The Data Interaction Game

Ben McCamish, Vahid Ghadakchi, Arash Termehchy, Behrouz Touri, Liang Huang

Information & **D**ata Manag**e**ment and **A**nalytics Laboratory (IDEA)

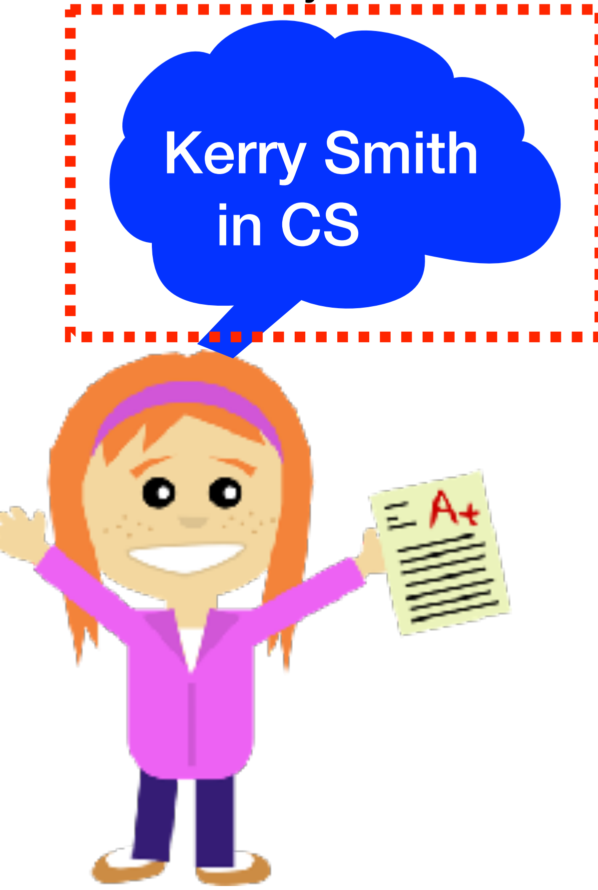
The User and the Database

Grades			
First_Name	Last_Name	Dept.	Grade
...	...	...	...
Sarah	Smith	CE	A
John	Smith	EE	B
Kerry	Smith	CS	D
...	...	...	...

- Users wish to find information from the database.

The intent is what the user is looking for in the database

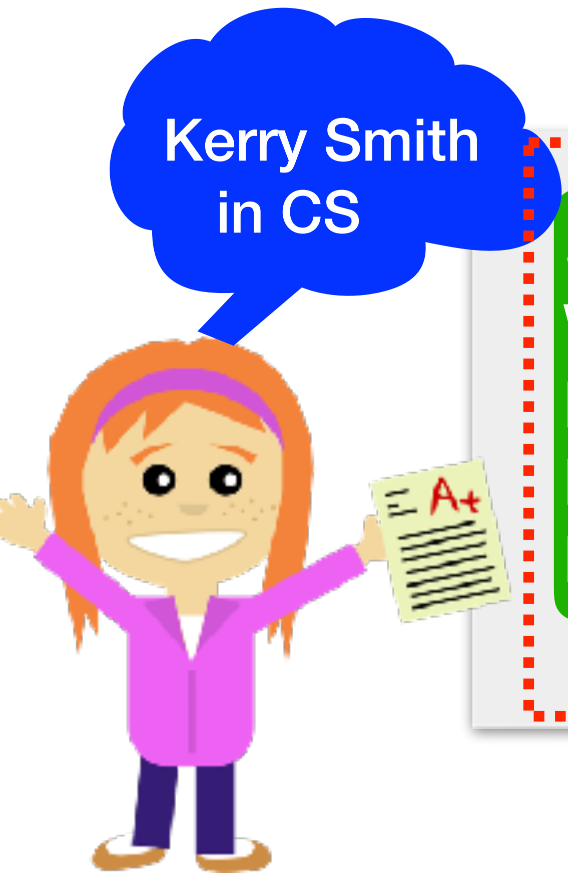
Intents they wish to find



Grades			
First_Name	Last_Name	Dept.	Grade
…	…	…	…
Sarah	Smith	CE	A
John	Smith	EE	B
Kerry	Smith	CS	D
…	…	…	…

- The user wishes to find Kerry Smith from the CS department in the database

Intents are expressed using queries



Use Queries to express intents

SELECT *
WHERE
First_Name='Kerry' and
Last_Name='Smith' and
Dept.='CS'

