Database Support for Recommender Systems

Mohamed F. Mokbel

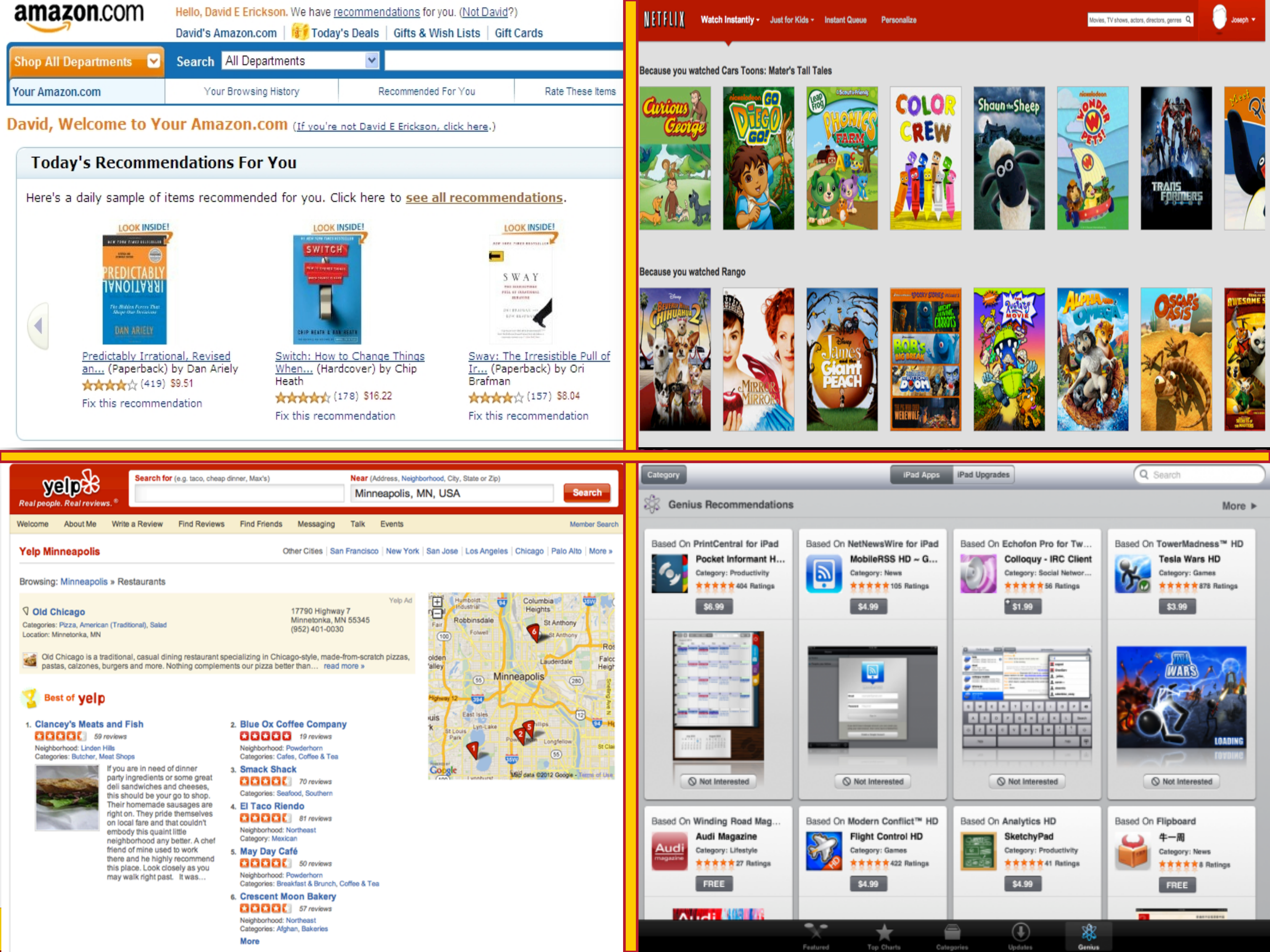
Department of Computer Science and Engineering,
University of Minnesota
www.cs.umn.edu/~mokbel
mokbel@cs.umn.edu

Talk Outline

- **Background on Recommender Systems**
- **DBMS for Recommender Systems**
- **RecBench:** *A Benchmark for Recommender System Architectures*
- **RecStore:** *A Storage Engine Support for Recommender Systems*
- **LARS:** *A Location-Aware Recommender System*
- **Recathon:** *A Context-Aware Recommender System*
- **Summary, Commercial Ads, and Acknowledgments**

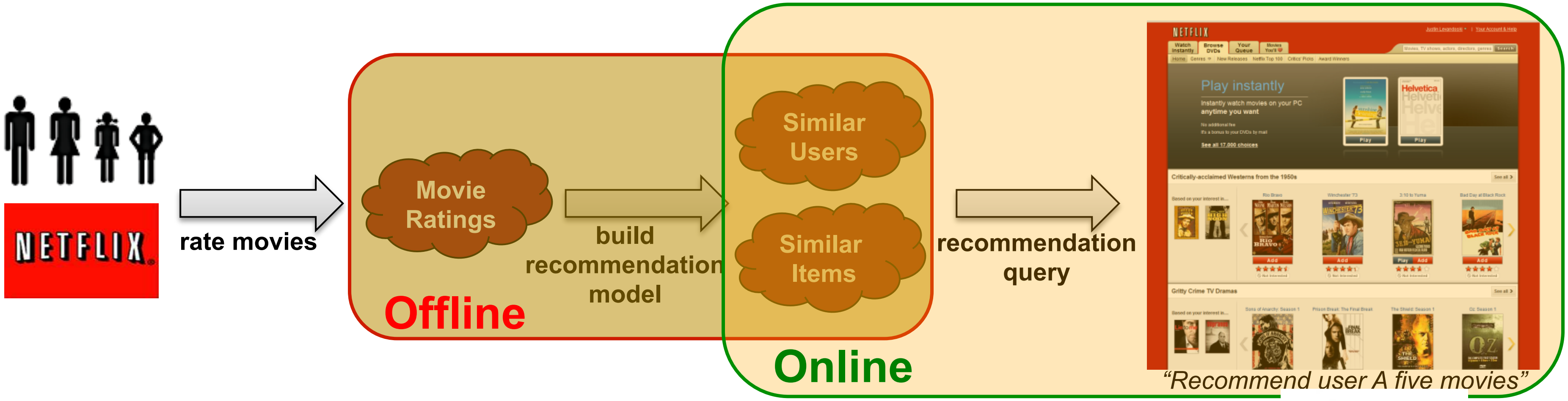
Talk Outline

- **Background on Recommender Systems**
- **DBMS for Recommender Systems**
- **RecBench:** *A Benchmark for Recommender System Architectures*
- **RecStore:** *A Storage Engine Support for Recommender Systems*
- **LARS:** *A Location-Aware Recommender System*
- **Recathon:** *A Context-Aware Recommender System*
- **Summary, Commercial Ads, and Acknowledgments**


