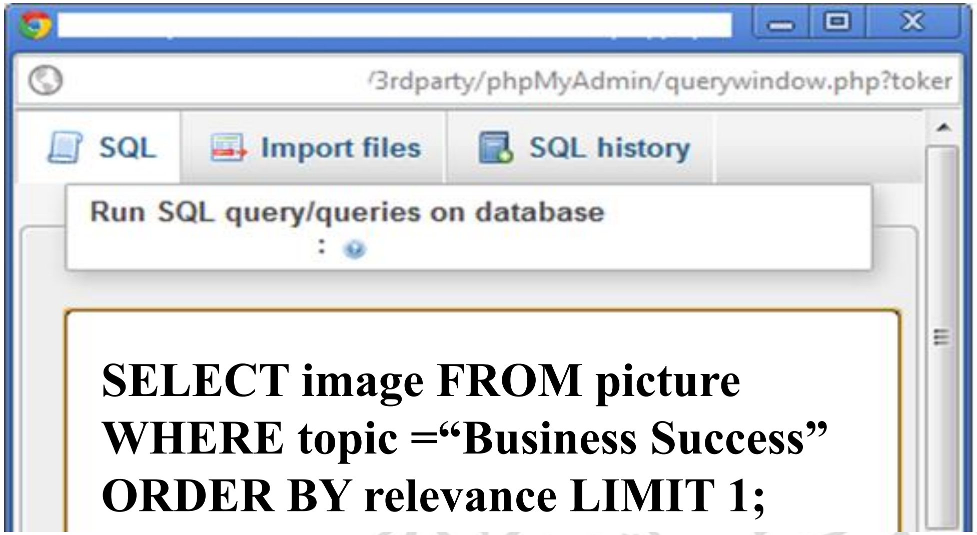


Closed world assumption

Problem definition

image	relevance

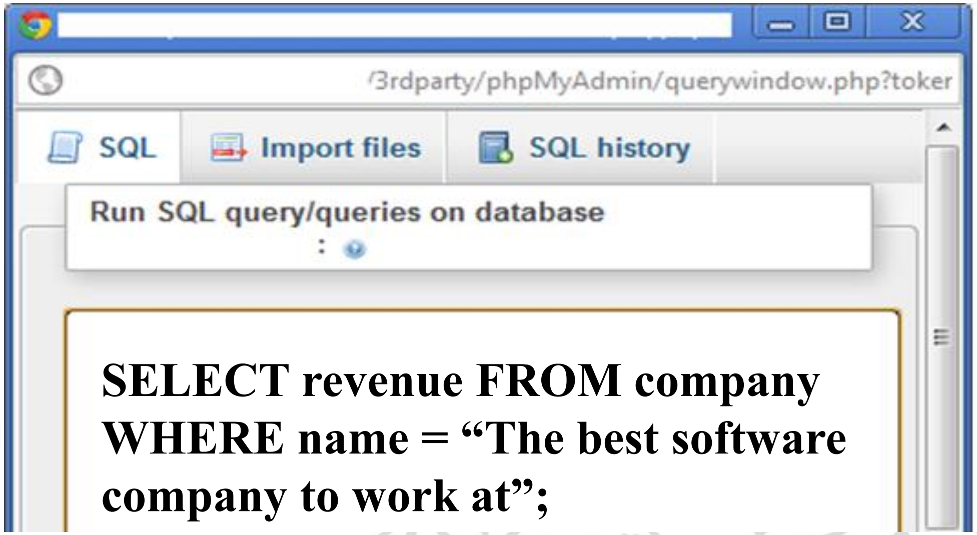



3rdparty/phpMyAdmin/querywindow.php?toker
SQL
Import files
SQL history
Run SQL query/queries on database :

```
SELECT image FROM picture
WHERE topic =“Business Success”
ORDER BY relevance LIMIT 1;
```

Problem definition

company_name	logo	revenue
Apple		US$ 156.508 billion
Intelligence Bussines Machine	IBM	US$ 106.916 billion
Microsoft	Microsoft	US$ 73.72 billion