Search

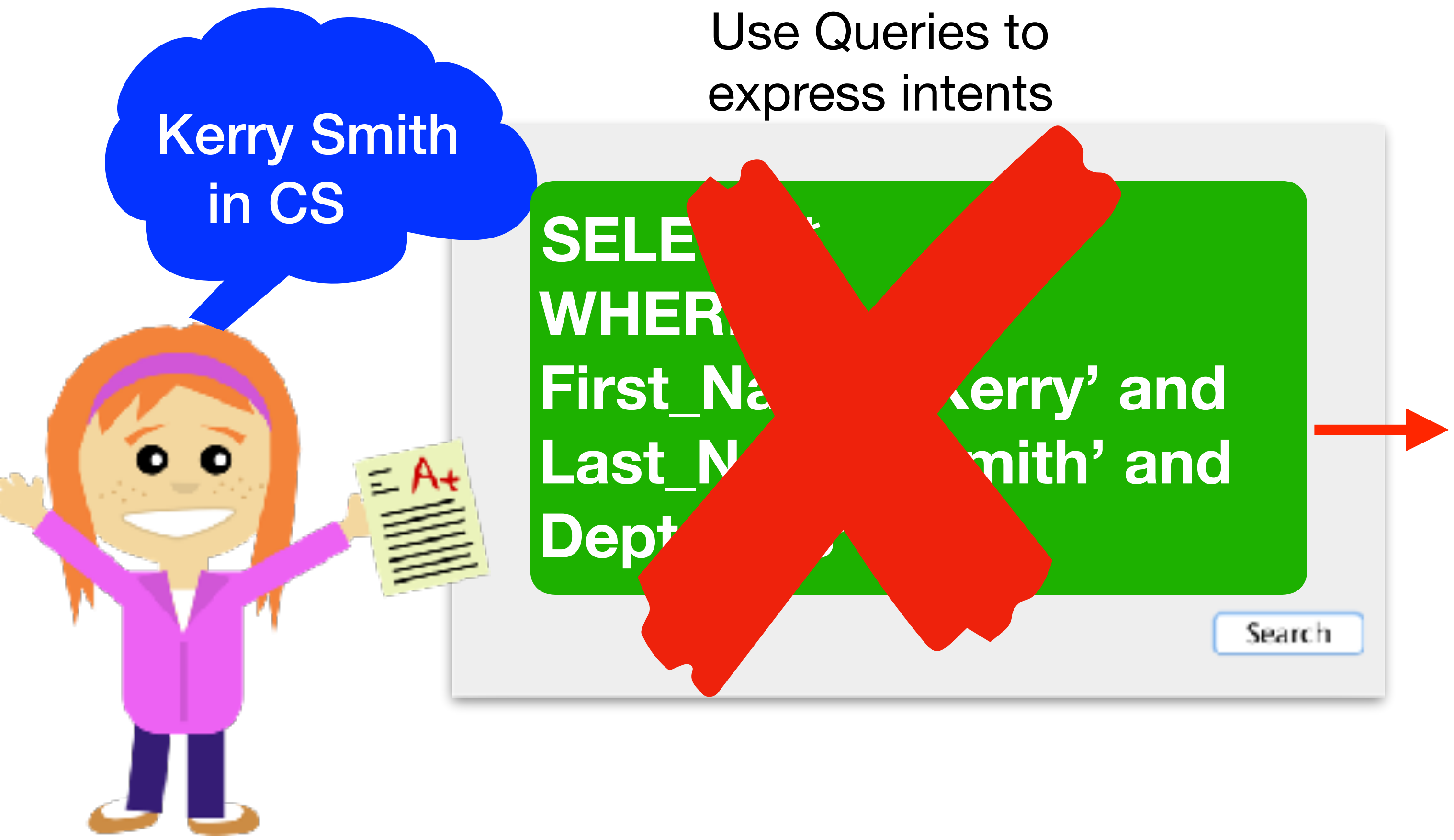
Grades			
First_Name	Last_Name	Dept.	Grade
...	...	...	...
Sarah	Smith	CE	A
John	Smith	EE	B
Kerry	Smith	CS	D
...	...	...	...

- The user expresses their intent with a SQL query

Most users do not know structure and content of database or SQL

Intents they wish to find

Use Queries to express intents

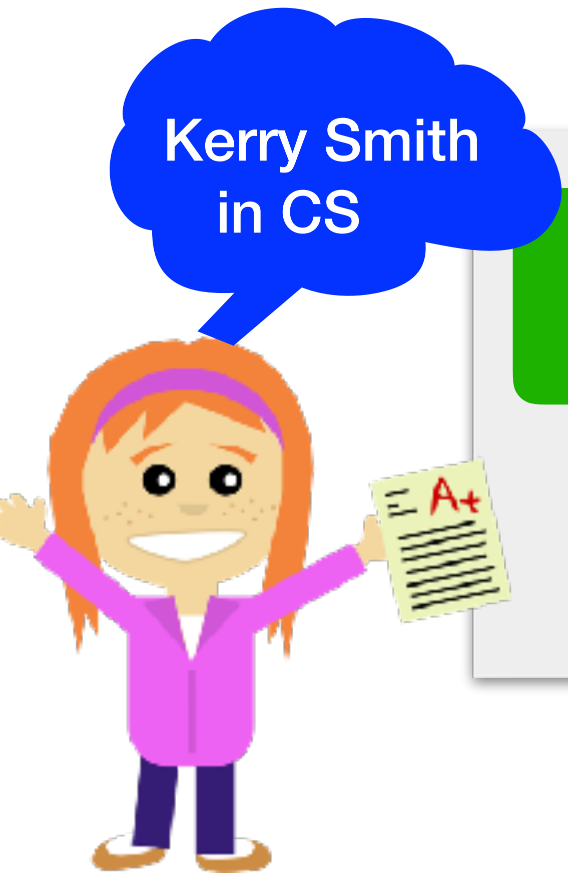


Grades			
First_Name	Last_Name	Dept.	Grade
...	...	...	...
Sarah	Smith	CE	A
John	Smith	EE	B
Kerry	Smith	CS	D
...	...	...	...