amazon.com
Hello, David E Erickson. We have recommendations for you. (Not David?)
David's Amazon.com
Today's Deals
Gifts & Wish Lists
Gift Cards
Shop All Departments
Search
All Departments
Your Amazon.com
Your Browsing History
Recommended For You
Rate These Items
David, Welcome to Your Amazon.com (If you're not David E Erickson, click here.)
Today's Recommendations For You
Here's a daily sample of items recommended for you. Click here to see all recommendations.
Predictably Irrational, Revised an... (Paperback) by Dan Ariely
(419) $9.51
Fix this recommendation
Switch: How to Change Things When... (Hardcover) by Chip Heath
(178) $16.22
Fix this recommendation
Sway: The Irresistible Pull of Ir... (Paperback) by Ori Brafman
(157) $8.04
Fix this recommendation
NETFLIX
Watch Instantly
Just for Kids
Instant Queue
Personalize
Because you watched Cars Toons: Mater's Tall Tales
Because you watched Rango
yelp
Real people. Real reviews.
Search for (e.g. taco, cheap dinner, Max's)
Near (Address, Neighborhood, City, State or Zip)
Minneapolis, MN, USA
Search
Welcome
About Me
Write a Review
Find Reviews
Find Friends
Messaging
Talk
Events
Member Search
Yelp Minneapolis
Other Cities
San Francisco
New York
San Jose
Los Angeles
Chicago
Palo Alto
More »
Browsing: Minneapolis » Restaurants
Old Chicago
Categories: Pizza, American (Traditional), Salad
Location: Minnetonka, MN
17790 Highway 7
Minnetonka, MN 55345
(952) 401-0030
Old Chicago is a traditional, casual dining restaurant specializing in Chicago-style, made-from-scratch pizzas, pastas, calzones, burgers and more. Nothing complements our pizza better than... read more »
Best of yelp
1. Clancey's Meats and Fish
59 reviews
Neighborhood: Linden Hills
Categories: Butcher, Meat Shops
If you are in need of dinner party ingredients or some great deli sandwiches and cheeses, this should be your go to shop. Their homemade sausages are right on. They pride themselves on local fare and that couldn't embody this quaint little neighborhood any better. A chef friend of mine used to work there and he highly recommend this place. Look closely as you may walk right past. It was...
2. Blue Ox Coffee Company
19 reviews
Neighborhood: Powderhorn
Categories: Cafes, Coffee & Tea
3. Smack Shack
70 reviews
Categories: Seafood, Southern
4. El Taco Riendo
81 reviews
Neighborhood: Northeast
Category: Mexican
5. May Day Café
50 reviews
Neighborhood: Powderhorn
Categories: Breakfast & Brunch, Coffee & Tea
6. Crescent Moon Bakery
57 reviews
Neighborhood: Northeast
Categories: Afghan, Bakeries
More
Minneapolis
Category
iPad Apps
iPad Upgrades
Search
Genius Recommendations
More
Based On PrintCentral for iPad
Pocket Informant H...
Category: Productivity
404 Ratings
$6.99
Not Interested
Based On NetNewsWire for iPad
MobileRSS HD ~ G...
Category: News
105 Ratings
$4.99
Not Interested
Based On Echofon Pro for Tw...
Colloquy - IRC Client
56 Ratings
$1.99
Not Interested
Based On TowerMadness™ HD
Tesla Wars HD
Category: Games
878 Ratings
$3.99
Not Interested
Based On Winding Road Mag...
Audi Magazine
Category: Lifestyle
27 Ratings
FREE
Based On Modern Conflict™ HD
Flight Control HD
Category: Games
422 Ratings
$4.99
Based On Analytics HD
SketchyPad
Category: Productivity
41 Ratings
$4.99
Based On Flipboard
牛一周
Category: News
8 Ratings
FREE
Featured
Top Charts
Categories
Updates
Genius

Recommender Systems

- Analyze user behavior to recommend personalized and interesting things to do/read/see

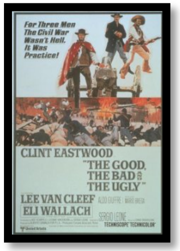


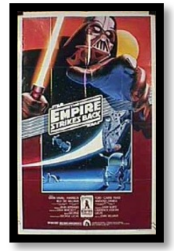
Introducing Ping. A social network for music.
Follow your favorite artists and friends to discover the music they're talking about, listening to, and downloading.

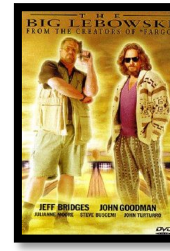
FIND OUT IF WE'RE DELIVERING TO YOUR NEIGHBORHOOD!