Run SQL query/queries on database :

```
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?

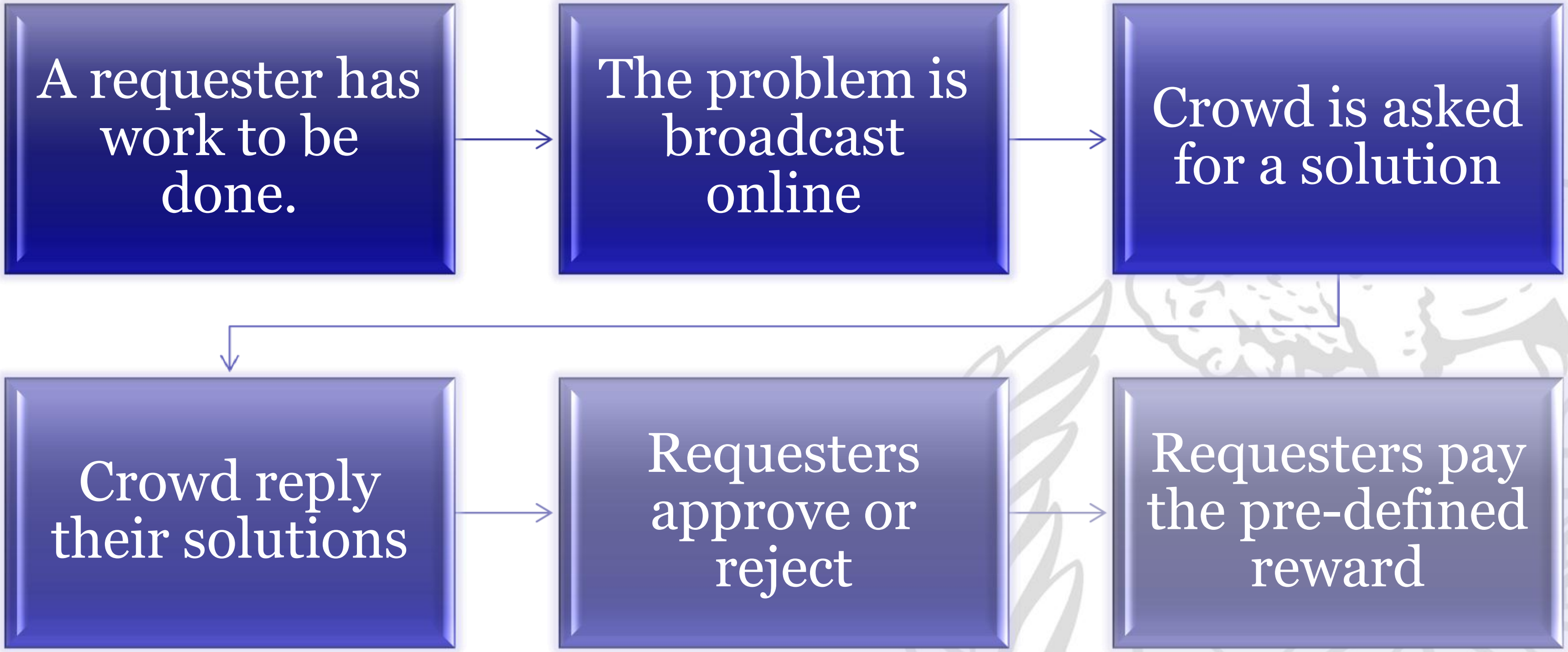
Image from imlworldwide.com

Crowdsourcing

Two main human capabilities that allow corrects 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, tittle, 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)

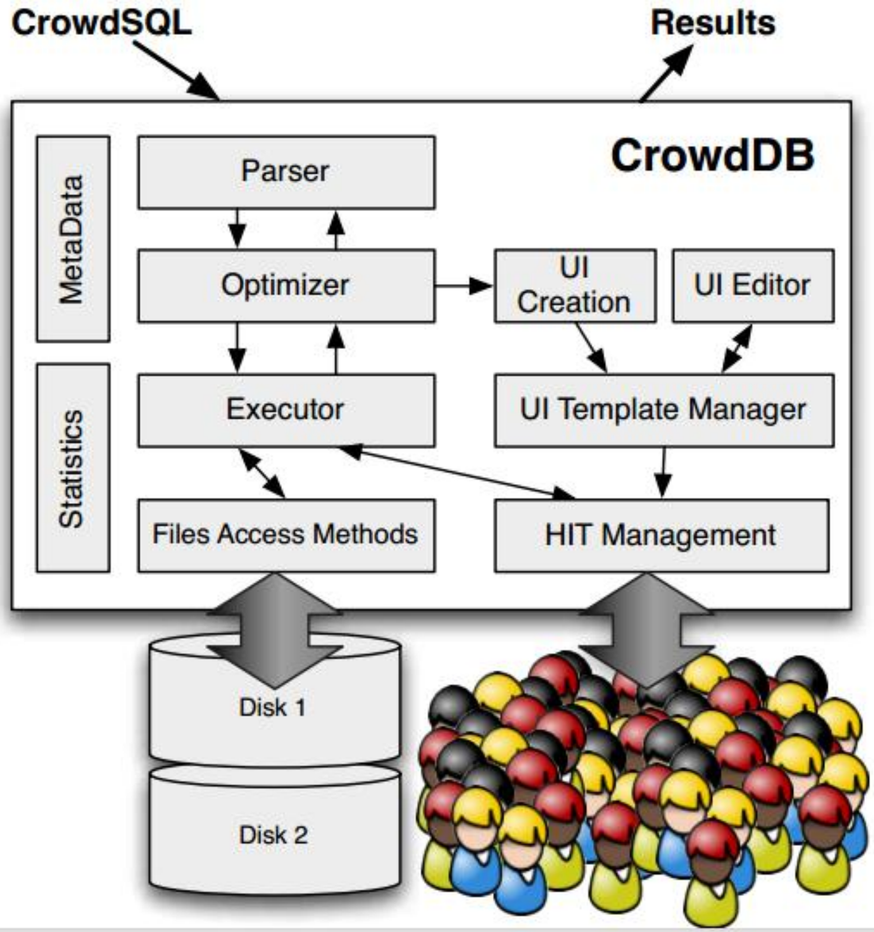
CrowdDB – Design Considerations

- 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

- Suppor 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 built in functions (CROWDEQUAL and CROWDORDER):

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

```
SELECT profile FROM department
WHERE name ~= "CS";
```

CrowdSQL – Subjective comparisons

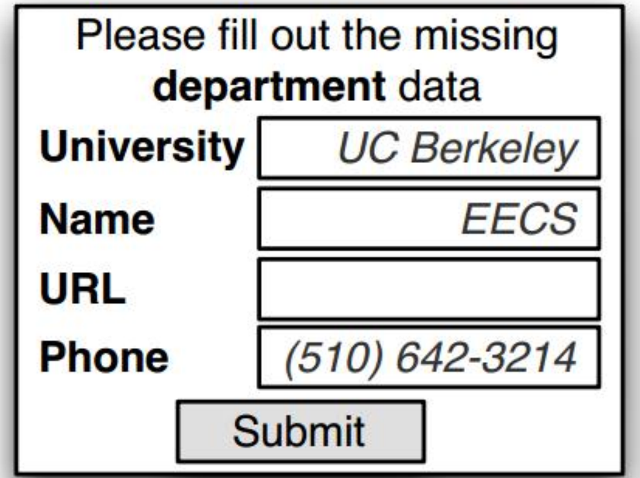
- CROWORDER : Used to ask crowd rank the result.

