Oregon State University

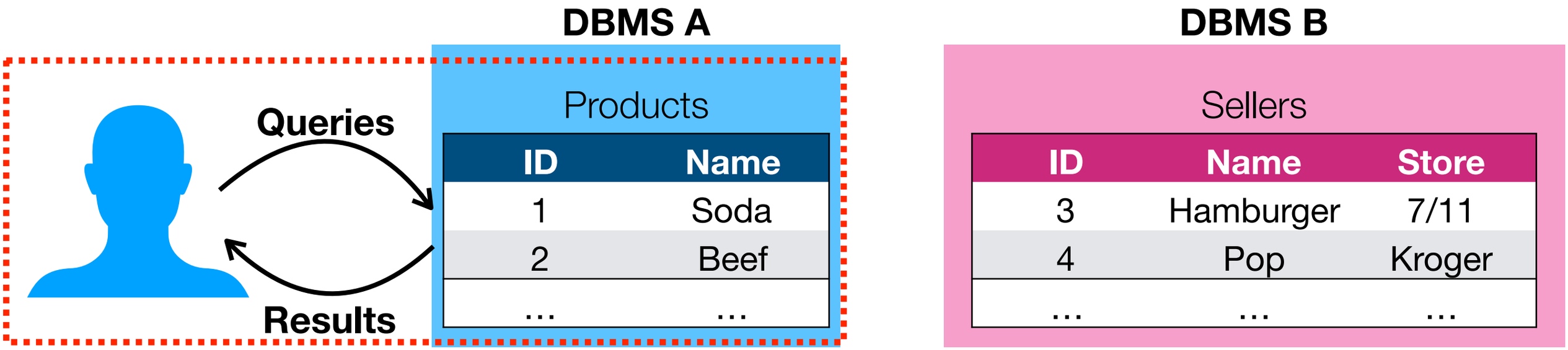
Progressive Interaction for Autonomous Entity Matching

Ben McCamish, Arash Termehchy

Oregon State University

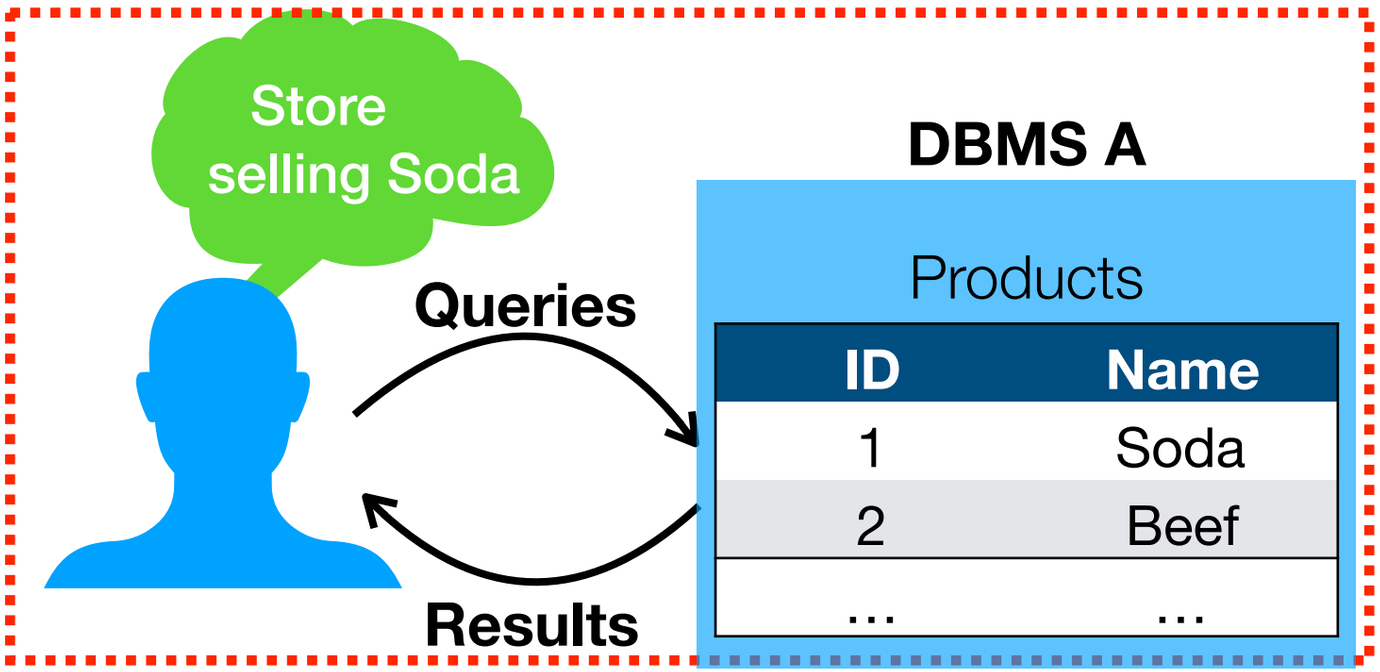
Information & **D**ata Manag**e**ment and **A**nalytics Laboratory (IDEA)

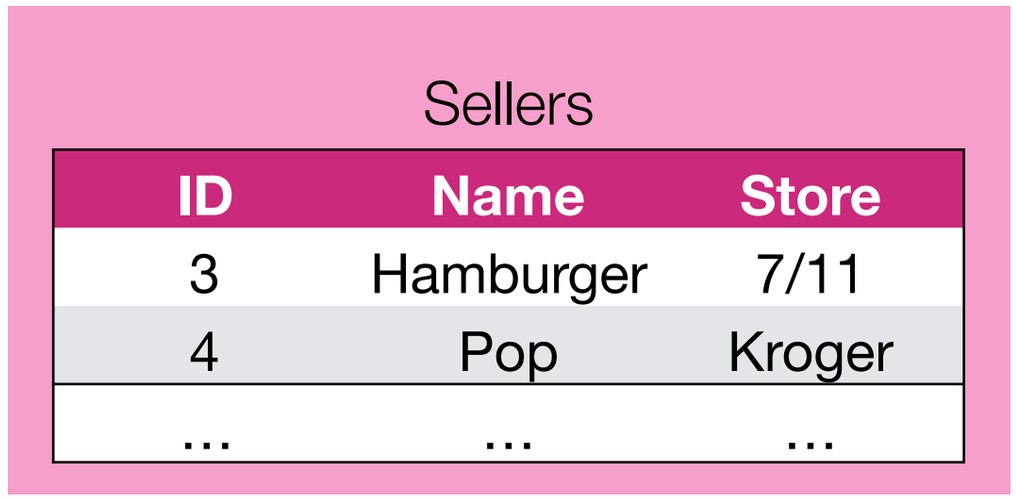
User interacts with local data source



- User interacts with DBMS A by using some query interface
  - They express their intents, what they are looking for
- Then the results are presented to the user