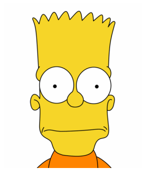
- *Collaborative filtering* process is the most commonly used one in Recommender Systems

Collaborative Filtering (CF)

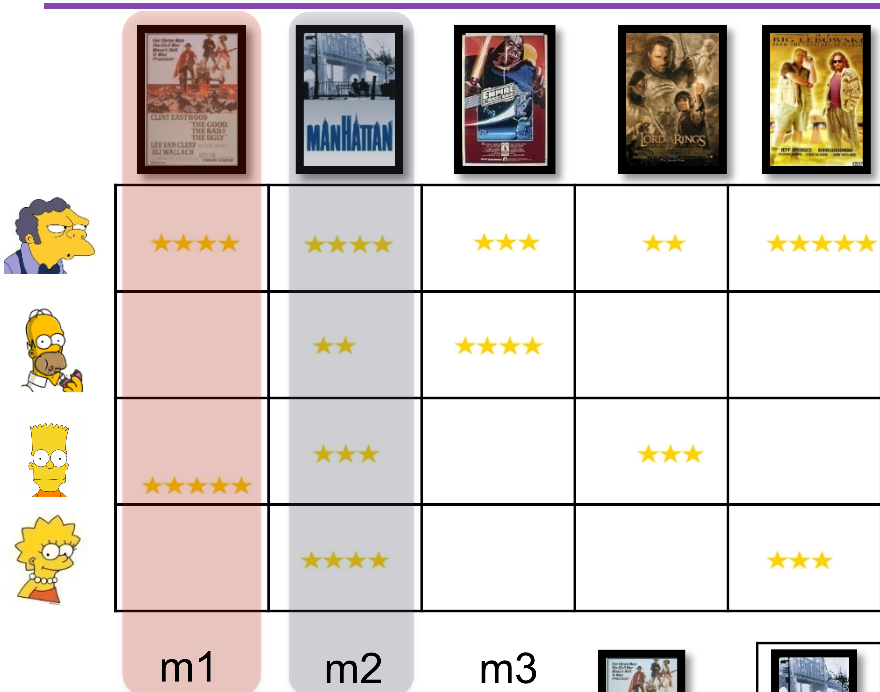






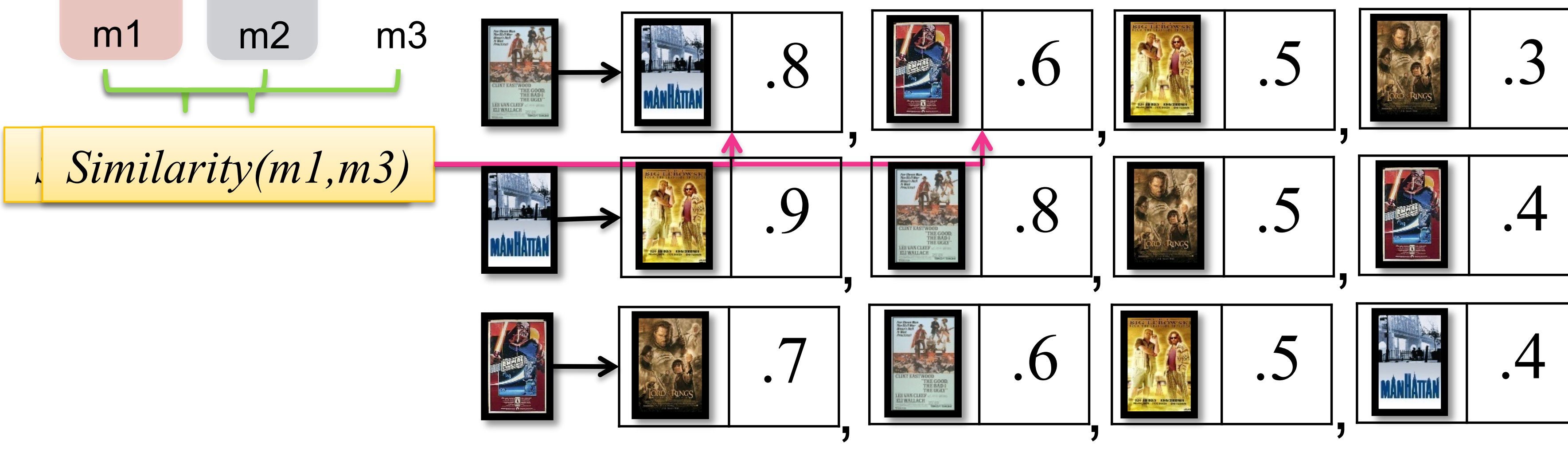
	★★★★	★★★★	★★★	★★	★★★★★
	?	★★	★★★★	?	?
	★★★★★	★★★		★★★	
		★★★★			★★★

Item-Based CF Model Building

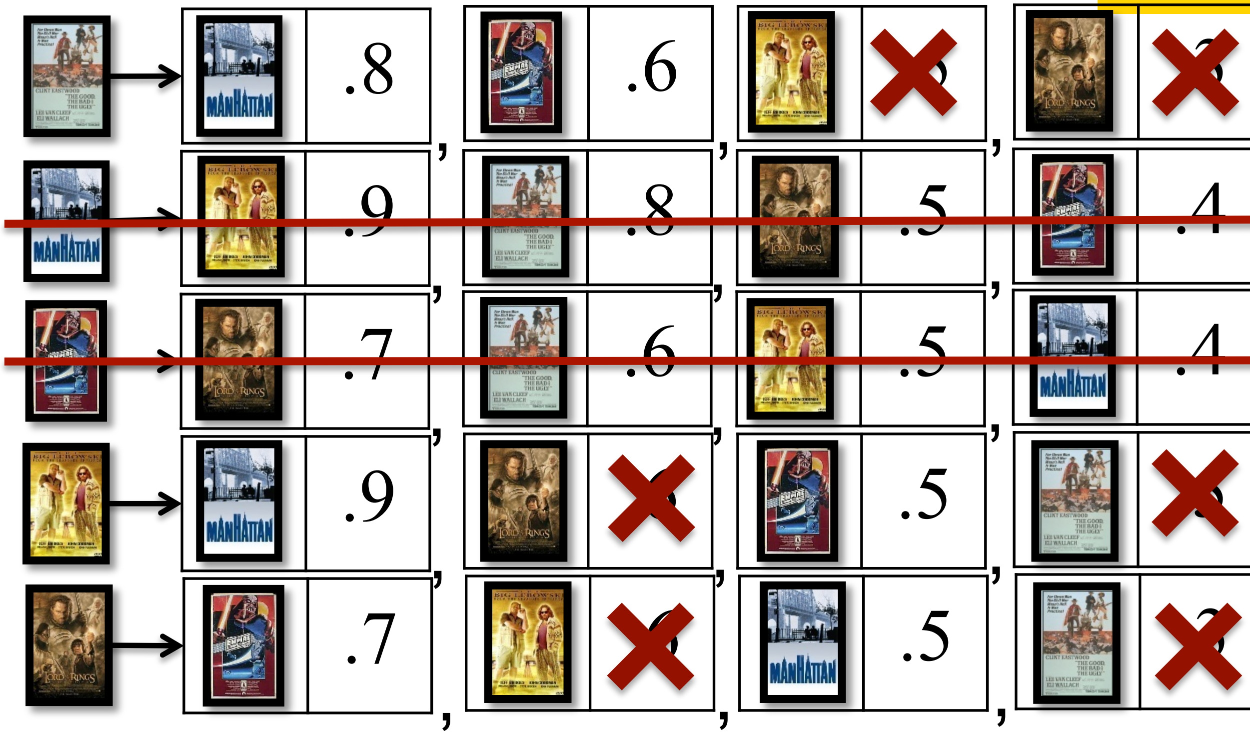


- **Similarity measures**
  1. Cosine distance
  2. Pearson correlation
  3. Spearman correlation
  4. Adjusted cosine distance

  . . .