```
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");
```

User Interface Generation

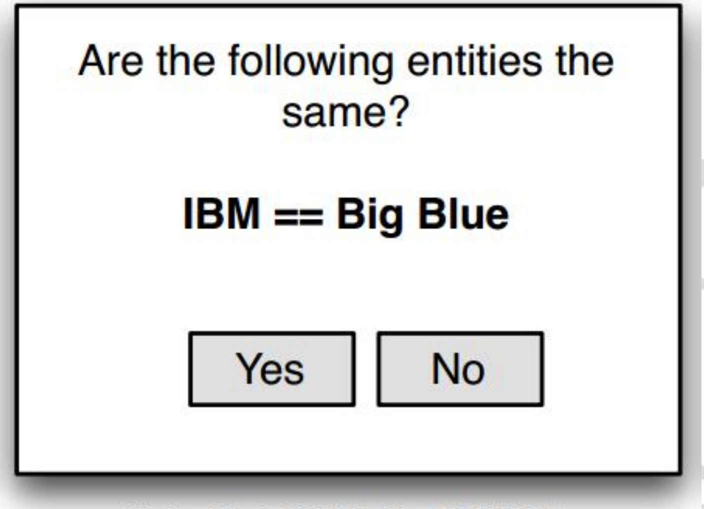
Key: Provide effective user interfaces.



Please fill out the missing **department** data

University	*UC Berkeley*
Name	*EECS*
URL	
Phone	*(510) 642-3214*

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



Are the following entities the same?

IBM == Big Blue

(b) CROWDEQUAL

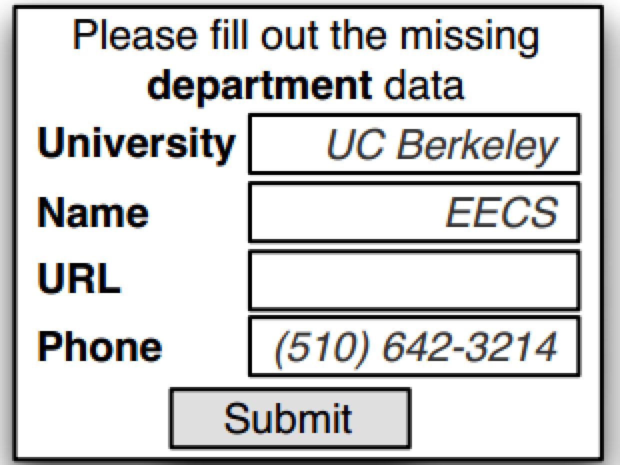
User Interface Generation

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

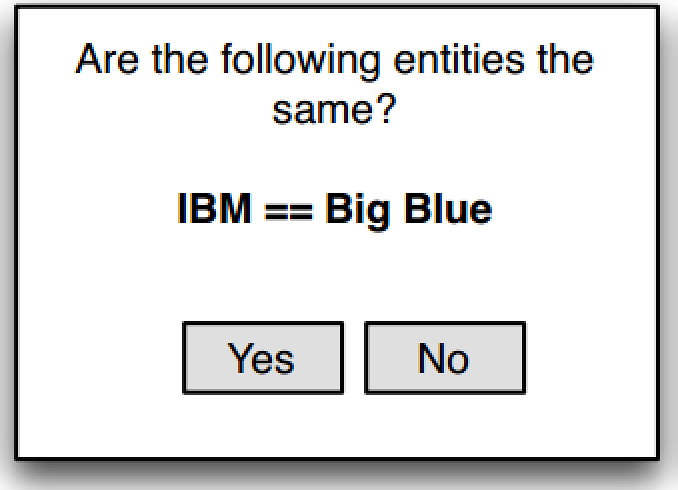
Key: Provide effective user interfaces.



Please fill out the missing **department** data

University	*UC Berkeley*
Name	*EECS*
URL	
Phone	*(510) 642-3214*

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



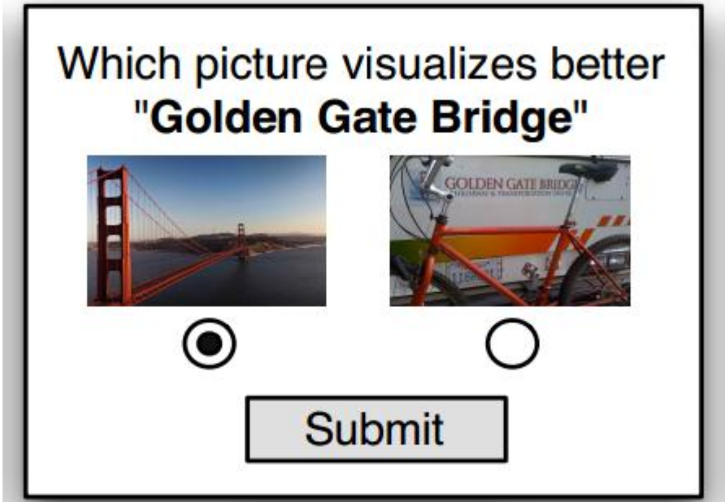
Are the following entities the same?

IBM == Big Blue

(b) CROWDEQUAL

User Interface Generation

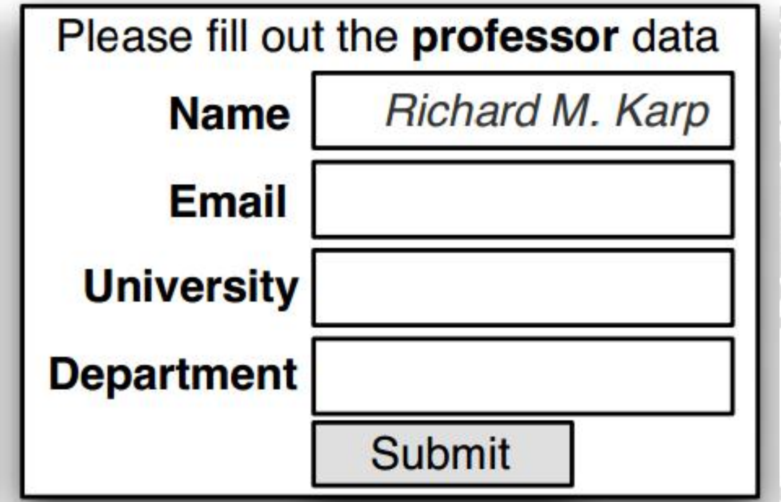
Key: Provide effective user interfaces.



Which picture visualizes better **"Golden Gate Bridge"**

● ○