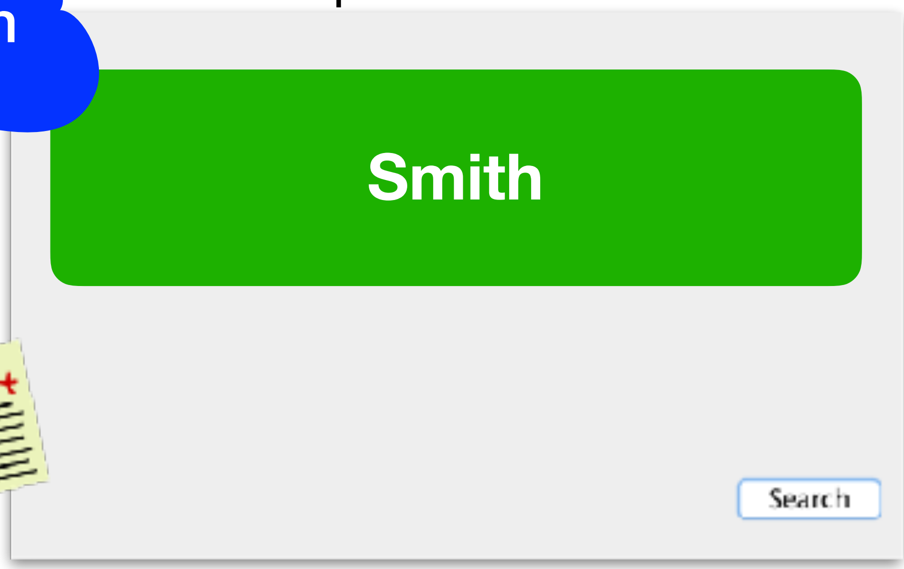
- Normal users such as scientists prefer to use keyword queries

Users prefer to use keyword queries as they are easier to use

Intents they wish to find



Use Queries to express intents



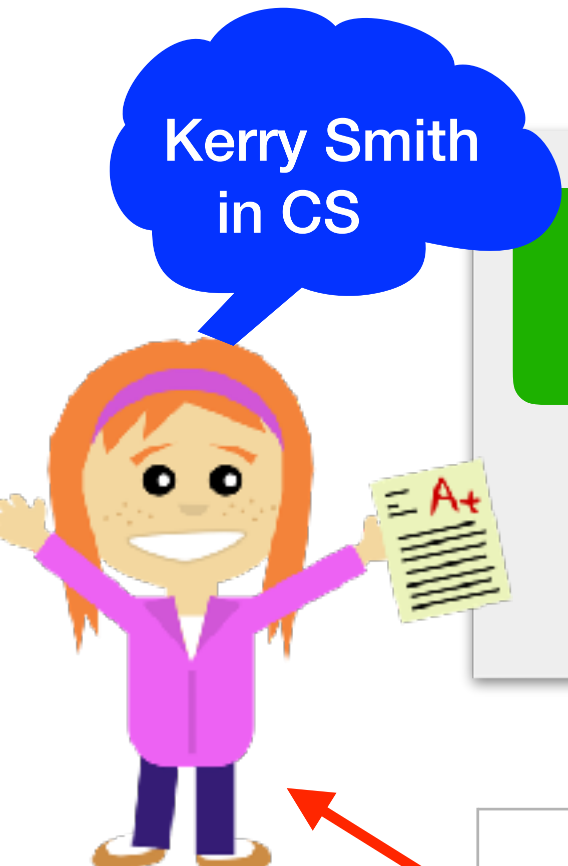
Grades

First_Name	Last_Name	Dept.	Grade
...	...	...	...
Sarah	Smith	CE	A
John	Smith	EE	B
Kerry	Smith	CS	D
...	...	...	...

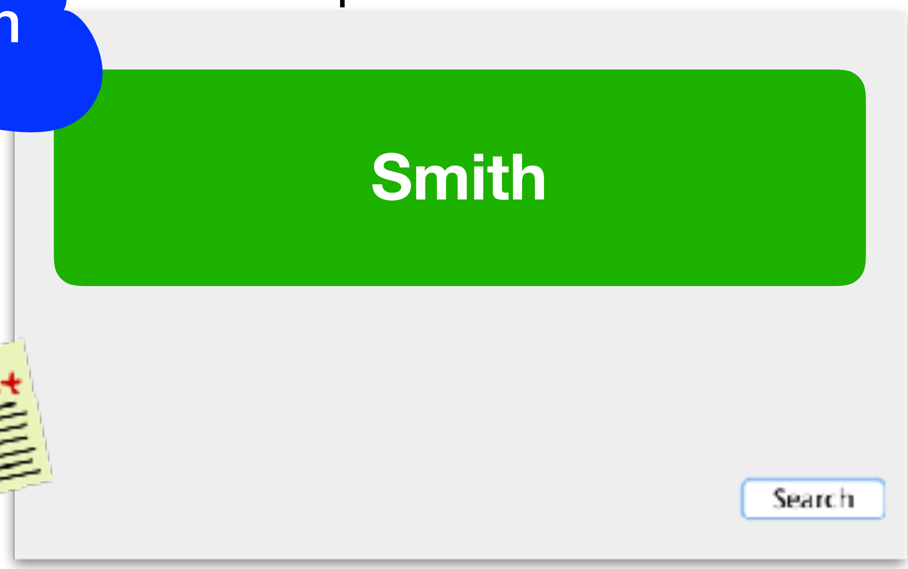
- Don’t need to know the structure or content of the database
- No need to know SQL or other structured query language