Item-Based CF Recommendations



(.8 X 2 + .6 X 4) / (.8 + .6) = 2.86

(.9 X 2 + .5 X 4) / (.9 + .5) = 2.71

(.5 X 2 + .7 X 4) / (.7 + .5) = 3.17

Talk Outline

- Background on Recommender Systems
- DBMS for Recommender Systems
- **RecBench:** *A Benchmark for Recommender System Architectures*
- **RecStore:** *A Storage Engine Support for Recommender Systems*
- **LARS:** *A Location-Aware Recommender System*
- **Recathon:** *A Context-Aware Recommender System*
- **Summary, Commercial Ads, and Acknowledgments**

Mohamed F. Mokbel. "DBMS Support for Recommender Systems". In **CIDR 2011**, Gong show, Asilomar, CA, January 2011.

Recommender Systems: Quality vs. Performance

- Recommender systems community have only focused on "quality" issues; "performance" is considered a secondary issue

"*We have chosen not to discuss computation performance of recommender algorithms*. Such performance is certainly important, and in the future we expect there to be work on the quality of time-limited and memory-limited recommendations."

Herlocker et al. "Evaluating Collaborative Filtering Recommender Systems", ACM TOIS 2004

"[Our] solution is based on a huge amount of models and predictors *which would not be practical as part of a commercial recommender system*. However, this result is a direct consequence of the nature and goal of the competition: *obtain the highest possible accuracy at any cost, disregarding completely the complexity of the solution and the execution performance*."

Team BelKor's Pragmatic Chaos, Winner of the 2009 Netflix Prize

- All heavy work are done offline
- Models are built over long time period, e.g., movie or books ratings
- The rank of one item in the system slowly change

Things have changed…

- **We live in an increasingly social and "real-time" world**
  - Number of things to recommend is growing exponentially
  - Users expressing opinions faster than ever
  - Recommendations change second-to-second

Facebook Posts

Blog/News Items

- **"Offline" step can <u>no longer</u> be tolerated**

Recommender Systems in DBMS ?

- **Incoming stream of ratings data: (*user*, *item*, *rating*)**
- **Ratings are used to build a recommendation model as:**
  - Item-based collaborative filtering: (*item*, *item*, *similarity*)
  - User-based collaborative filtering: (*user*, *user*, *similarity*)
- **Recommendation query:**
  - Item-based collaborative filtering:
    - *Given a user u, find the top-k items that are most similar to the items that u has liked before*
  - User-based collaborative filtering:
    - *Given a user u, find the top-k items that the users who are similar to u have liked*

Recommender Systems have all the ingredients of a data management problem

DBMS Challenge

Lets not try to find a new way of doing recommendation*

* ACM RecSys community is already doing excellent job in this frontier. Lets start from there.

We need to provide online support and scale up the computations of existing recommender methods.

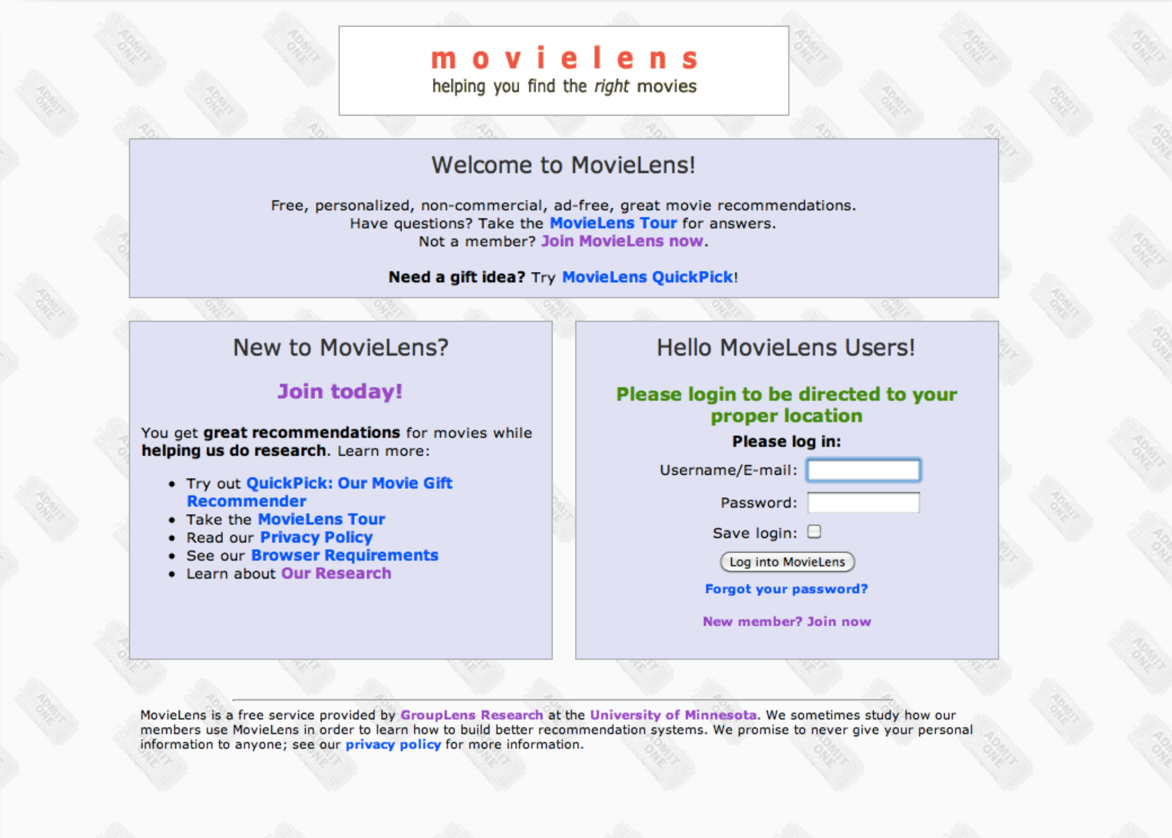
Can DBMS do it ?