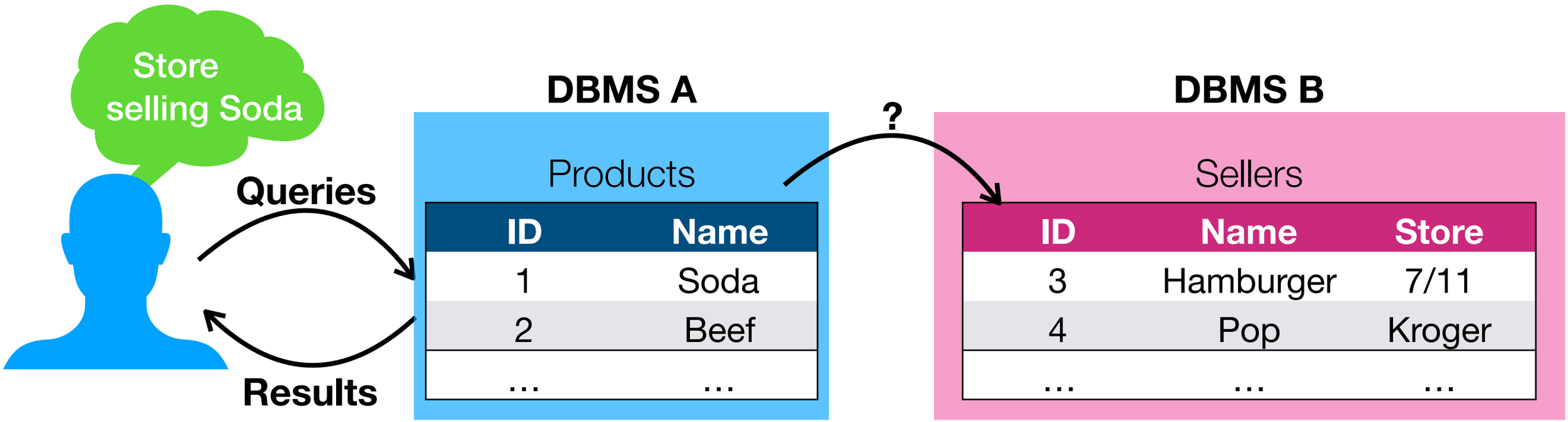
DBMS A not able to satisfy query





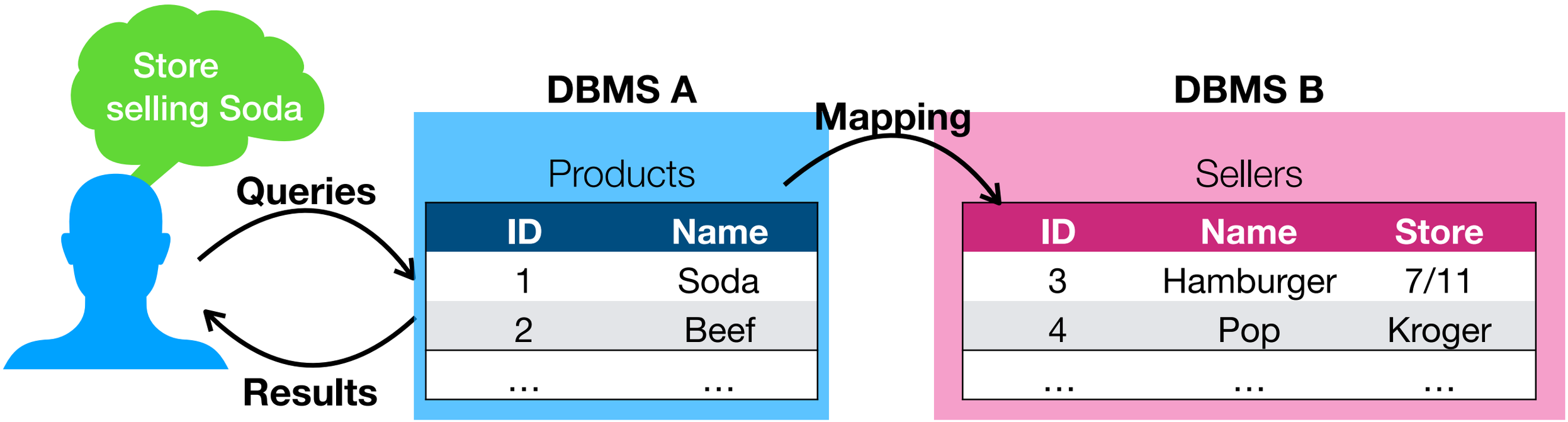
- User queries its local data source, DBMS A
- DBMS A does **not** have the desired information
- Must find the desired information in external data source, DBMS B

DBMS A cannot query



- DBMS A needs to submit queries to DBMS B
- DBMS B schema and representation of entities is different
- DBMS A **does not** know schema or representation
  - Cannot properly formulate queries

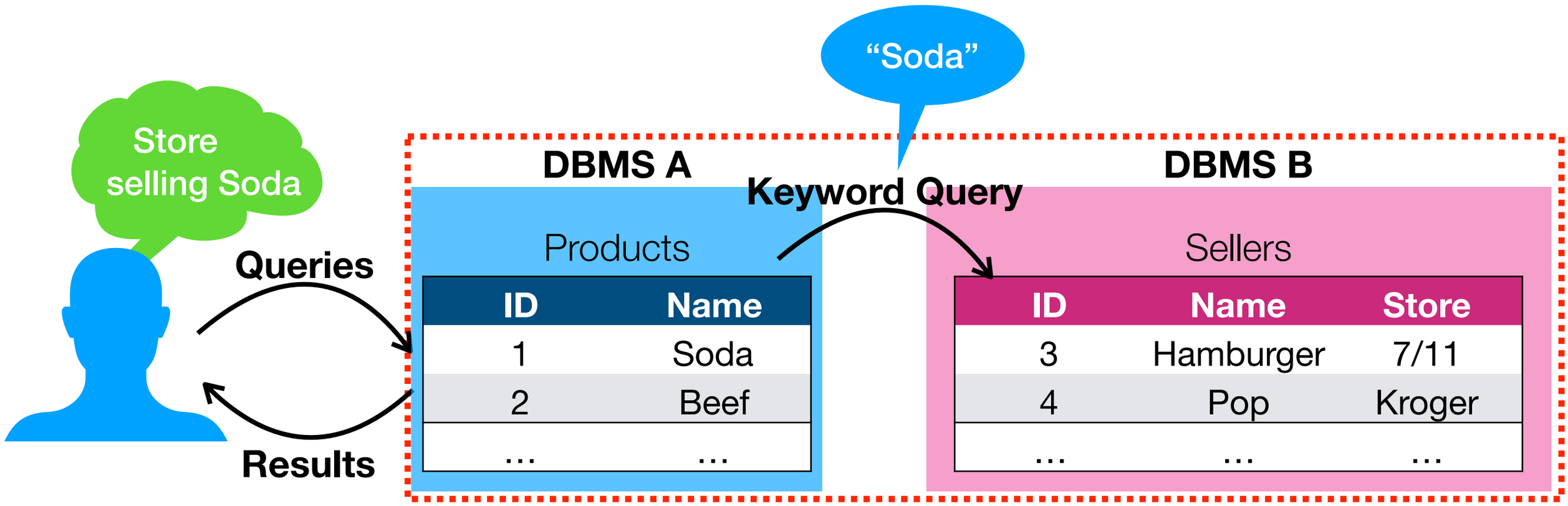
DBMS A queries DBMS B



- Traditionally a mapping between two DBMSs
- However this is **costly**
  - Needs to be updated when the schema changes, manually
  - Manually develop this mapping, takes time