Database struggles with keyword queries

Intents they wish to find



Use Queries to express intents



Grades

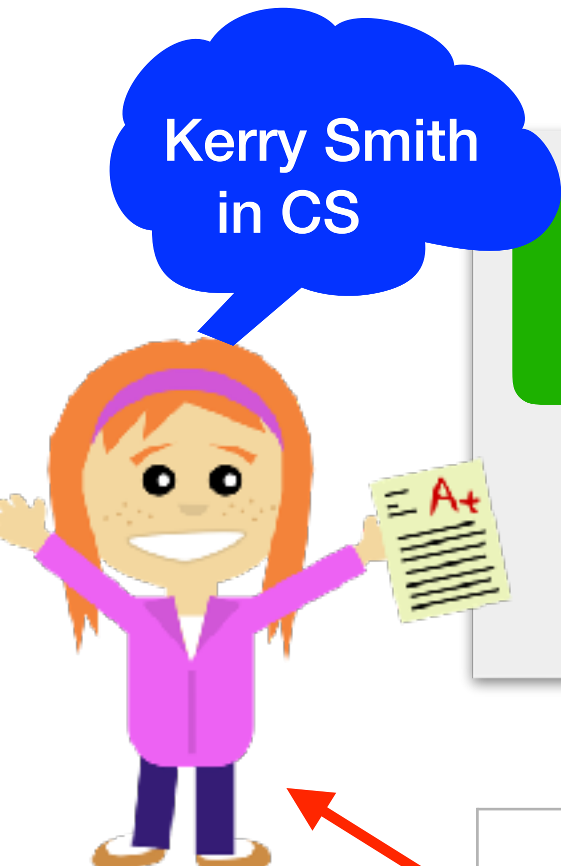
First_Name	Last_Name	Dept.	Grade
...	...	...	...
Sarah	Smith	CE	A
John	Smith	EE	B
Kerry	Smith	CS	D
...	...	...	...

Results

First_Name	Last_Name	Dept.	Grade
Sarah	Smith	CE	A
John	Smith	EE	B

- Since keyword queries are imprecise, database system struggles to satisfy the user

Users learn by interacting with database systems

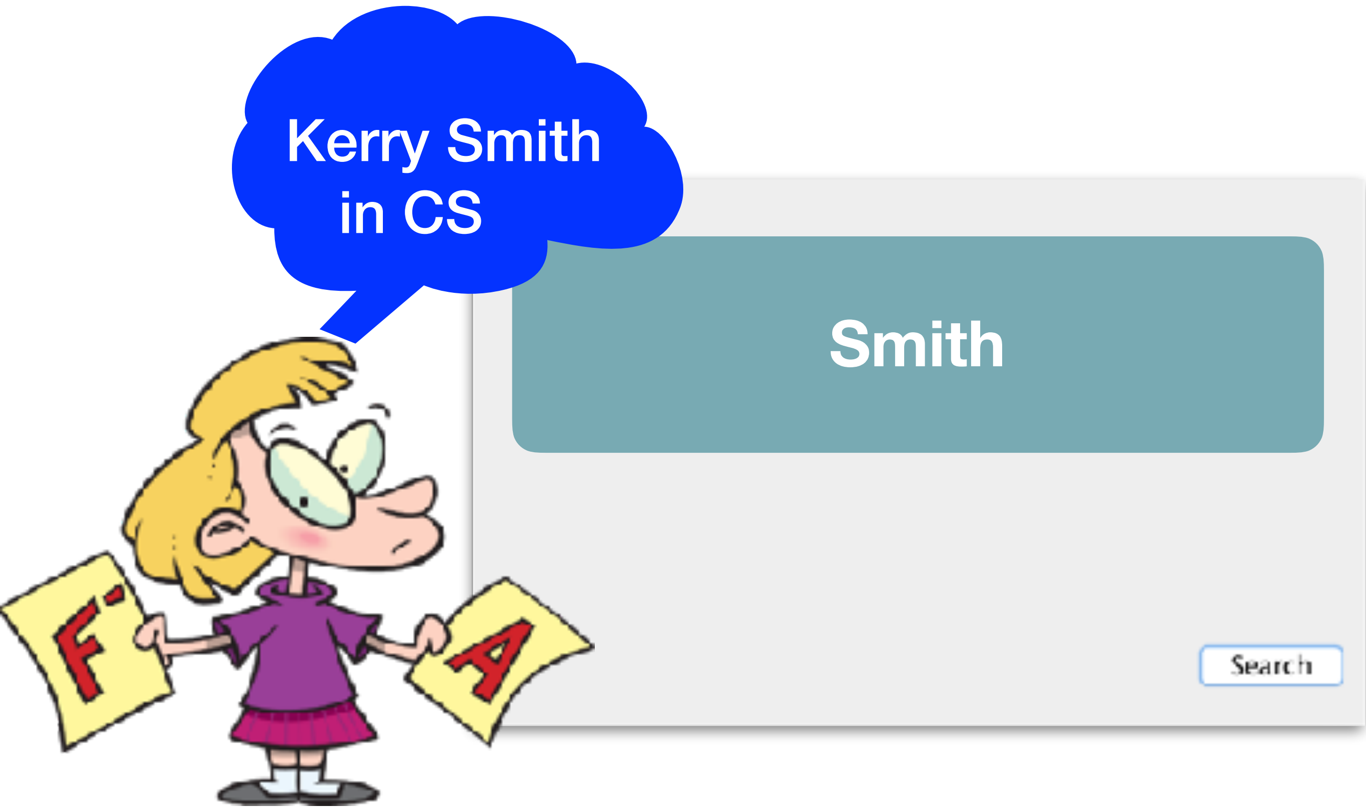


Grades			
First_Name	Last_Name	Dept.	Grade
...	...	...	...
Sarah	Smith	CE	A
John	Smith	EE	B
Kerry	Smith	CS	D
...	...	...	...