Talk Outline

- Background on Recommender Systems
- DBMS for Recommender Systems
- **RecBench:** *A Benchmark for Recommender System Architectures*
- **RecStore:** *A Storage Engine Support for Recommender Systems*
- **LARS:** *A Location-Aware Recommender System*
- **Recathon:** *A Context-Aware Recommender System*
- **Summary, Commercial Ads, and Acknowledgments**

Justin J. Levandoski, Michael D. Ekstrand, Michael J. Ludwig, Ahmed Eldawy, Mohamed F. Mokbel and John T. Riedl. " **RecBench**: Benchmarks for Evaluating Performance of Recommender System Architectures". In Proceedings of the International Conference on Very Lage Databases, **VLDB 2011**, Seattle, WA, September 2011

MovieLens



http://www.movielens.org

- **MovieLens**: A Movie Recommender System, built and maintained at University of Minnesota (GroupLens Research)
  - 10 Million ratings
  - 10,000 Movies
  - 72,000 Users

ACM Software Award 2010: – GroupLens Collaborative Filtering Recommender Systems: Peter Bergstrom, Lee R Gordon, Jonathan L Herlocker, Neophytos Iacovou, Joseph A Konstan, Shyong (Tony) K. Lam, David Maltz, Sean McNee, Bradley N Miller, Paul J Resnick, John T Riedl, Mitesh Suchak

RecBench:
A Benchmark for Recommender System Architectures

- **Goals:**
  ① Prompt DB & RecSys research communities to work together
  ② A benchmark to test performance of different system architectures
- **Six common recommendation tasks are carefully selected**
- **Every task is implemented on three different architectures**

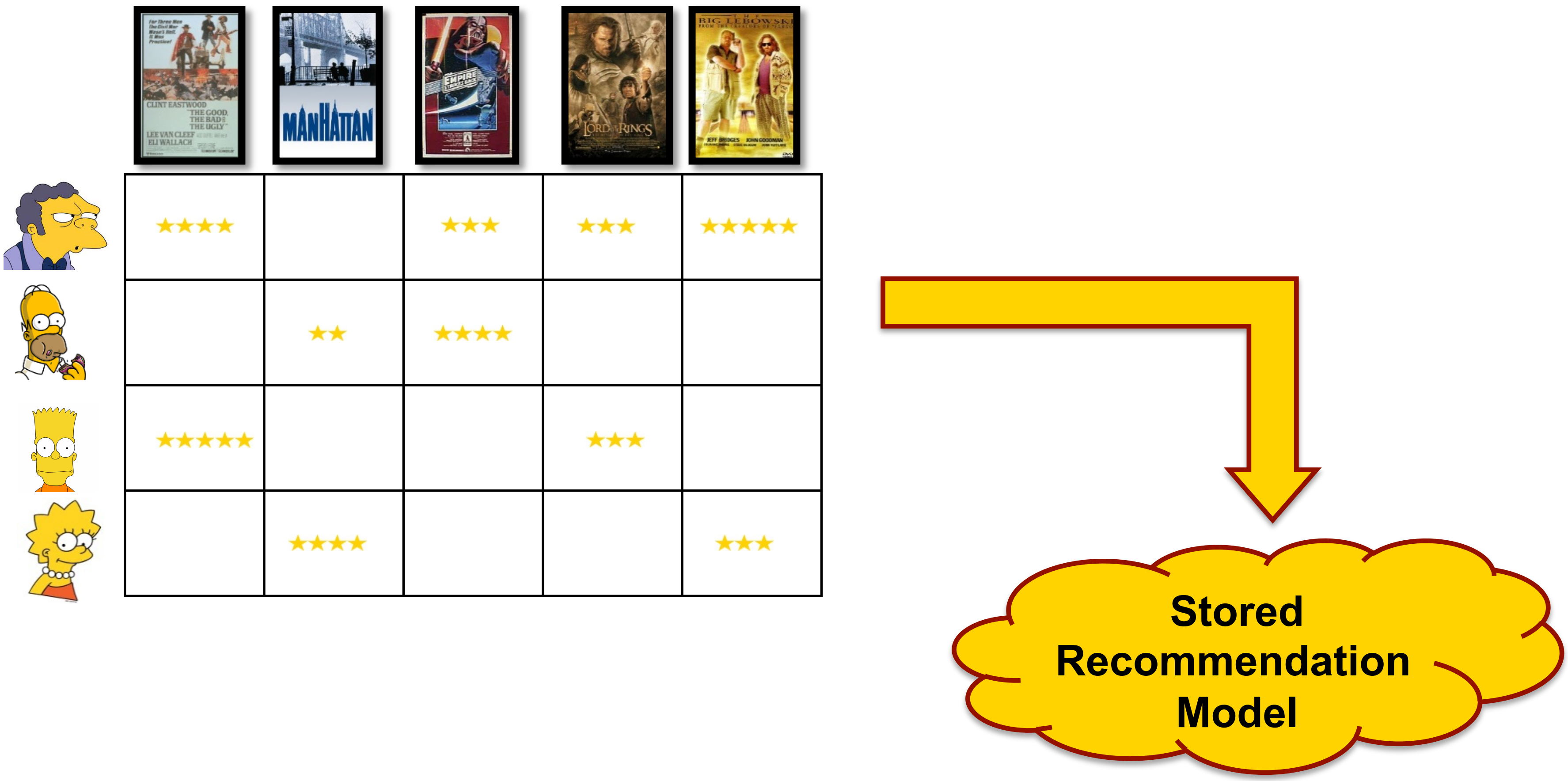
MultiLens
- "Hand-built" system
- Code optimized for item-based CF
- Uses DBMS for metadata and text-search queries