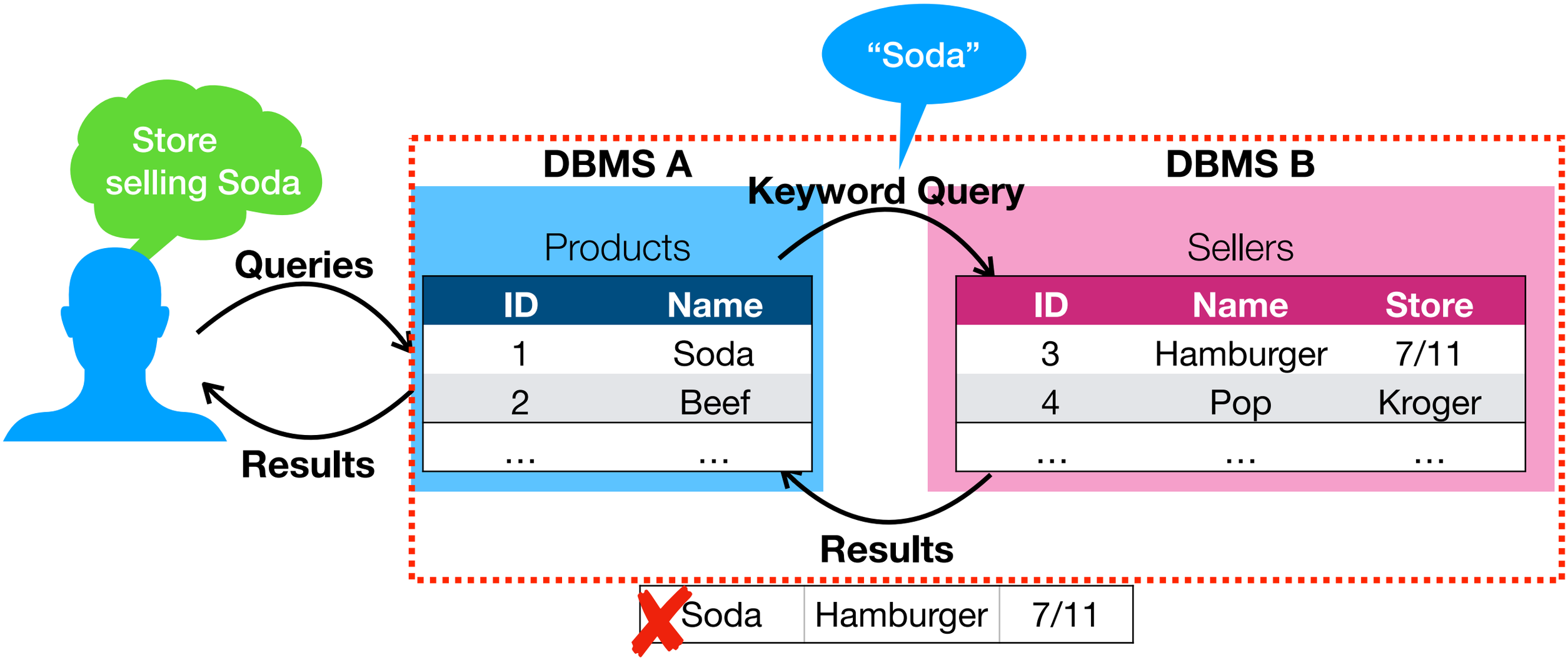
Oregon State University

What if DBMS A learns through interactions?



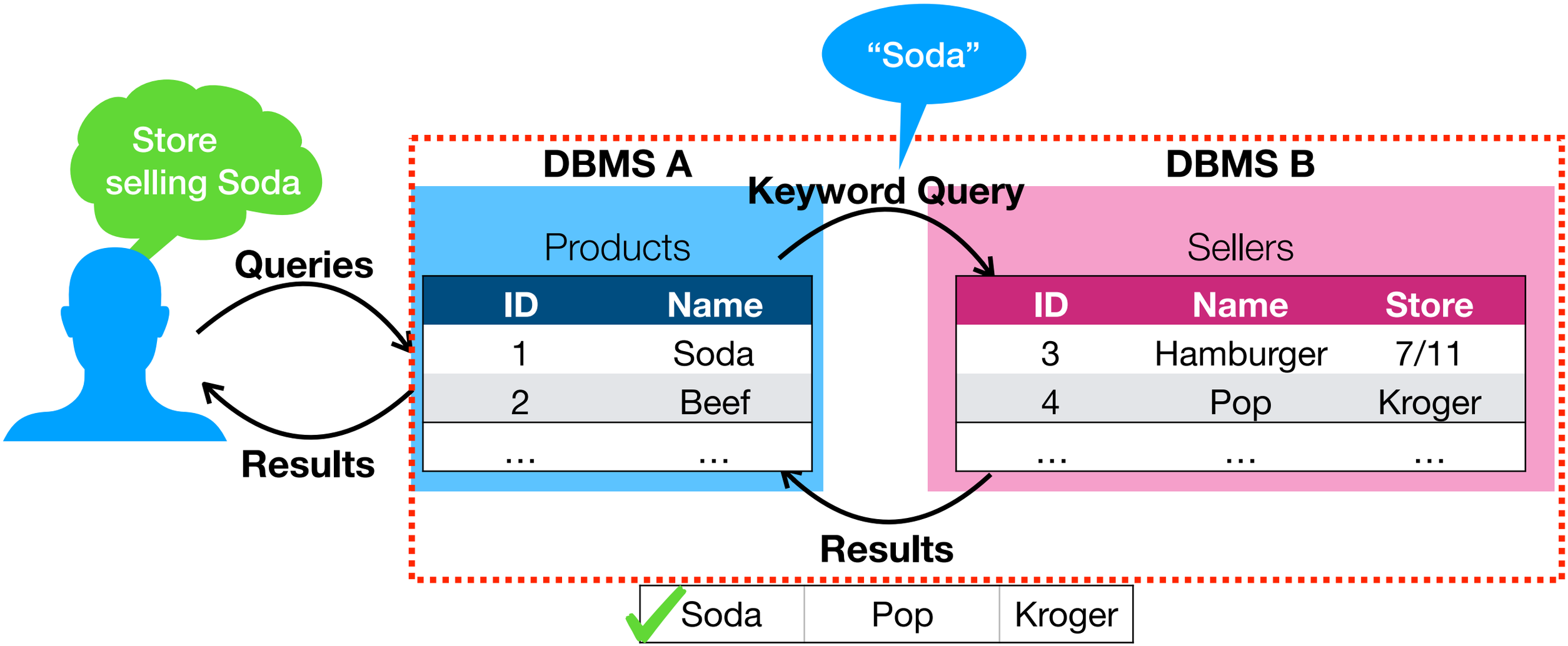
- DBMS A wants to find similar entities in other DBMS, sends some query
- There is often a common query language
  - **Keyword Queries**
- Other DBMSs understand this, but results are not very effective

Results are returned



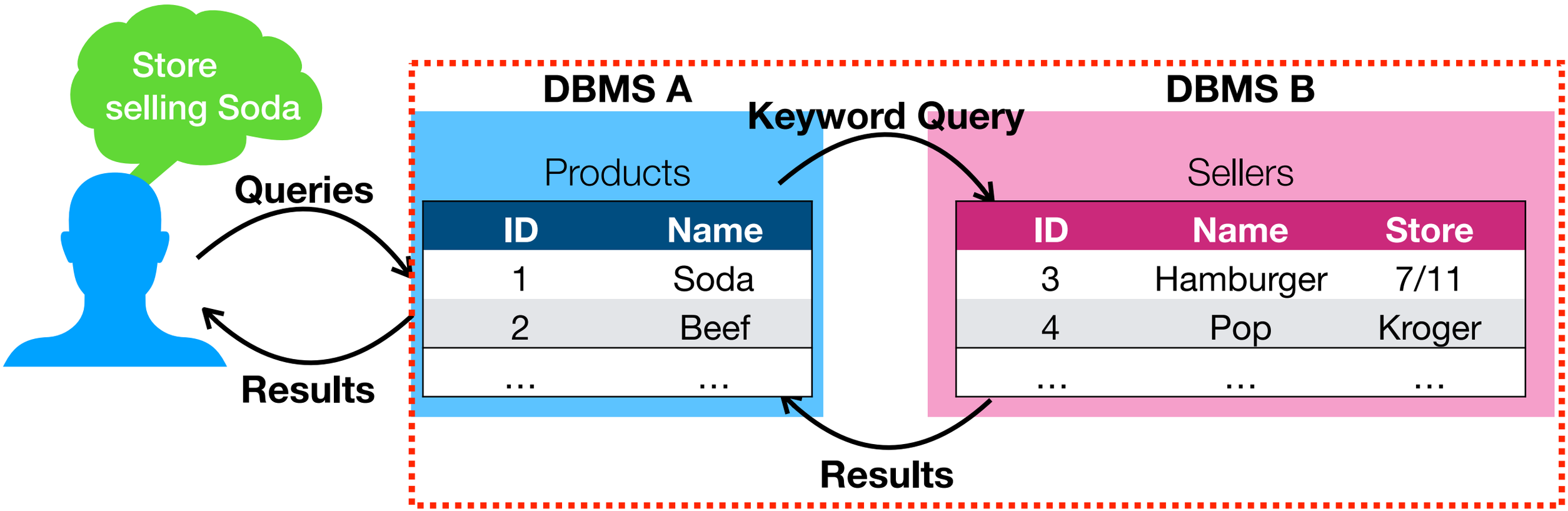
- Results are returned to the user
- User gives some feedback on the results
  - This **is not** what the user is looking for