Results			
First_Name	Last_Name	Dept.	Grade
Kerry	Smith	CS	D

- Learning and reformulating query allowed the user to find the desired student

Database system can also learn from interactions



Grades

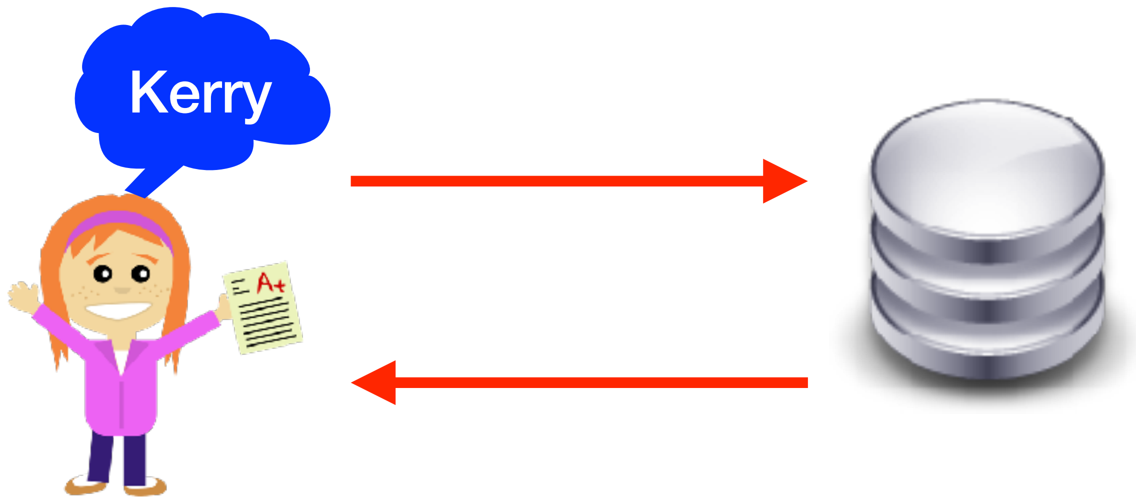
First_Name	Last_Name	Dept.	Grade
...	...	...	...
Sarah	Smith	CE	A
John	Smith	EE	B
Kerry	Smith	CS	D
...	...	...	...

Results

First_Name	Last_Name	Dept.	Grade
Kerry	Smith	CS	D

- User gives feedback to database through clicks
- Database system has learned to return *Kerry Smith* in CS department

Naturally data interaction is a game between two rational agents



- Two Players: **user** and **database system**
- Agents learn and adapt
- Final goal: user to get desired information
    - Database system understands the intent behind users queries
    - User expresses intent in a way that DBMS understands
- **User Strategy**: How intents are expressed using queries
- **DBMS Strategy**: How to map imprecise queries to desired queries
- **Payoff**: The amount of desired information the user receives.

User thinks of what they want to find in DBMS

Intent #	Intent
e_1	*John Smith in EE*
e_2	*Sarah Smith in CE*
e_3	*Kerry Smith in CS*

Query #	Query
q_1	? *"Smith CE"*
q_2	*"Smith"*

- The intent can be multiple tuples
- They need to decide how to express their intent to DBMS

Grades

First_Name	Last_Name	Dept.	Grade
...	...	...	...
Sarah	Smith	CE	A
John	Smith	EE	B
Kerry	Smith	CS	D
...	...	...	...

User strategy is mapping of intents to queries

Intent #	Intent
e_1	*John Smith in EE*
e_2	*Sarah Smith in CE*
e_3	*Kerry Smith in CS*

Query #	Query
q_1	*"Smith CE"*
q_2	*"Smith"*

- Use keyword queries
- Row-stochastic mapping from intents to queries.

User Strategy

	q_1	q_2
e_1	0	1
e_2	0.5	0.5
e_3	0	1

Grades

First_Name	Last_Name	Dept.	Grade
...	...	...	...
Sarah	Smith	CE	A
John	Smith	EE	B
Kerry	Smith	CS	D
...	...	...	...

User may use a single query for multiple intents

Intent #	Intent
e_1	*John Smith in EE*
e_2	*Sarah Smith in CE*
e_3	*Kerry Smith in CS*

Query #	Query
q_1	*"Smith CE"*
q_2	*"Smith"*

- Due to the lack of knowledge, saving time, …

- Makes it hard to interpret the exact intent behind the query.

User Strategy

	q_1	q_2
e_1	0	1
e_2	0.5	0.5
e_3	0	1

Grades			
First_Name	Last_Name	Dept.	Grade
…	…	…	…
Sarah	Smith	CE	A
John	Smith	EE	B
Kerry	Smith	CS	D
…	…	…	…

DBMS receives query and needs to decide what user wants

Query #	Query
q_1	"Smith CE"
q_2	"Smith"

Intent #	Intent
e_1	$ans(y) \leftarrow Grades(x, \text{'Smith'}, \text{'EE'}, y)$
e_2	$ans(y) \leftarrow Grades(x, \text{'Smith'}, \text{'CE'}, y)$
e_3	$ans(y) \leftarrow Grades(x, \text{'Smith'}, \text{'CS'}, y)$

?

Grades

First_Name	Last_Name	Dept.	Grade
…	…	…	…
Sarah	Smith	CE	A
John	Smith	EE	B
Kerry	Smith	CS	D
…	…	…	…

- How should it map the received keyword queries to the user's actual information needs?