(c) CROWDORDER



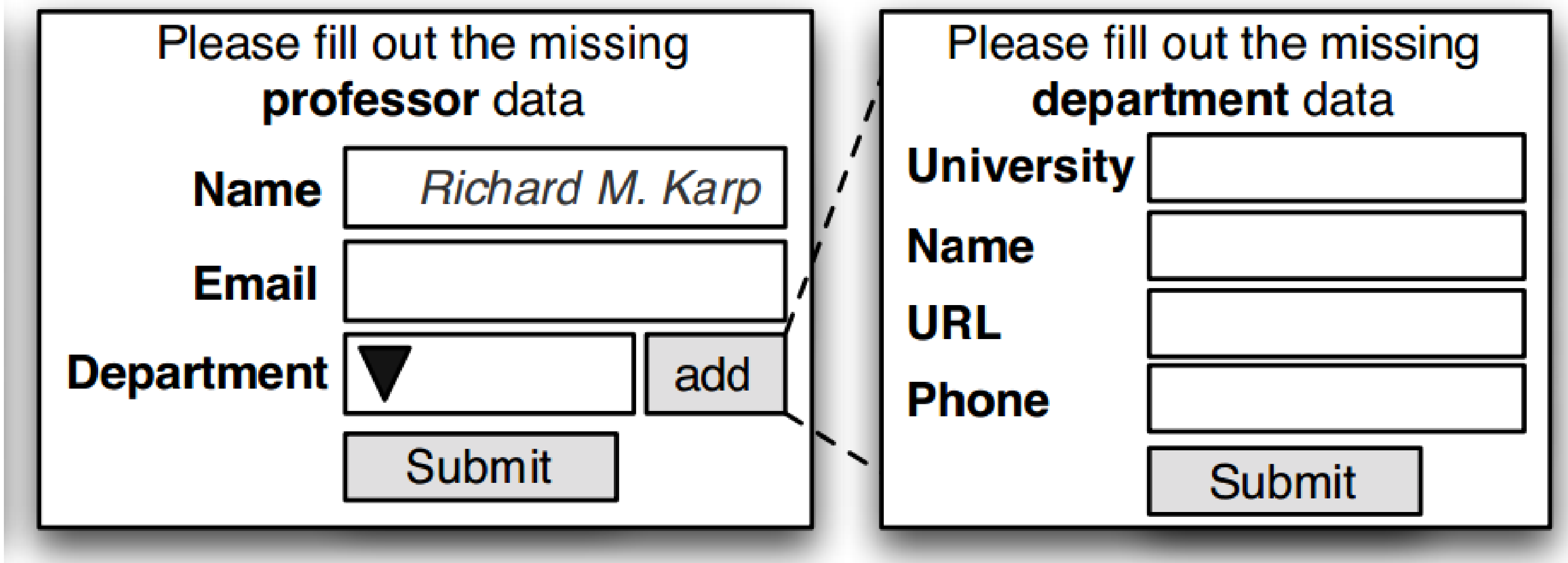
Please fill out the **professor** data

Name	*Richard M. Karp*
Email	
University	
Department	

(d) Foreign Key(normalized)

User Interface Generation

Key: Provide effective user interfaces.


Please fill out the missing **professor** data
Name
Richard M. Karp
Email
Department
add
Submit
Please fill out the missing **department** data
University
Name
URL
Phone
Submit

(e) Foreign Key (denormalized)

Query Processing – Crowd Operators

Current version of CrowdDB has three crowd operators:

- CrowdProbe:

  Crowd missing information about CROWD columns and new tuples. (Uses generated UI)

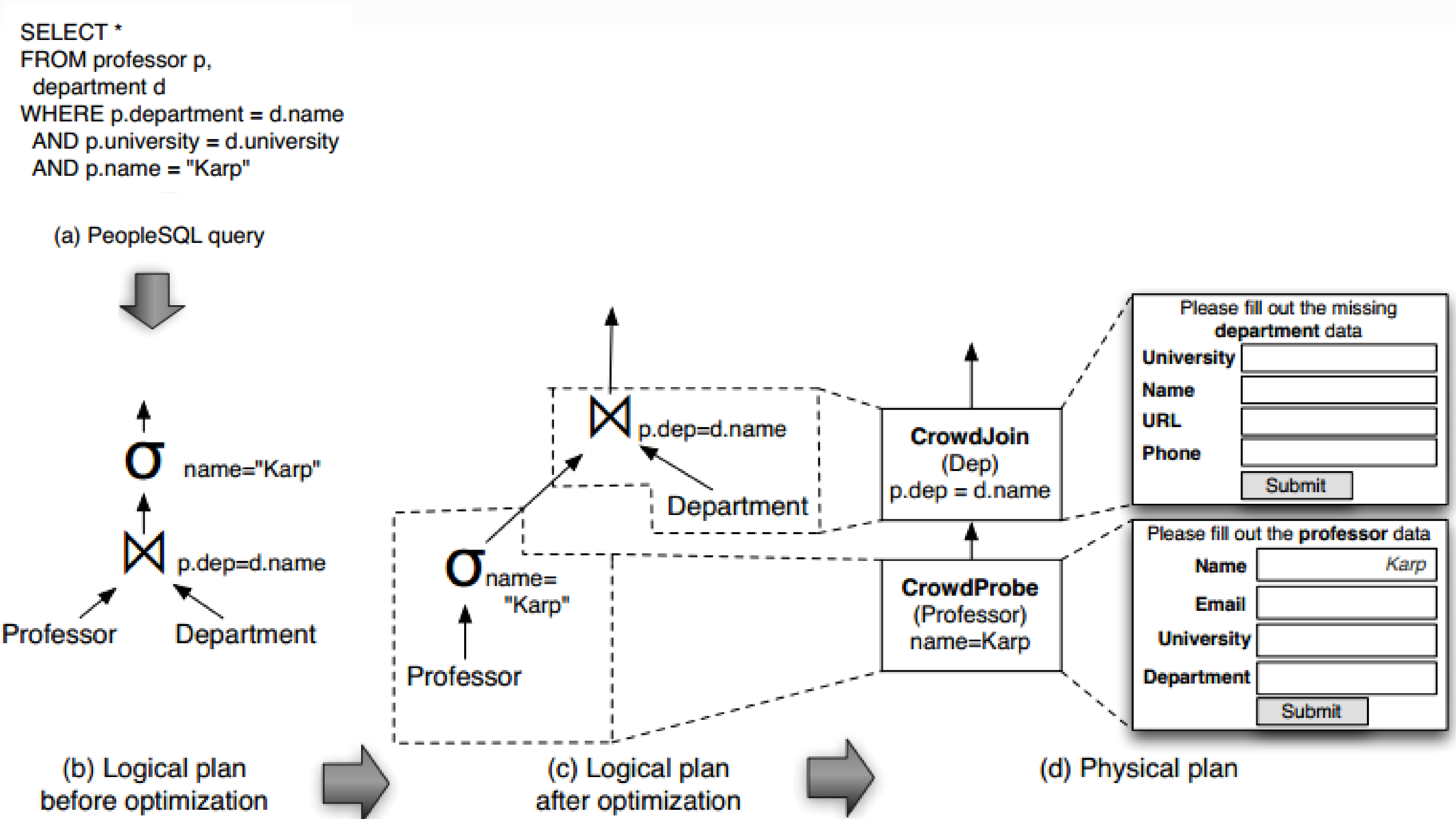
Query Processing – Crowd Operators

- 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



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

(a) PeopleSQL query

Please fill out the missing **department** data

University
Name
URL
Phone
Submit

Please fill out the **professor** data

Name *Karp*
Email
University
Department
Submit

(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:

```
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

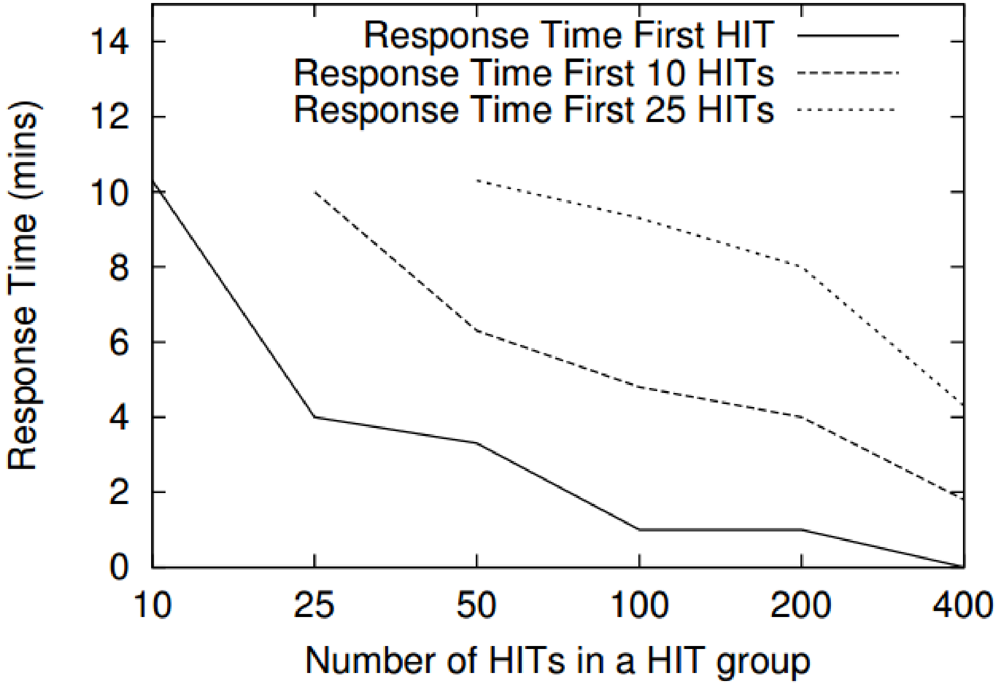


Figure 4: Response Time (min): Vary Hit Group *(1 Asgn/HIT, 1 cent Reward)*

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

Response time decrease dramatically as size of group is increased.