Results are returned



- Results are returned to the user
- User gives some feedback on the results
  - This **is** the answer the user wanted

Utilize the feedback and learn



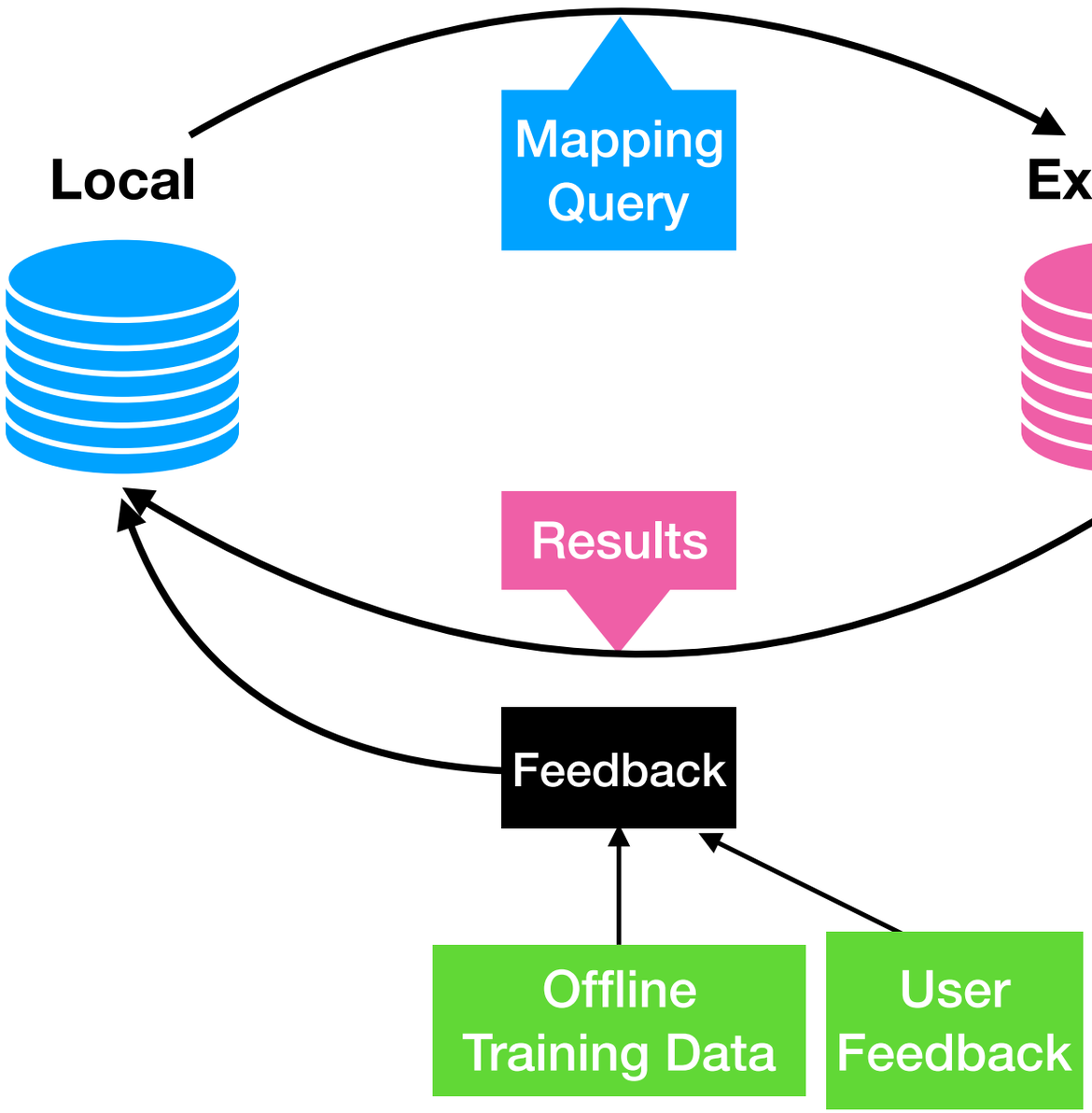
- Can build the mapping over time through interaction and feedback

- **Our Goal:** Learn this mapping between DBMS A and DBMS B

- **Method:** Establish a common language or means of communication between the two DBMSs