DBMS receives query and needs to decide what user wants

Query #	Query
q_1	*"Smith CE"*
q_2	*"Smith"*

Database System Strategy

	e_1	e_2	e_3
q_1	0	1	0
q_2	0.5	0	0.5

Intent #	Intent
e_1	*ans(y)← Grades(x,'Smith', 'EE', y)*
e_2	*ans(y)← Grades(x,'Smith', 'CE', y)*
e_3	*ans(y)← Grades(x,'Smith', 'CS', y)*

Sarah Smith in CE

Grades

First_Name	Last_Name	Dept.	Grade
...	...	...	...
Sarah	Smith	CE	A
John	Smith	EE	B
Kerry	Smith	CS	D
...	...	...	...

- Row-stochastic mapping from queries to intents

Payoff: expected effectiveness of communicating every intent

Intent #	Intent
e_1	*John Smith in EE*
e_2	*Sarah Smith in CE*
e_3	*Kerry Smith in CS*

Query #	Query
q_1	*"Smith CE"*
q_2	*"Smith"*

User Strategy

	q_1	q_2
e_1	0	1
e_2	1	0
e_3	0	1

Database Strategy

	e_1	e_2	e_3
q_1	0	1	0
q_2	0.5	0	0.5

$$r(U,D) = \sum_{i=1}^{m} \boxed{\pi_i} \sum_{j=1}^{n} U_{ij} \sum_{\ell=1}^{o} D_{j\ell}\ prec(e_i, e_\ell)$$

- Prior on how often intents are queried for

Oregon State University

Payoff: expected effectiveness of communicating every intent

Intent #	Intent
e_1	*John Smith in EE*
e_2	*Sarah Smith in CE*
e_3	*Kerry Smith in CS*

Query #	Query
q_1	*"Smith CE"*
q_2	*"Smith"*

User Strategy

	q_1	q_2
e_1	0	1
e_2	1	0
e_3	0	1

Database Strategy

	e_1	e_2	e_3
q_1	0	1	0
q_2	0.5	0	0.5

$$r(U, D) = \sum_{i=1}^{m} \pi_i \sum_{j=1}^{n} U_{ij} \sum_{\ell=1}^{o} D_{j\ell}\ prec(e_i, e_\ell)$$

- Computed using user feedback, such as clicks
- Any user satisfaction metric can be used

Oregon State University

What algorithms model user learning?

- Research in psychology and empirical game theory shows that humans exhibit reinforcement learning behavior
- Components of reinforcement learning:
  - Select a query based on its past success, i.e., exploitation.
  - Explore and try new/ less successful queries to gain new knowledge, i.e., exploration.
    - Sacrifice immediate success for more success in the long run.

We evaluate user learning using human learning algorithms from empirical game theory.

- These algorithms generally differ in
  - How much they use past interactions
    - ✦ **Short-Term Memory** - Only remembers most recent interaction
    - ✦ **Long-Term Memory** - Remembers all of previous interactions
  - The degree of exploration versus exploitation
  - Reinforcement formula: e.g., use payoff versus discounted payoff.

Empirical evaluation of user learning methods

- Dataset
  - Yahoo! interaction history of ~200,000 interactions (101 hours)
  - Each interaction record contains: Query entered, Timestamp, User ID, Returned urls, which results were clicked, and which clicks are **not** noise.
  - It can model database users as our users do **not** know the schema.
- Experiment Design
  - Train and test the algorithms on how accurately they predict what the user will do next, given the previous interactions

Roth and Erevs Method closely resembles user learning

- Reinforces a query based on its payoff.

- Picks a query **randomly** to express an intent with a probability proportional to its accumulated success **(exploration)**

Method	Mean Squared Error
Win-Stay/Lose-Randomize	0.0713
Latest Reward	0.3421
Bush and Mosteller's	0.0673
Cross's	0.0686
Roth and Erev	0.0666
Roth and Erev Modified	0.0666
UCB-1	0.1624

Roth and Erevs Method closely resembles user learning

- As it picks queries randomly, it may use new/ less frequently used queries once in a while (**exploration).**

Method	Mean Squared Error
Win-Stay/Lose-Randomize	0.0713
Latest Reward	0.3421
Bush and Mosteller's	0.0673
Cross's	0.0686
Roth and Erev	0.0666
Roth and Erev Modified	0.0666
UCB-1	0.1624

How should the DBMS learn and adapt its strategy?

- Web search systems use reinforcement learning algorithms, e.g., UCB-1
  - They assume that user does not learn to change her strategy
- Intuitive answer:
  - User and DBMS have identical interest, so user learning only helps.
  - Thus, DBMS may use current online learning methods.

How should the DBMS learn and adapt its strategy?

- Intuitive answer:
  - User and DBMS have identi[illegible] in[illegible]est, so user learning only helps.
  - Thus, DBMS may use curr[illegible]e learning methods.
- **Wrong!!**

User/DBMS may trap in cycles and **not** communicate effectively

- Intuitive answer:
    - User and DBMS have identical intere… …er learning only helps.
    - Thus, DBMS may use current onlin… …methods used in IR.
- **Wrong!!**

1. There are games in which players learn and collaborate but effectiveness **decreases** over time!
    - The players may get trapped in a cycle
2. Current online learning algorithms, e.g., UCB-1, assume that users do **not** learn and have a **fixed** strategy
    - They cannot discover user intents accurately where users learn (**dynamic environment**)

How our DBMS algorithm works

Students

	name	year	dept_name	school
e1	Kerry Smith	Senior	CS	EECS
e2	John Smith	Junior	EE	EECS
e3	Sarah Smith	Senior	CE	EECS