Experiment 2: Responsiveness, Vary Reward

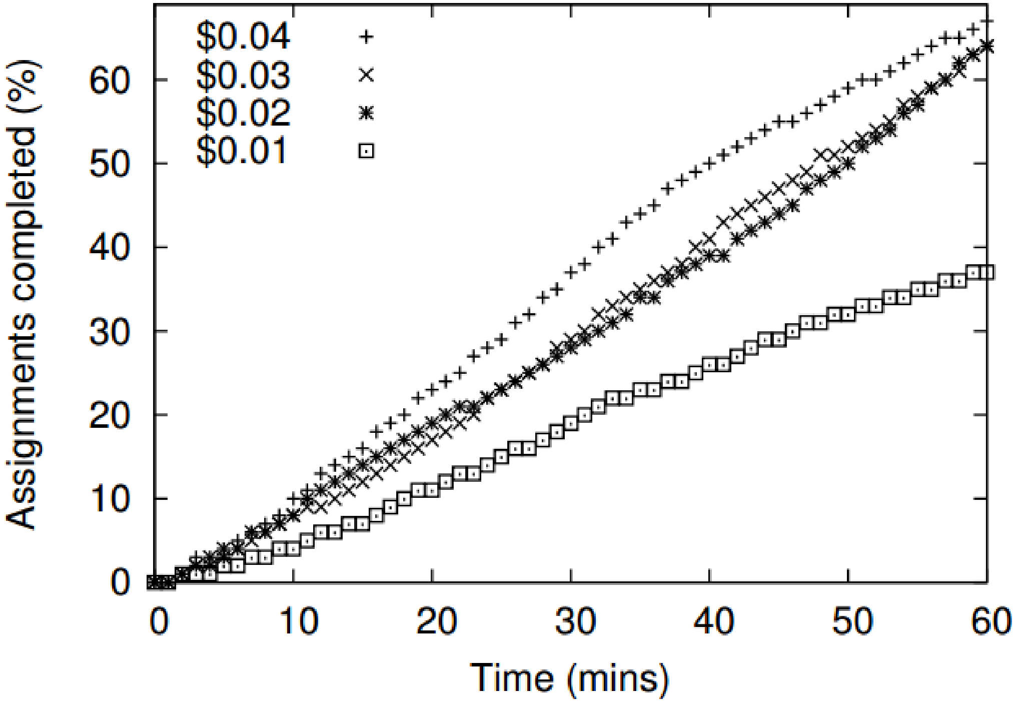


Figure 6: Completion (%): Vary Reward *(100 HITs/Group, 5 Asgn/HIT)*

How response time varies as a function of the reward.