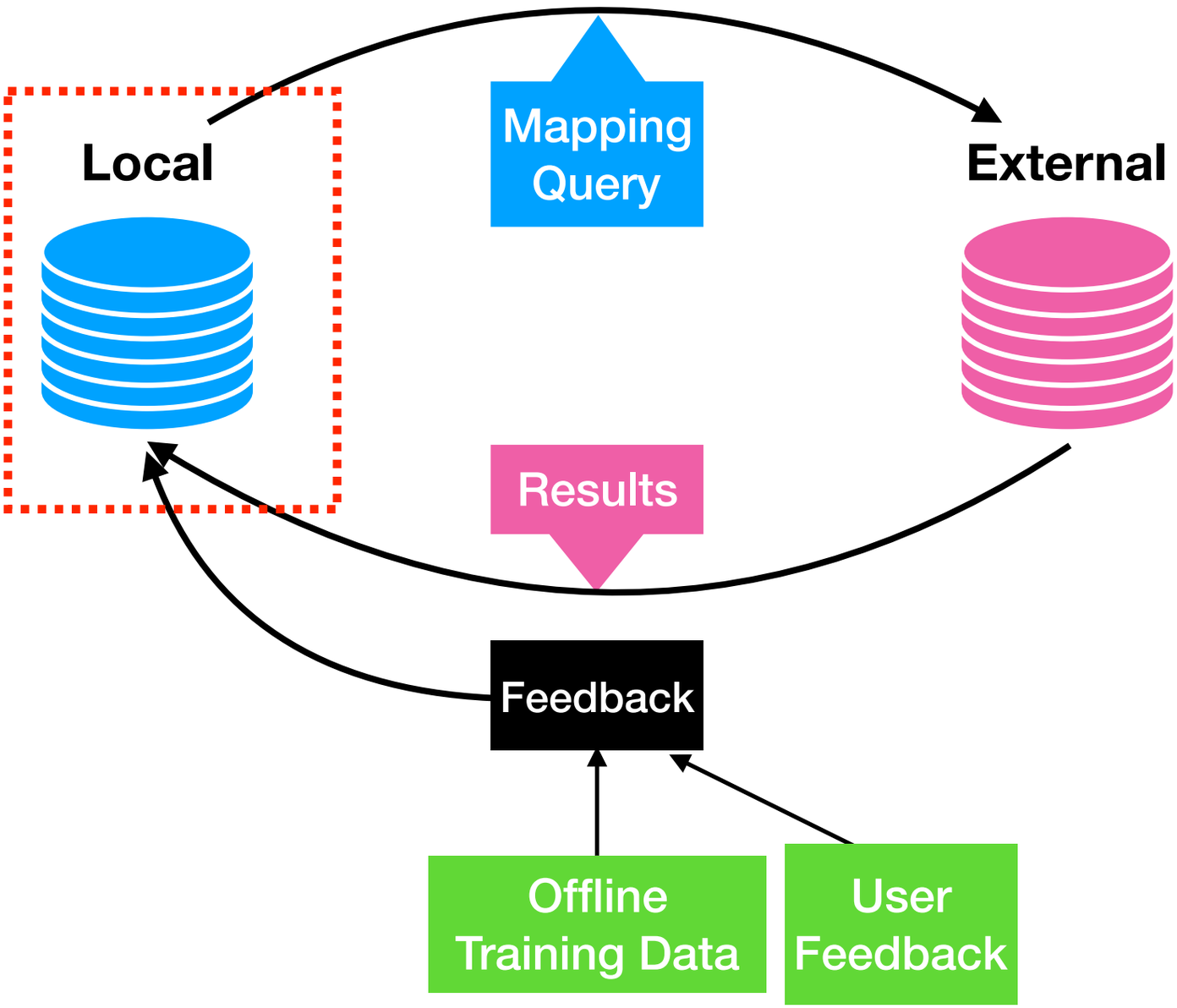
Oregon State University

Our Framework



- Local and External DBMS
- Communicate via keyword queries and results

Intents



Products

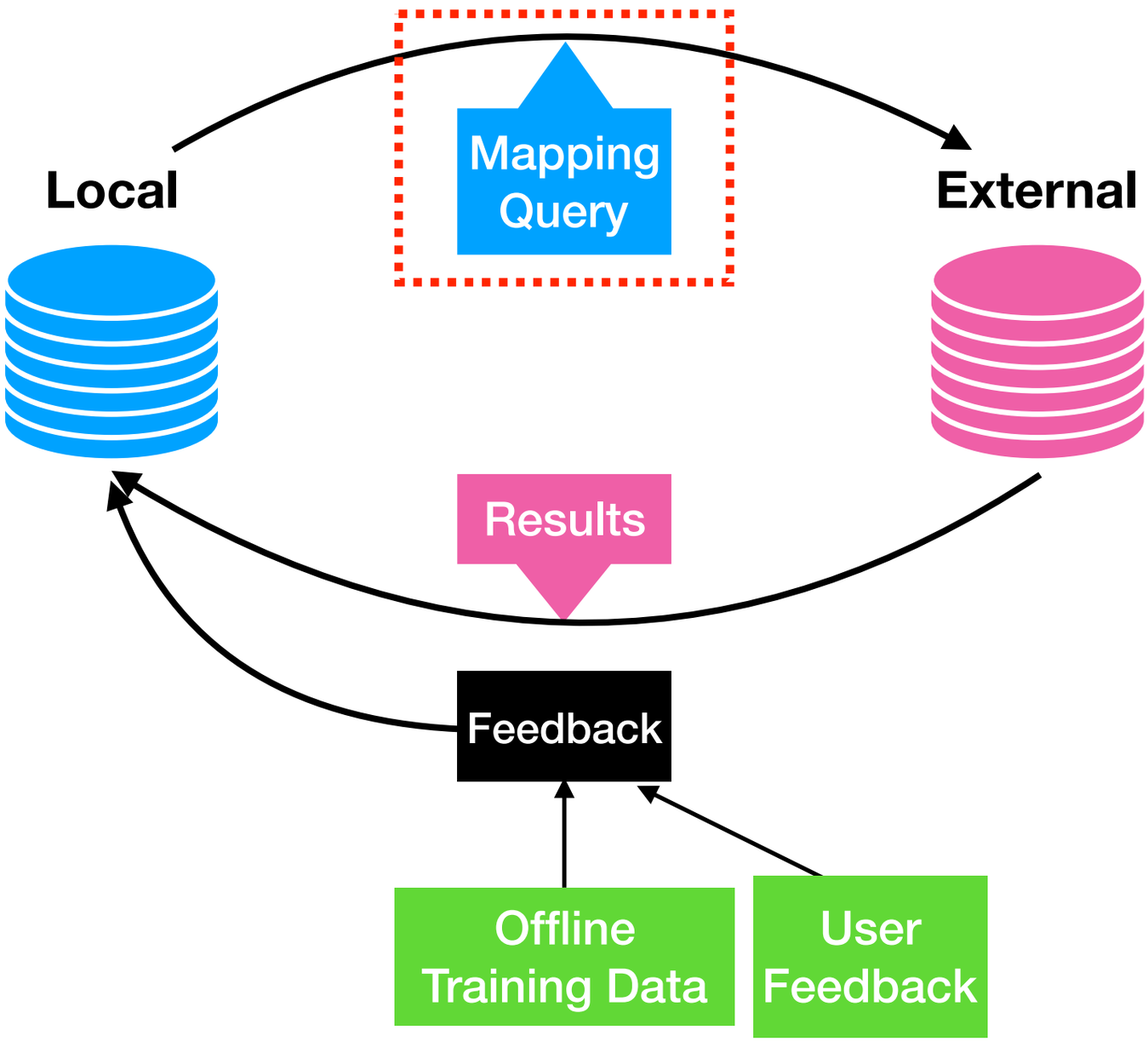
ID	Name
1	Soda
2	Beef

Local DBMS Intents

Intent #	Intent
e1	1 Soda
e2	2 Beef

- Local DBMS has intents
- Defined by the user
- Doesn’t require user however

Mapping Queries



DBMS A Queries

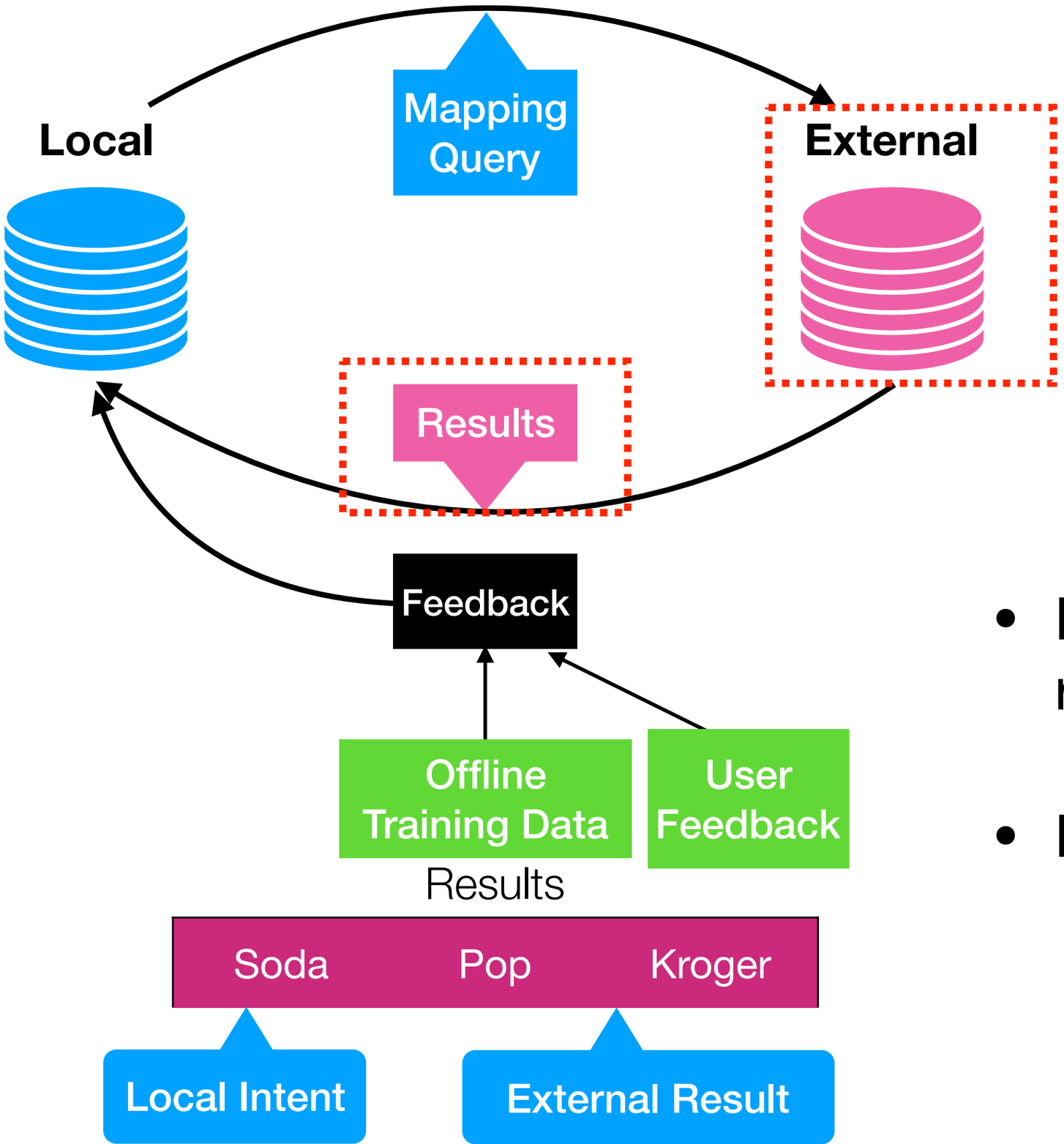
Query #	Query
s1	1 soda
s2	2 beef
s3	soda
s4	beef