Database

- This is a toy example to illustrate the learning algorithm

The reward matrix

Q: Smith

Students

	name	year	dept_name	school
e1	Kerry Smith	Senior	CS	EECS
e2	John Smith	Junior	EE	EECS
e3	Sarah Smith	Senior	CE	EECS

- Keeps track of all reward accumulated
- Initialized to all 1 for this example

Reward Matrix

	e_1	e_2	e_3
q_1	1	1	1
q_2	1	1	1

DBMS strategy is constructed from the reward matrix

Q: Smith

Students

	name	year	dept_name	school
e1	Kerry Smith	Senior	CS	EECS
e2	John Smith	Junior	EE	EECS
e3	Sarah Smith	Senior	CE	EECS

Database Strategy

	e_1	e_2	e_3
q_1	1		
q_2			

Reward Matrix

	e_1	e_2	e_3
q_1	1	1	1
q_2	1	1	1

- $D_{ij} = R_{ij} / \text{sum}(R_i)$
- $D_{11} = 1 / \text{sum}(R_i)$

DBMS strategy is constructed from the reward matrix

Q: Smith

Students

	name	year	dept_name	school
e1	Kerry Smith	Senior	CS	EECS
e2	John Smith	Junior	EE	EECS
e3	Sarah Smith	Senior	CE	EECS

Database Strategy

	e_1	e_2	e_3
q_1	0.33		
q_2			

Reward Matrix

	e_1	e_2	e_3
q_1	1	1	1
q_2	1	1	1

- $D_{ij} = R_{ij} / \text{sum}(R_i)$
- $D_{11} = 1 / 3$

DBMS strategy is constructed from the reward matrix

Q: Smith

Students

	name	year	dept_name	school
e1	Kerry Smith	Senior	CS	EECS
e2	John Smith	Junior	EE	EECS
e3	Sarah Smith	Senior	CE	EECS

Database Strategy

	e_1	e_2	e_3
q_1	0.33	0.33	0.33
q_2	0.33	0.33	0.33

Reward Matrix

	e_1	e_2	e_3
q_1	1	1	1
q_2	1	1	1

- $D_{ij} = R_{ij} / \text{sum}(R_i)$

DBMS returns results based on its random strategy

Students

	name	year	dept_name	school
e1	Kerry Smith	Senior	CS	EECS
e2	John Smith	Junior	EE	EECS
e3	Sarah Smith	Senior	CE	EECS

Database Strategy

	e_1	e_2	e_3
q_1	0.33	0.33	0.33
q_2	0.33	0.33	0.33

Reward Matrix

	e_1	e_2	e_3
q_1	1	1	1
q_2	1	1	1

- User submits q1
- DBMS returns e1 to the user randomly with a probability of 0.33
  - As opposed to the current systems, it does **not** return the top-K answers.

Feedback from user updates the reward matrix

Students

	name	year	dept_name	school
e1	Kerry Smith	Senior	CS	EECS
e2	John Smith	Junior	EE	EECS
e3	Sarah Smith	Senior	CE	EECS

Database Strategy

	e_1	e_2	e_3
q_1	0.33	0.33	0.33
q_2	0.33	0.33	0.33

Reward Matrix

	e_1	e_2	e_3
q_1	2	1	1
q_2	1	1	1

- It satisfies the user, they give some feedback such as a click
- Add add 1 to the reward matrix
- Reward matrix is updated

The DBMS strategy is updated from the reward matrix

Students

	name	year	dept_name	school
e1	Kerry Smith	Senior	CS	EECS
e2	John Smith	Junior	EE	EECS
e3	Sarah Smith	Senior	CE	EECS

- Use reward matrix to update database strategy
- Q1,E1 is reinforced since user gave good feedback
- All other intents for that query have their probabilities implicitly reduced

Database Strategy

	e_1	e_2	e_3
q_1	0.5	0.25	0.25
q_2	0.33	0.33	0.33

Reward Matrix

	e_1	e_2	e_3
q_1	2	1	1
q_2	1	1	1

Oregon State University

How our algorithm works

Students

	name	year	dept_name	school
e1	Kerry Smith	Senior	CS	EECS
e2	John Smith	Junior	EE	EECS
e3	Sarah Smith	Senior	CE	EECS

Database Strategy

	e_1	e_2	e_3
q_1	0.5	0.25	0.25
q_2	0.33	0.33	0.33

Reward Matrix

	e_1	e_2	e_3
q_1	2	1	1
q_2	1	1	1

- However, there may be so many intents and queries to make this impractical
- We keep features for the queries and intents in practice
  - Such as n-grams of the query and tuples

Theoretical guarantees

- **Theorem:** If user and database system learn use the Roth and Erev method, the payoff of the game will remain the same or increase.
  - The result also holds if the user does **not** learn and has a fixed strategy.
    - Slow learners!
  - The sequence of payoffs converges stochastically sparking (almost surely).

DBMS learning and query processing are inefficient for DB with multiple tables

Q: Smith CS

Department

dept_id	name	school
1	CS	EECS
2	EE	EECS

⋈

Students

name	year	dept_id
Kerry Smith	Senior	1
Bob Smith	Junior	1

=

Students ⋈ Department

name	year	dept_id	name	school	Prob
Kerry Smith	Senior	1	CS	EECS	P1
Bob Smith	Junior	1	CS	EECS	P2

- The keyword query is sent to each table.
- The answers are in the join of matching tuples from different tables.
- The join must be materialized to compute probabilities and then sampled.
- DBMS may have to do several joins as it does **not** know the join user is looking for.

We leverage sampling over join techniques to improve efficiency

- We use **reservoir sampling** to eliminate the need for join materialization.