PostgreSQL
- Unmodified DBMS
- Ratings relation: ratings(usr,itm,rating)
- Model relation: model(itm, itm, sim)
- All tasks implemented in standard SQL

Custom DBMS
- DBMS (PostgreSQL) modified to optimize for fast recommender model updates
- SQL same as unmodified DBMS approach

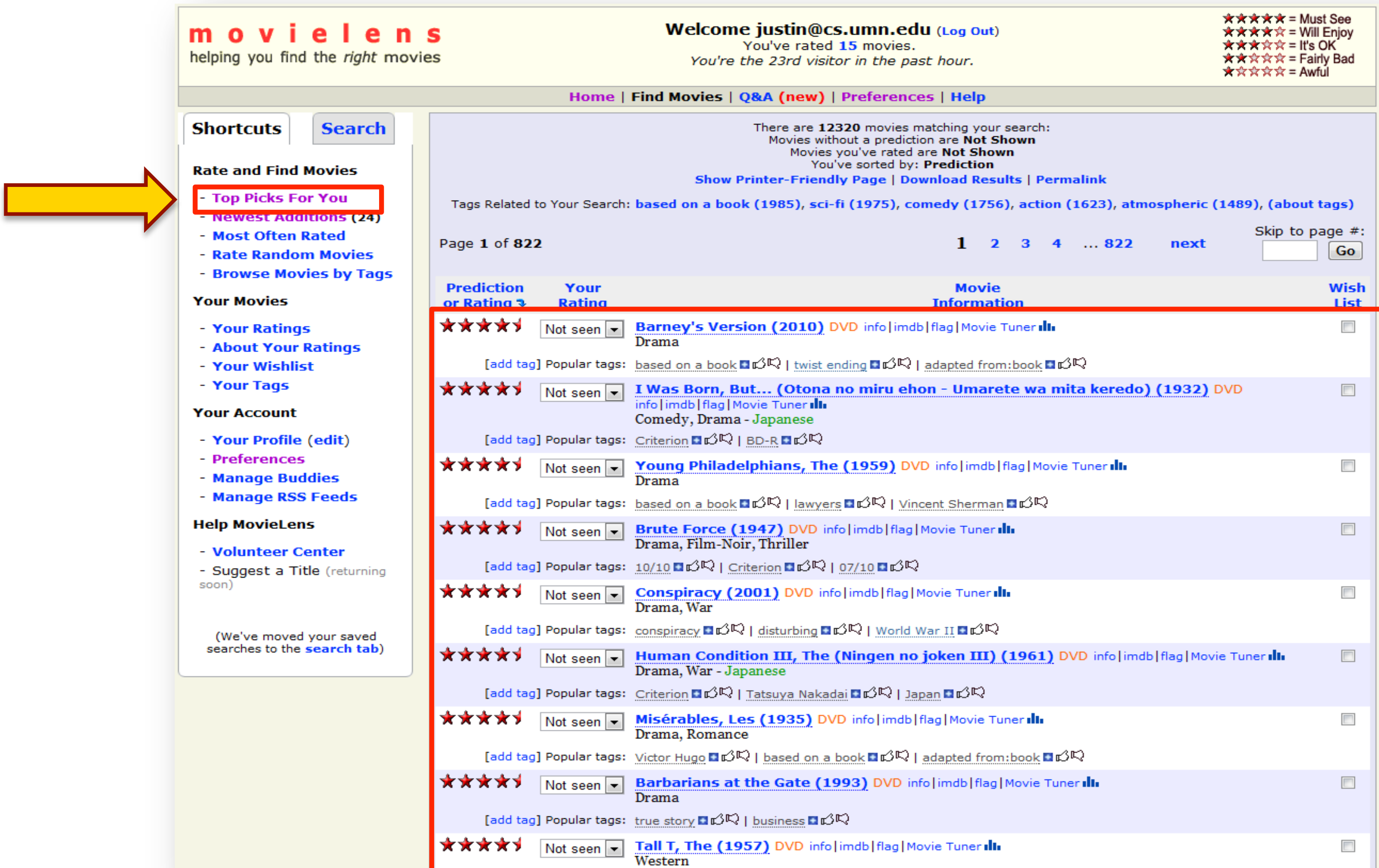
RecBench – Task 1: Initialization

Prepare system to start serving user recommendations



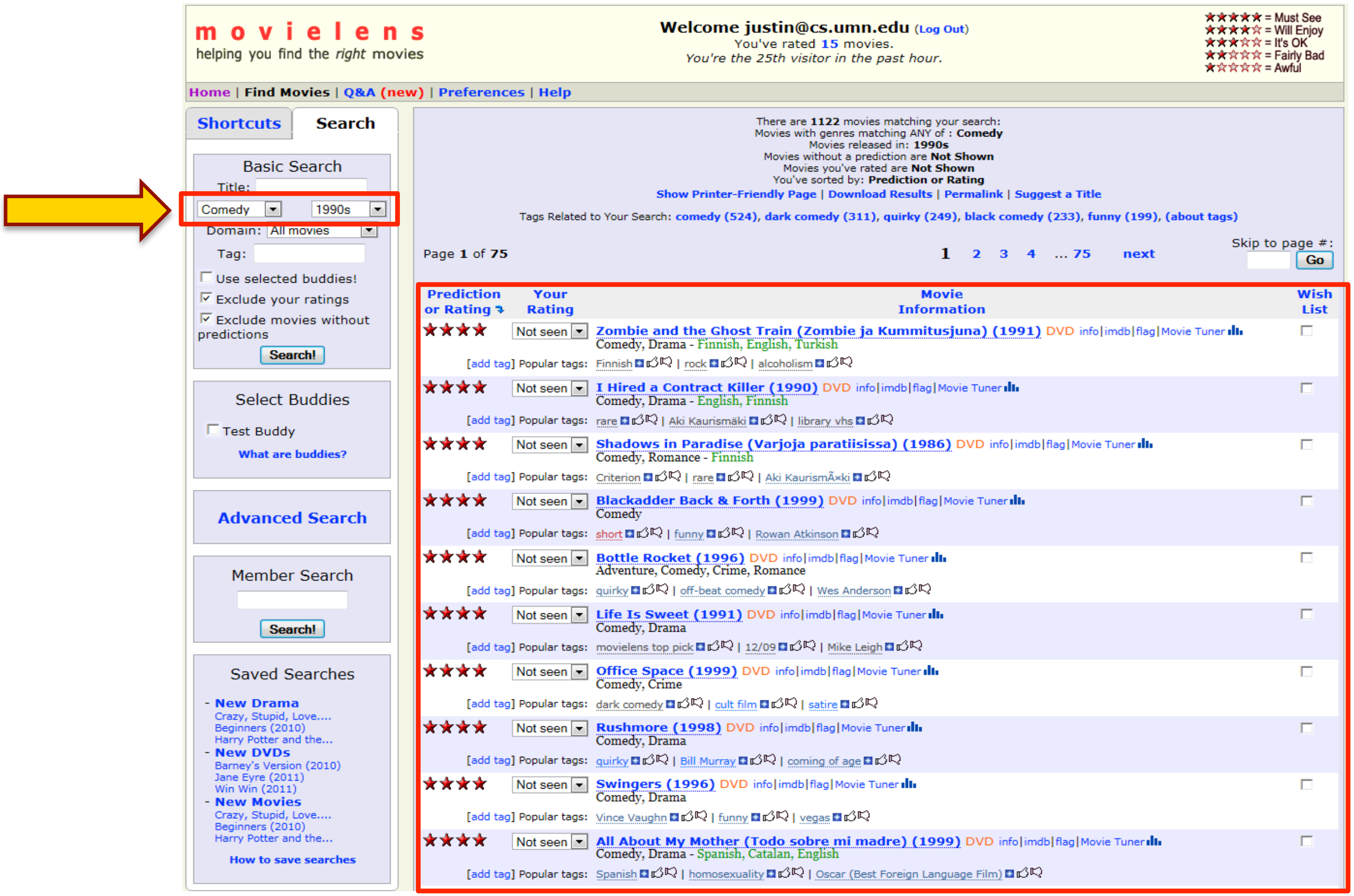