Experiment 2: Responsiveness, Vary Reward

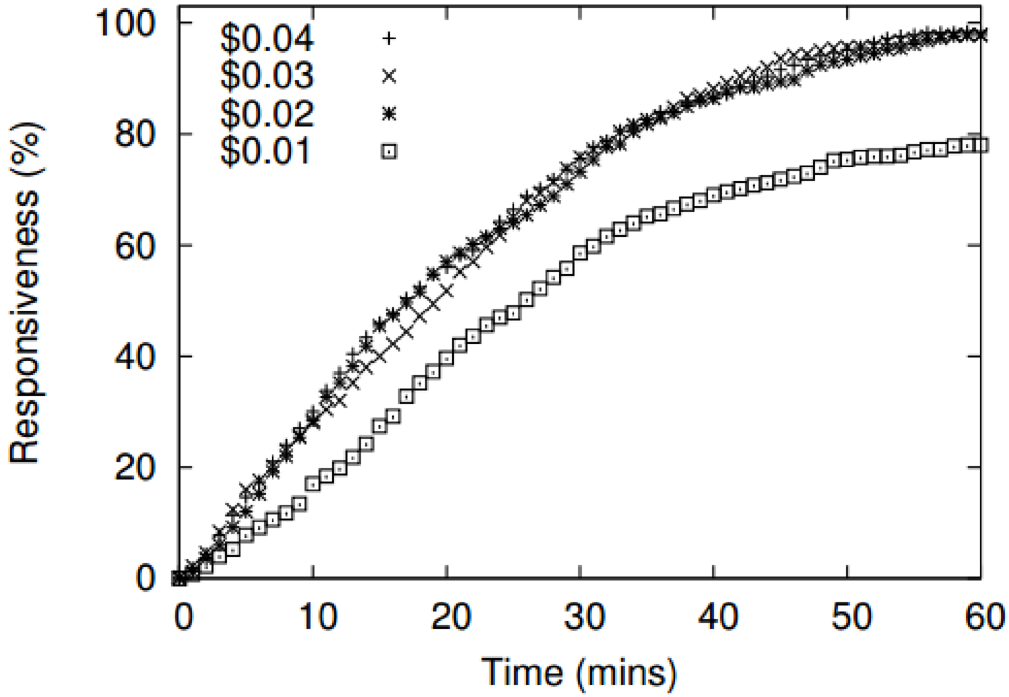


Figure 7: Completion (%): Vary Reward *(100 HITs/Group, 5 Asgn/HIT)*

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

Experiment 3: Worker Affinity and Quality

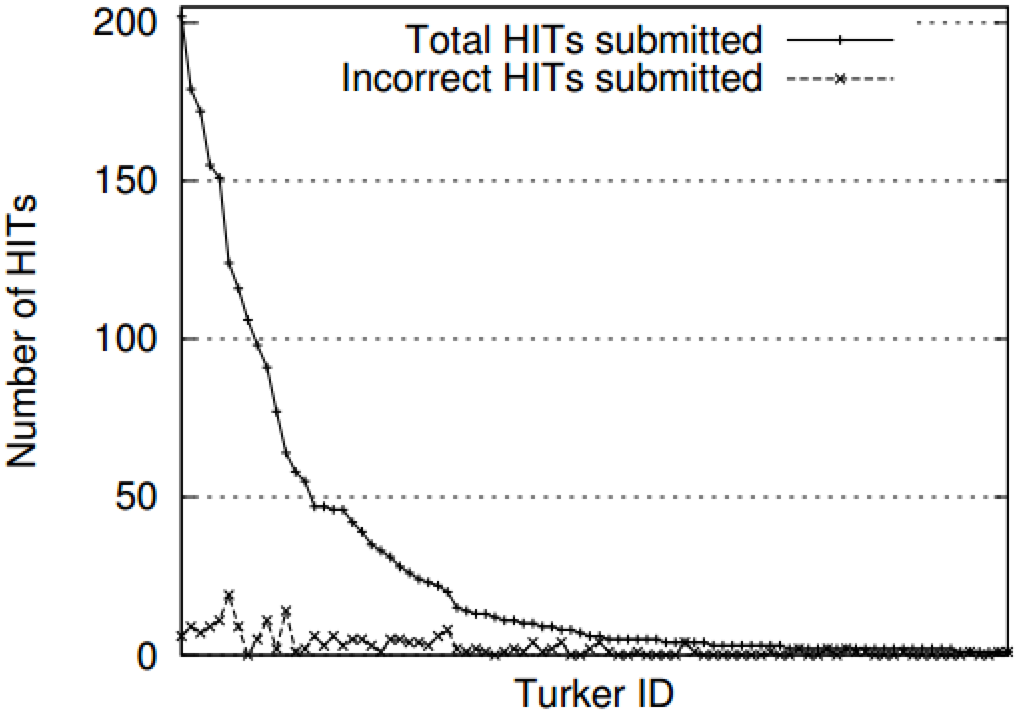
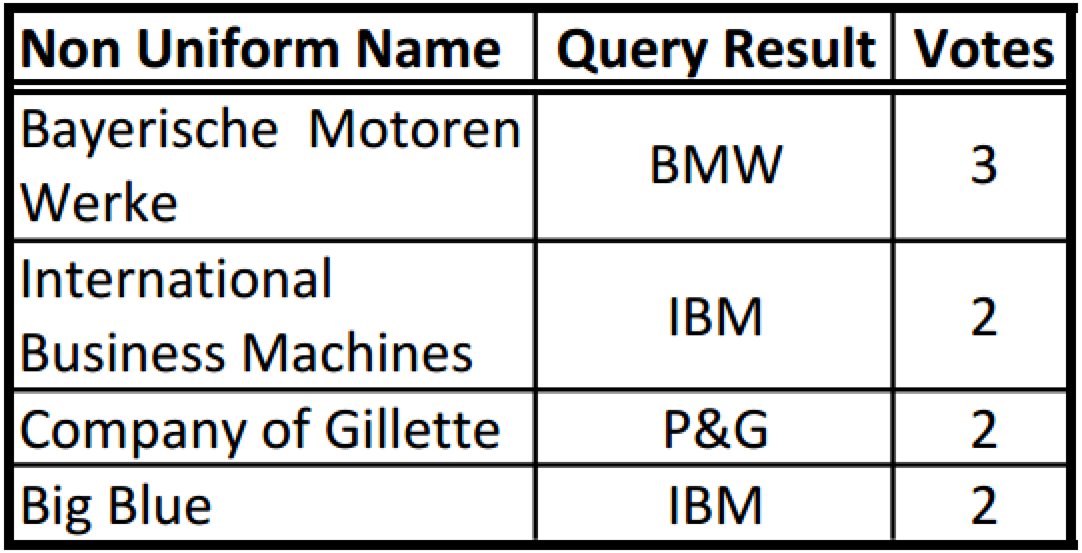