Q: Smith CS

Department

dept_id	name	school
1	CS	EECS
2	EE	EECS

⋈

Students

name	year	dept_id
Kerry Smith	Senior	1
Bob Jones	Junior	1

Students ⋈ Department

=

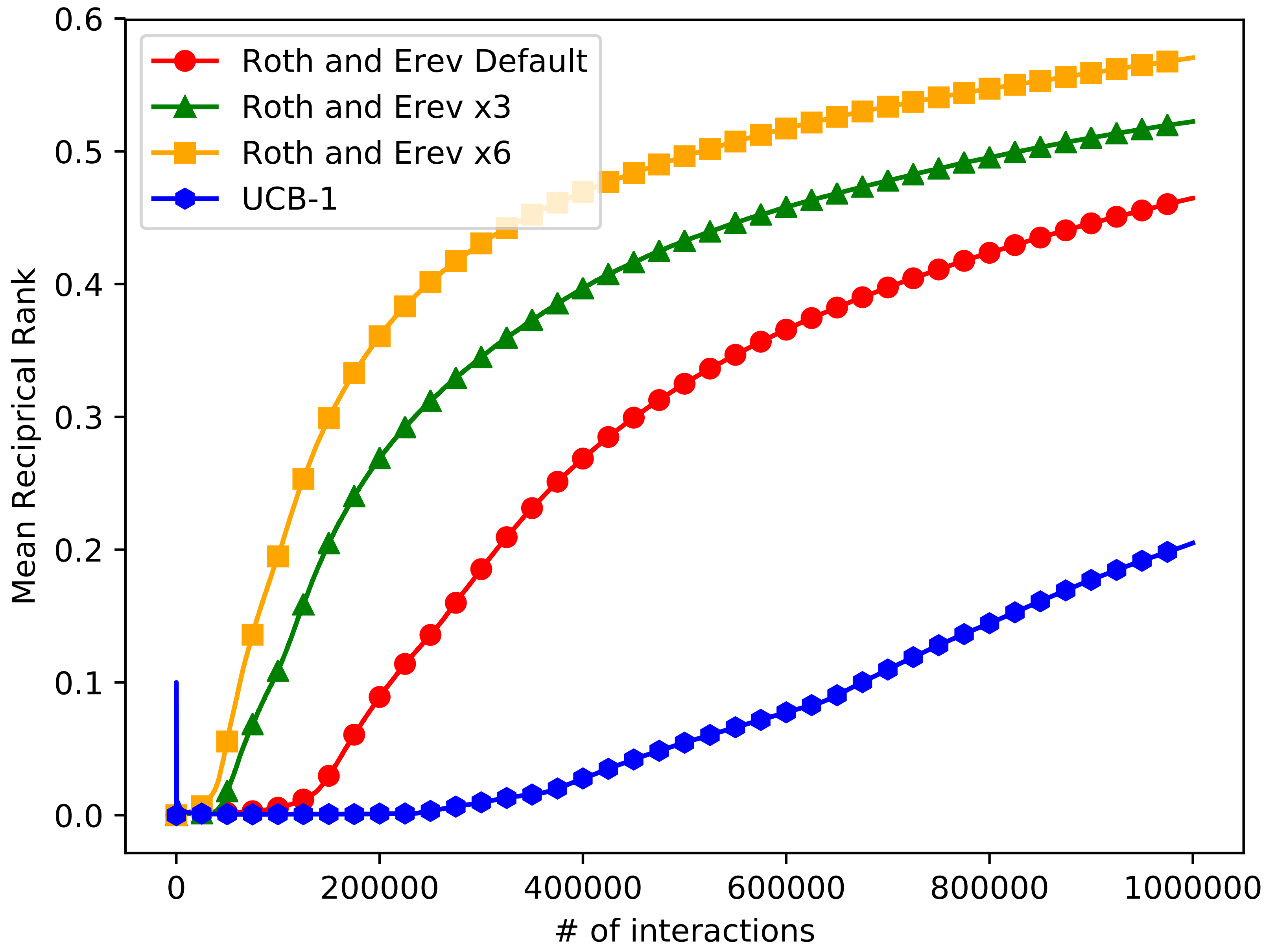
name	year	dept_id	name	school	Prob
Kerry Smith	Senior	1	CS	EECS	P1

- We can be smarter and first sample tuples from the base tables and then join only the sampled tuples. (**Poisson-Olken algorithm**)
  - Joins significantly fewer tuples.
  - Details in the paper.

Evaluating Effectiveness: Experiment setting

- Dataset
  - Yahoo! query workload, same format as the user study
  - Used to create queries and a database of URLs
- Experimental Setup
  - User learns based on Roth and Erev during interaction
  - Use Mean Reciprocal Rank to measure effectiveness

Our learning algorithm outperforms UCB-1 in the long run



Oregon State University

Experimental evaluation: Efficiency

- We use subsets of Freebase database
  - e.g., **TV Program:** 7 tables and 291,026 tuples
- We use subsets of from the Bing query log whose relevant answers are in these databases.
  - e.g., 621 queries over TV Program
- Run for 1000 interactions

Database	Reservoir (sec)	Poisson-Olken (sec)
TV Program	0.298	0.171

Conclusion & Future Work

- The interaction between user and DBMS is better modeled as a collaborative game.
- DBMS should use randomized learning strategies, considering the user learns.
- We use sampling over join to efficiently implement DBMS learning.
- Data integration between databases is the next step
    - Where databases communicate to establish a common mapping.