Strategy

	s1	s2	s3	s4
e1	0.5	0.1	0.4	0
e2	0	0.4	0.3	0.3

- Sends keyword queries
- Called Mapping Queries

Returned Results



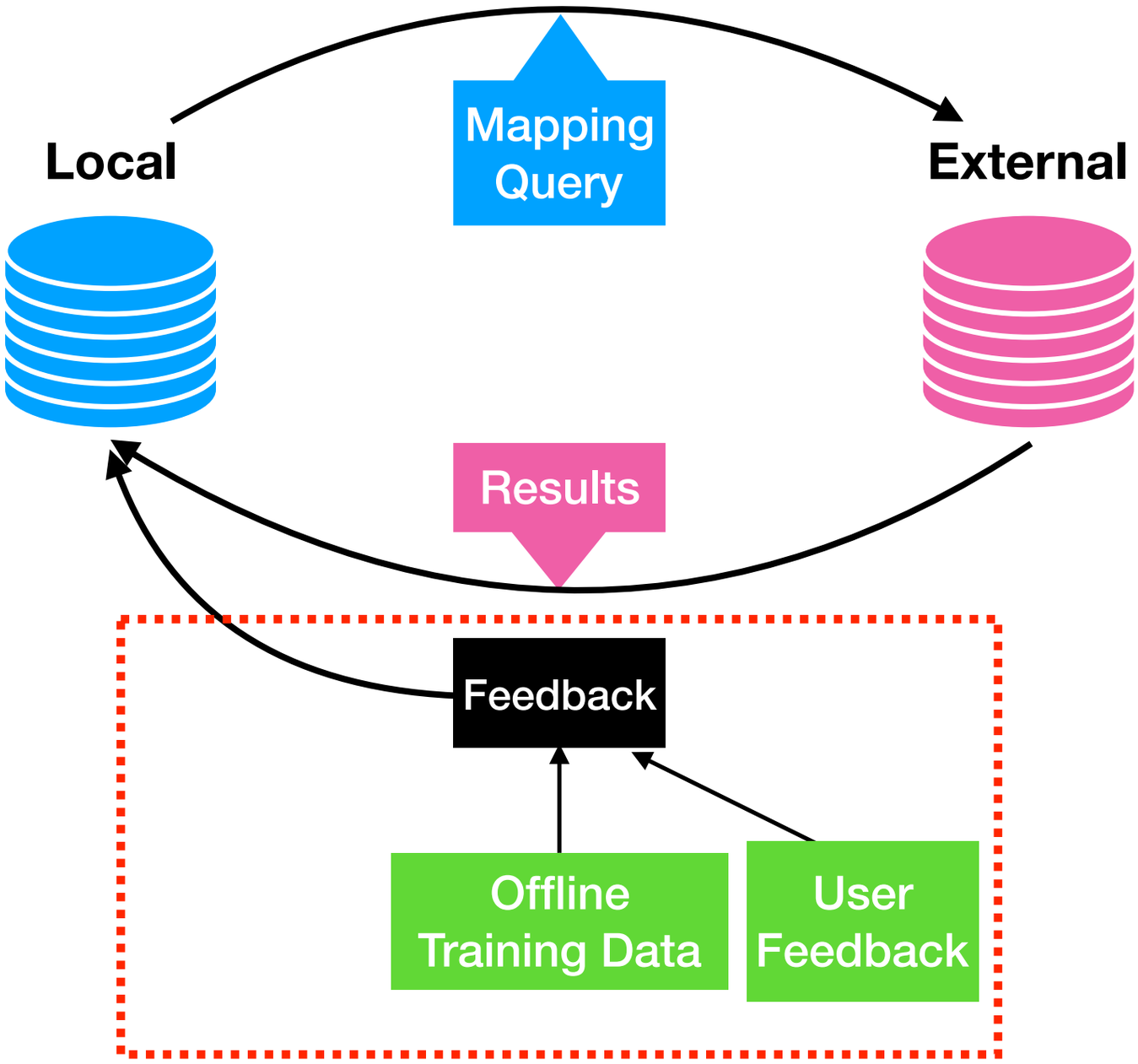
Sellers

ID	Name	Store
3	Hamburger	7/11
4	Pop	Kroger
…	…	…

- External DBMS returns some results
- External DBMS can also learn

Oregon State University

Feedback



- Feedback on whether the returned results are correct
- Can come from user, but doesn't have to
- Can use a model built on previous user feedback

Local DBMS Strategy

Local DBMS

Intents

Intent #	Intent
e1	1 Soda
e2	2 Beef

Mapping Queries

Query #	Query
s1	1 soda
s2	2 beef
s3	soda
s4	beef

Strategy

	s1	s2	s3	s4
e1	0.5	0.1	0.4	0
e2	0	0.4	0.3	0.3

Products

ID	Name
1	Soda
2	Beef

External DBMS

Sellers

ID	Name	Store
3	Hamburger	7/11
4	Pop	Kroger
...	...	...

- Local DBMS has a strategy to send queries for intents
- External DBMS may also have a strategy

Local DBMS Strategy

Local DBMS

Intents

Intent #	Intent
e1	1 Soda
e2	2 Beef

Mapping Queries

Query #	Query
s1	1 soda
s2	2 beef
s3	soda
s4	beef

Strategy

	s1	s2	s3	s4
e1	0.5	0.1	0.4	0
e2	0	0.4	0.3	0.3

Products

ID	Name
1	Soda
2	Beef

External DBMS

Sellers

ID	Name	Store
3	Hamburger	7/11
4	Pop	Kroger
...	...	...

- Suppose local DBMS has the intent e1

Local DBMS Strategy

Local DBMS

Intents

Intent #	Intent
e1	1 Soda
e2	2 Beef

Mapping Queries

Query #	Query
s1	1 soda
s2	2 beef
s3	soda
s4	beef

Strategy

	s1	s2	s3	s4
e1	0.5	0.1	0.4	0
e2	0	0.4	0.3	0.3

Products

ID	Name
1	Soda
2	Beef

External DBMS

Sellers

ID	Name	Store
3	Hamburger	7/11
4	Pop	Kroger
...	...	...

- Consults strategy to see what mapping query to send
- Sends s3 with 0.4 probability

Local DBMS Strategy

Local DBMS

Intents

Intent #	Intent
e1	1 Soda
e2	2 Beef

Mapping Queries

Query #	Query
s1	1 soda
s2	2 beef
s3	soda
s4	beef

Strategy

	s1	s2	s3	s4
e1	0.5	0.1	0.4	0
e2	0	0.4	0.3	0.3

Products

ID	Name
1	Soda
2	Beef

External DBMS

Sellers

ID	Name	Store
3	Hamburger	7/11
4	Pop	Kroger
...	...	...

- When results are returned and feedback given, strategy is updated
- Uses reinforcement learning method

Reinforcement Learning

- Select a query based on 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

Reinforcing Local Strategy

Local DBMS

Intents

Intent #	Intent
e1	1 Soda
e2	2 Beef

Mapping Queries

Query #	Query
s1	1 soda
s2	2 beef
s3	soda
s4	beef

Strategy

	s1	s2	s3	s4
e1	0.5	0.1	0.4	0
e2	0	0.4	0.3	0.3

Products

ID	Name
1	Soda
2	Beef

External DBMS

Sellers

ID	Name	Store
3	Hamburger	7/11
4	Pop	Kroger
...	...	...