RecBench – Task 2: Pure Recommend

Produce top-k recommendations from system's entire item pool



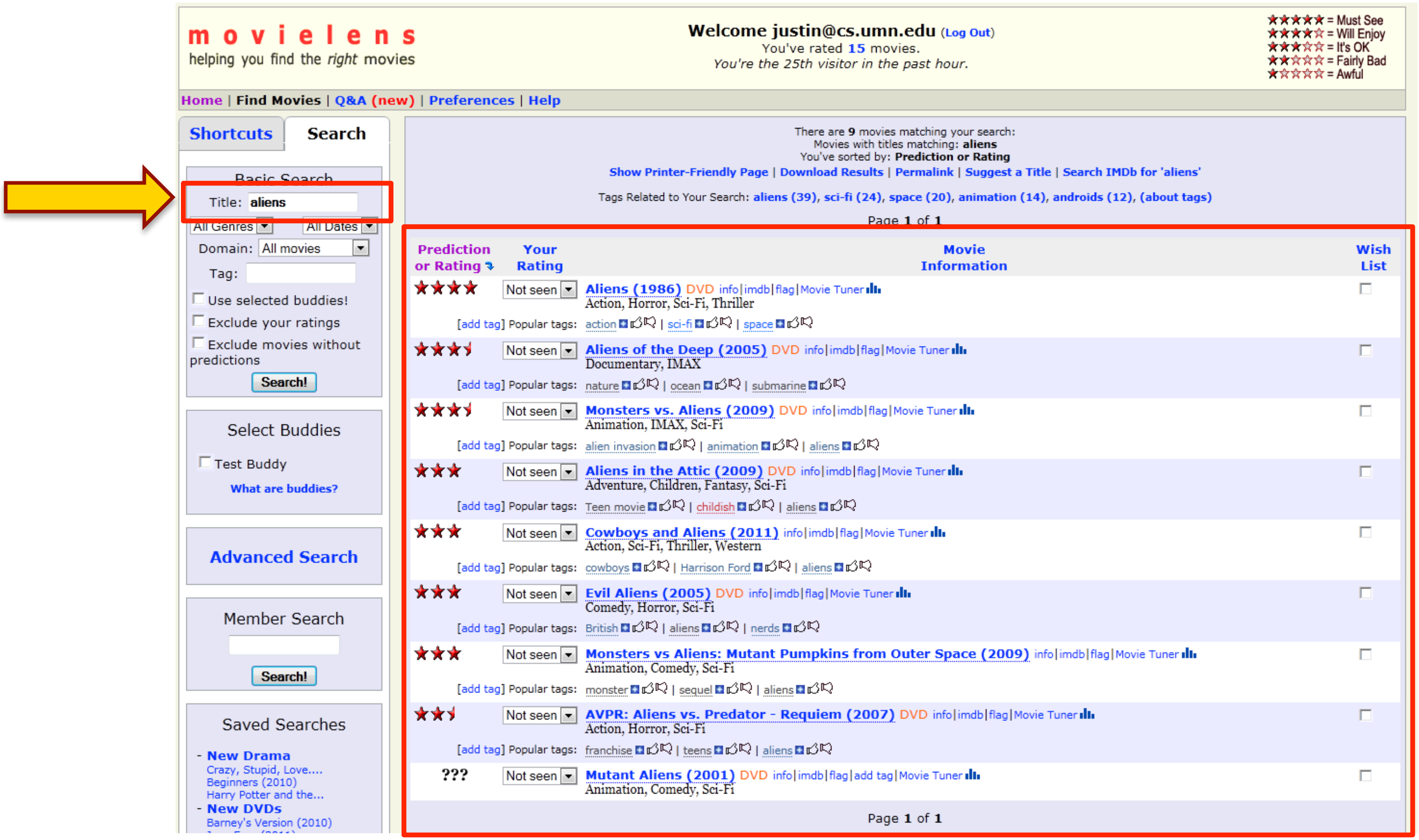
RecBench – Task 3: Filtered Recommend

Produce top-k recommendations that match item constraints



RecBench – Task 4: Blended Recommend

Produce top-k recommendations based on blended text-search and recommendation score



RecBench – Task 5: Item Prediction

Generate a user's predicted rating for a target item

RecBench – Task 6: New Item(s)