Figure 8: HITs/Quality by Worker *(Any HITs/Group, 5 Asgn/HIT, Any Reward)*

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

Complex Queries: Entity Resolution on Companies

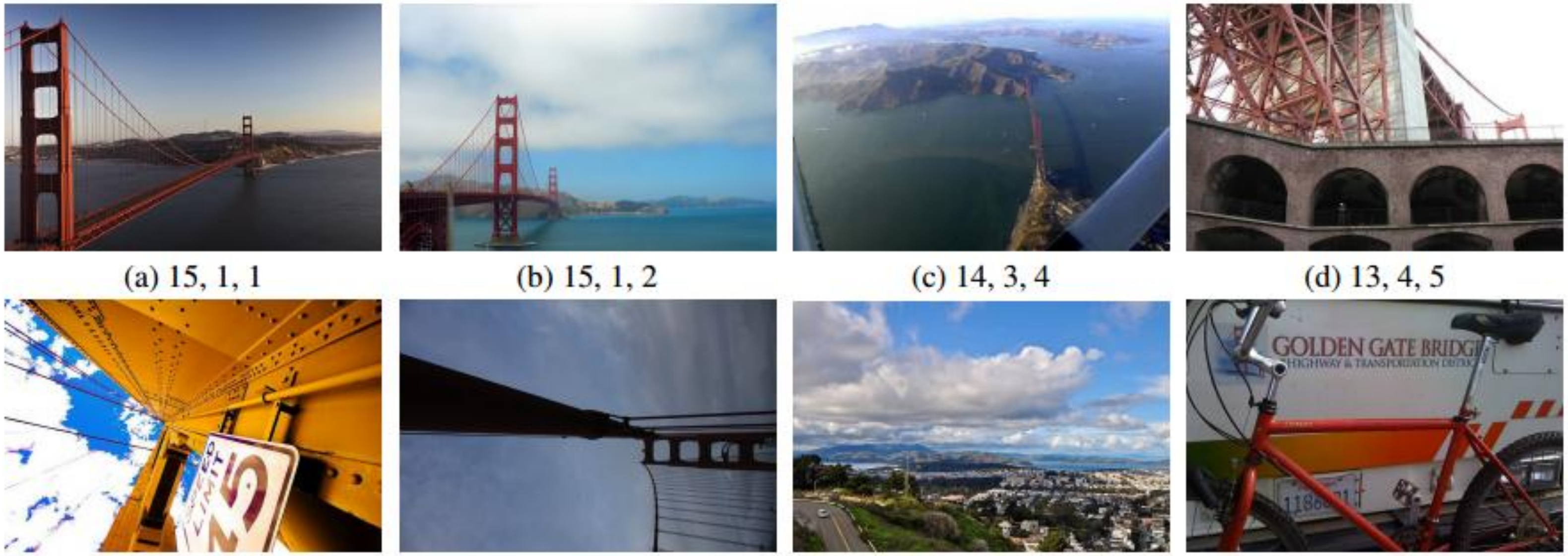


Non Uniform Name	Query Result	Votes
Bayerische Motoren Werke	BMW	3
International Business Machines	IBM	2
Company of Gillette	P&G	2
Big Blue	IBM	2

Figure 9: Entity Resolution on Company Names

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

Complex Queries: Ordering Pictures



(a) 15, 1, 1 (b) 15, 1, 2 (c) 14, 3, 4 (d) 13, 4, 5

(e) 10, 5, 6 (f) 9, 6, 3 (g) 4, 7, 7 (h) 4, 7, 8

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 superstrikertwo under the Creative Commons Attribution 2.0 Generic license.

[2] Amazon. AWS Case Study: Smartsheet, 2006.

[3] Amazon Mechanical Turk. http://www.mturk.com, 2010.

[4] S. Amer-Yahia et al. Crowds, Clouds, and Algorithms: Exploring the Human Side of "Big Data" Applications. In SIGMOD, 2010.

[5] M. Armbrust et al. PIQL: A Performance Insightful Query Language. In SIGMOD, 2010.

[6] M. S. Bernstein et al. Soylent: A Word Processor with a Crowd Inside. In ACM SUIST, 2010.

[7] M. J. Carey and D. Kossmann. On saying “Enough already!” in SQL. SIGMOD Rec., 26(2):219–230, 1997.

[8] S. S. Chawathe et al. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In IPSJ, 1994.

References

[9] K. Chen et al. USHER: Improving Data Quality with Dynamic Forms. In ICDE, pages 321–332, 2010.

[10] A. Doan, R. Ramakrishnan, and A. Halevy. Crowdsourcing Systems on the World-Wide Web. CACM, 54:86–96, Apr. 2011.

[11] L. M. Haas et al. Optimizing Queries Across Diverse Data Sources. In VLDB, 1997.

[12] J. M. Hellerstein et al. Adaptive Query Processing: Technology in Evolution. IEEE Data Eng. Bull., 2000.

[13] J. M. Hellerstein and J. F. Naughton. Query Execution Techniques for Caching Expensive Methods. In SIGMOD, pages 423–434, 1996.

[14] E. Huang et al. Toward Automatic Task Design: A Progress Report. In HCOMP, 2010.

[15] P. G. Ipeirotis. Analyzing the Amazon Mechanical Turk Marketplace. http://hdl.handle.net/2451/29801, 2010.

[16] P. G. Ipeirotis. Mechanical Turk, Low Wages, and the Market for Lemons. http://behind-the-enemy-lines.blogspot.com/2010/07/mechanical-turk-low-wages-and-market.html, 2010.

[17] A. G. Kleppe, J. Warmer, and W. Bast. MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley, 2003.

References

[18] G. Little. How many turkers are there?
http://groups.csail.mit.edu/uid/deneme/?p=502, 2009.
[19] G. Little et al. TurKit: Tools for Iterative Tasks on Mechanical Turk.
In HCOMP, 2009.
[20] A. Marcus et al. Crowdsourced Databases: Query Processing with
People. In CIDR, 2011.
[21] Microsoft. Table Column Properties (SQL Server), 2008.
[22] A. Parameswaran et al. Human-Assisted Graph Search: It's Okay to
Ask Questions. In VLDB, 2011.
[23] A. Parameswaran and N. Polyzotis. Answering Queries using
Humans, Algorithms and Databases. In CIDR, 2011.
[24] J. Ross et al. Who are the Crowdworkers? Shifting Demographics in
Mechanical Turk. In CHI EA, 2010.
[25] D. Schall, S. Dustdar, and M. B. Blake. Programming Human and
Software-Based Web Services. Computer, 43(7):82–85, 2010.
[26] Turker Nation. http://www.turkernation.com/, 2010.
[27] Turkopticon. http://turkopticon.differenceengines.com/, 2010.
[28] T. Yan, V. Kumar, and D. Ganesan. CrowdSearch: Exploiting Crowds
for Accurate Real-time. Image Search on Mobile Phones. In
MobiSys, 2010.

Questions…?

Thank you.