- The probabilities of queries allow for exploration and exploitation

Reinforcing Local Strategy

Local DBMS

Intents

Intent #	Intent
e1	1 Soda
e2	2 Beef

Mapping Queries

Query #	Query
s1	1 soda
s2	2 beef
s3	soda
s4	beef

Strategy

	s1	s2	s3	s4
e1	0.5	0.1	0.4	0
e2	0	0.4	0.3	0.3

Products

ID	Name
1	Soda
2	Beef

External DBMS

Sellers

ID	Name	Store
3	Hamburger	7/11
4	Pop	Kroger
...	...	...

- Suppose the feedback given for this query was positive
- Then the strategy is reinforced as such

Reinforcing Local Strategy

Local DBMS

Intents

Intent #	Intent
e1	1 Soda
e2	2 Beef

Mapping Queries

Query #	Query
s1	1 soda
s2	2 beef
s3	soda
s4	beef

Strategy

	s1	s2	s3	s4
e1	0.5	0.1	0.45	0
e2	0	0.4	0.3	0.3

Products

ID	Name
1	Soda
2	Beef

External DBMS

Sellers

ID	Name	Store
3	Hamburger	7/11
4	Pop	Kroger
...	...	...

- Increase probability for mapping query sent

Reinforcing Local Strategy

Local DBMS

Intents

Intent #	Intent
e1	1 Soda
e2	2 Beef

Mapping Queries

Query #	Query
s1	1 soda
s2	2 beef
s3	soda
s4	beef

Strategy

	s1	s2	s3	s4
e1	0.45	0.09	0.45	0
e2	0	0.4	0.3	0.3

Products

ID	Name
1	Soda
2	Beef

External DBMS

Sellers

ID	Name	Store
3	Hamburger	7/11
4	Pop	Kroger
...	...	...

- Implicitly decreases probability for others

Reinforcing Local Strategy

Local DBMS

Intents

Intent #	Intent
e1	1 Soda
e2	2 Beef

Mapping Queries

Query #	Query
s1	1 soda
s2	2 beef
s3	soda
s4	beef

Strategy

	s1	s2	s3	s4
e1	0.45	0.09	0.45	0
e2	0	0.4	0.3	0.3

Products

ID	Name
1	Soda
2	Beef

External DBMS

Sellers

ID	Name	Store
3	Hamburger	7/11
4	Pop	Kroger
...	...	...

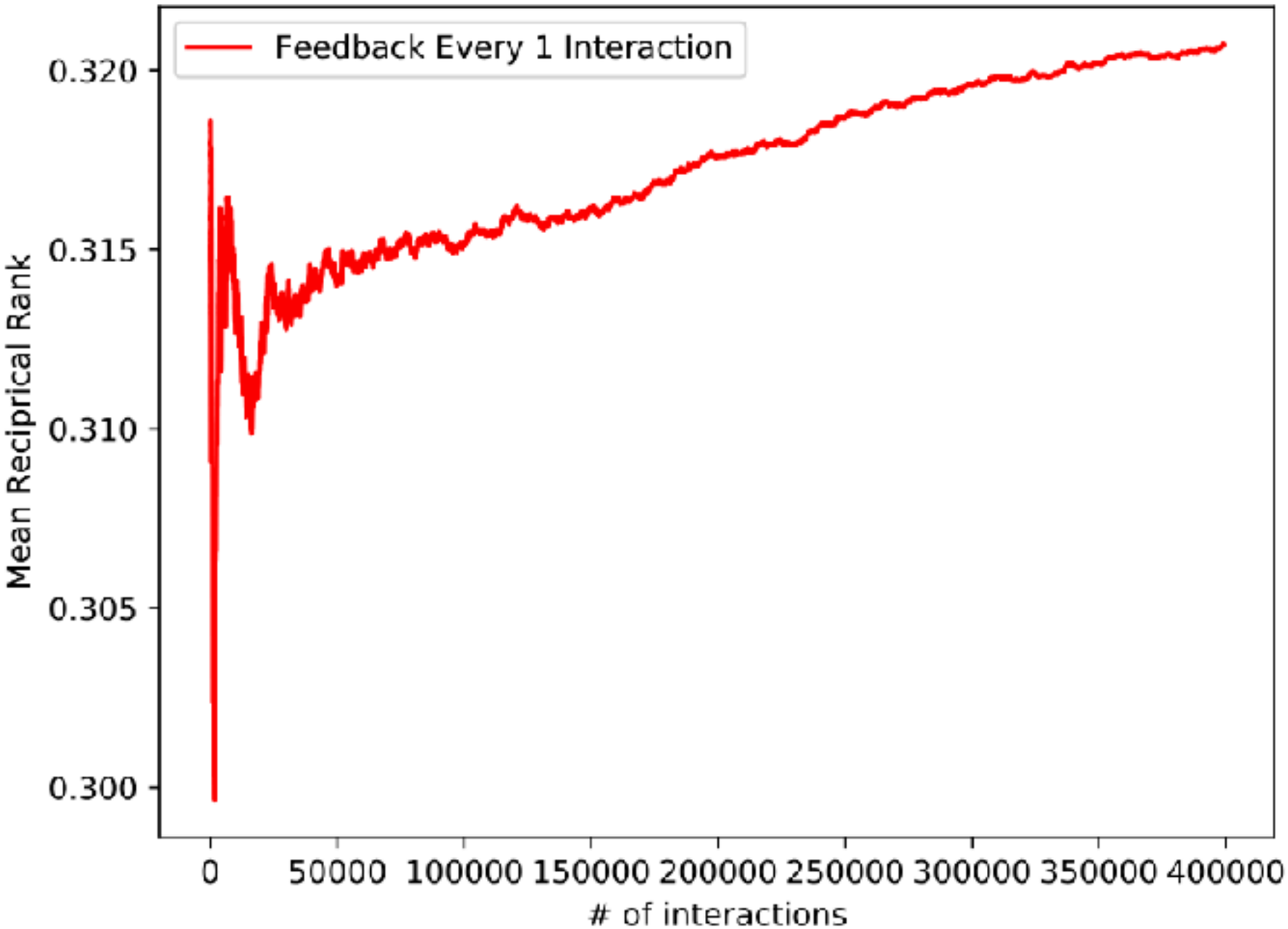
- External DBMS may also learn, but we don't focus on that here

- In both cases when the external DBMS learns and doesn't learn, it will converge, based on our previous results

Our experiments

- Use two databases, each containing information on products
  - One is an Amazon database and the other a Google database
- Approximately 1400 tuples in the Amazon and 3200 tuples in the Google dataset
- We have the ground truth, which is used as simulated user feedback
- Single tuples are used as intents and they have single match
- The receiver does not learn
- Cache simulated user feedback

Results for learning every time

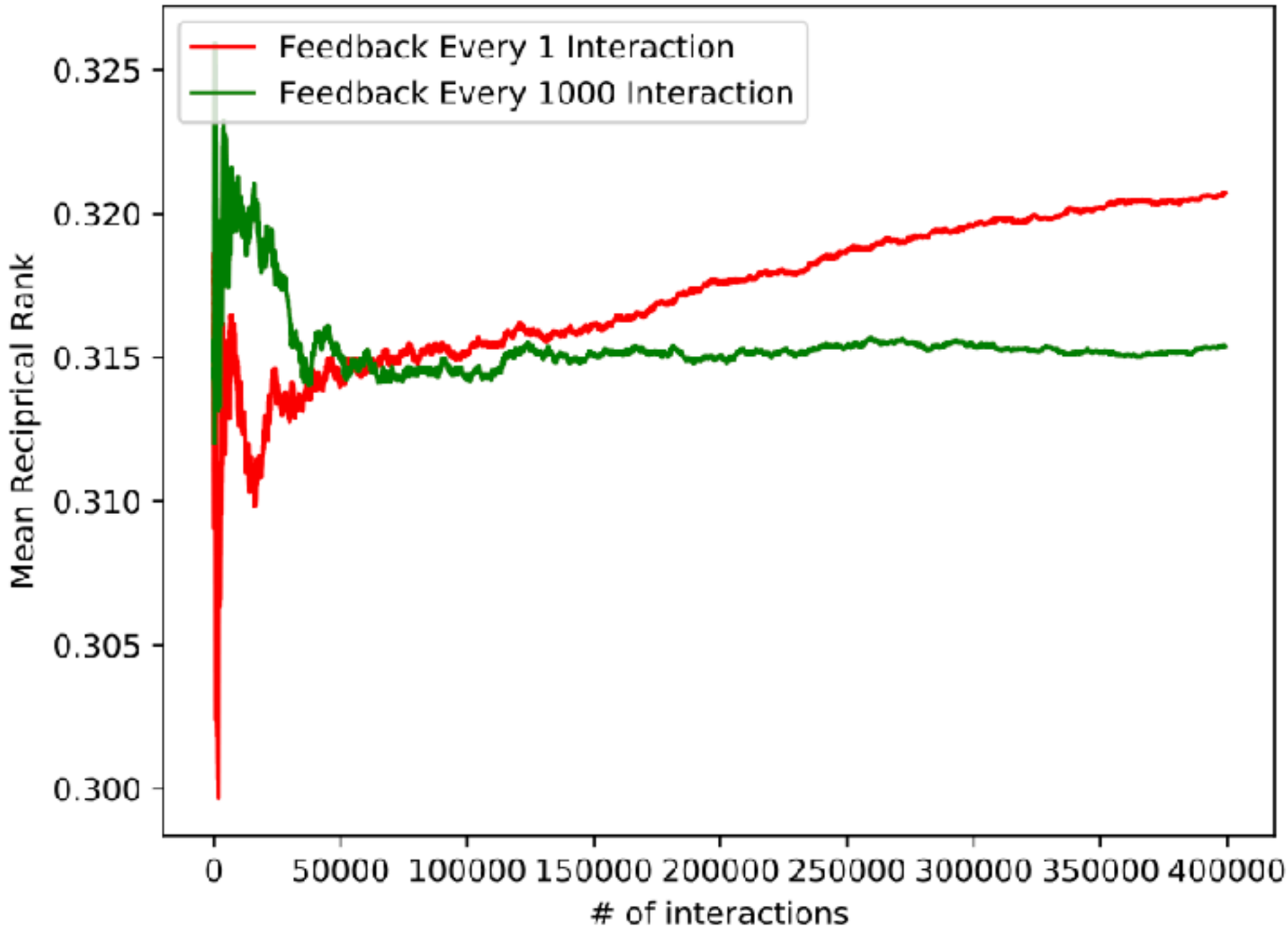


Reducing User Feedback

- Need to reduce the amount of feedback required from the user during interaction between DBMSs

- We looked at what happens when the user is only asked for feedback every 1000 interactions

Reducing User Feedback



- Stopped using user feedback after 10,000 interactions

Another way of reducing user feedback

- Create a model to generalize the feedback on similarity between entities

- At some point we stop updating this model and receiving feedback

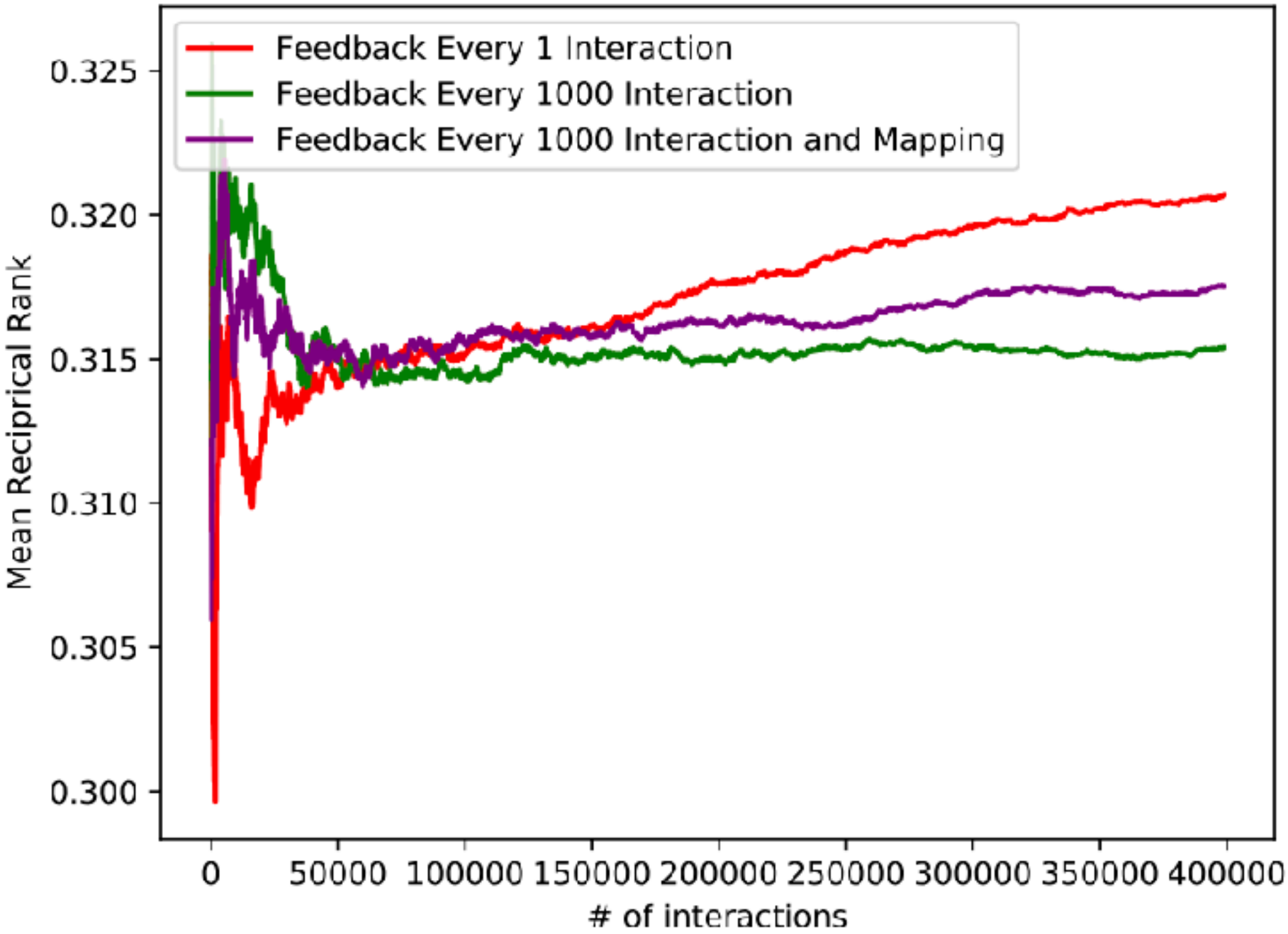
- Then we can use this model to help guide the learning when user feedback is unavailable

- The weight is updated when the user is consulted

Mapping

	pop	hamburger
soda	0.8	0.4
beef	0.3	0.9

Results of Mapping of features



Oregon State University

Open problems

- What ways can we reduce the amount of feedback from the user?
  - Using some informed semi-supervised learning
- Generalize what we learn from feedback
  - Learning a matching function so we don’t need to consult user

Open problems

- How does interaction work with more than two DBMS interacting?
- Interaction between DBMSs can happen without users
  - DBMS can interact and learn to communicate on their own, pick their own intents and continue to learn
- There may be databases with not a one to one mapping
  - Database containing information on whether people smoke
  - One may categorize as “Smoke”, “No Smoke”
  - Other may categorize as “Heavy Smoker”, “Light Smoker”, “No Smoke”, “Vape”

Open problems

- How much does the mapping query length impact the interaction over time?
  - Larger or smaller queries, changing the length over time
  - Using the returned tuples from the external DBMSs to expand vocabulary
- External DBMS may have some limitations on how many queries it can receive