Incorporate new item(s) into the system for recommendation

RecBench: Summary of Results

- **Datasets**
  - MovieLens: 10 M movie ratings, 10K movies, and 72K users
  - Netflix Challegne: 100 M movie ratings, 18K movies, and 480K users
- **Tasks 1 & 2 (Initialization & Pure Recommend)**
  - PostgresSQL has by far the worst performance
  - Custom DBMS does a good job but not as excellent as MultiLens
- **Tasks 3 & 4 (Filtered & Blended Recommend)**
  - CustomDBMS way outperforms Multilens as it takes advantage of its select & top-k operators
  - PostgresSQL performance in the middle.
- **Tasks 5 & 6 (Item Prediction & New Items)**
  - MultiLens outperforms others in Task 5 (a basic component in Task 1)
  - CustomDBMS outperforms others in Task 6 due to the built-in incrementally maintained statistics

Talk Outline

- **Background on Recommender Systems**
- **DBMS for Recommender Systems**
- **RecBench:** *A Benchmark for Recommender System Architectures*
- **RecStore:** *A Storage Engine Support for Recommender Systems*
- **LARS:** *A Context-Aware Recommender System*
- **Recathon:** *A Context-Aware Recommender System*
- **Summary, Commercial Ads, and Acknowledgments**

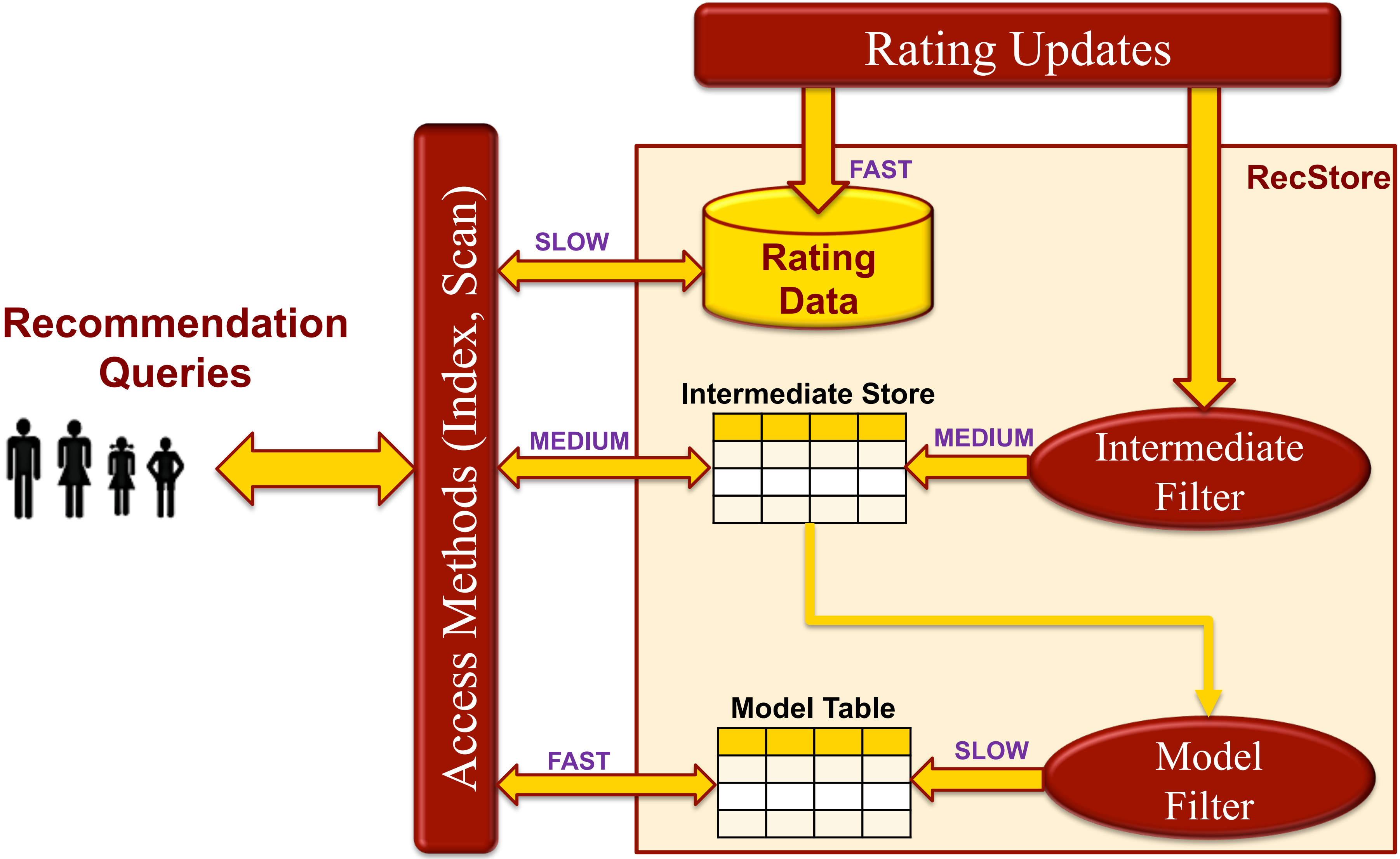
Justin J. Levandoski, Mohamed Sarwat, Mohamed F. Mokbel, and Michael D. Ekstrand. " **RecStore**: An Extensible and Adaptive Framework for Online Recommender Queries inside the Database Engine". In Proceedings of the International Conference on Extending Database Technology, **EDBT 2012**, Berlin, Germany, March 2012

RecStore:
A Storage Engine Support for Recommender Systems

RecStore pushes the recommender model building inside the Database Engine to provide online support and scale up the computations of existing recommender methods.

- **Adaptivity of RecStore**
  - RecStore is adaptive to different system workloads (Query Intensive vs. Update Intensive)
- **Extensibility of RecStore**
  - RecStore is extensible to support many recommendation methods (e.g., item-based CF, user-based CF).

RecStore: Architecture



RecStore: Query Latency vs. Maintenance Cost

Materialize-All

-Low Latency Recommendation Query.
-High Storage and maintenance Cost.

Materialize-None

-High Latency Recommendation Query
-Low Storage and maintenance Cost.

Intermediate Store Only

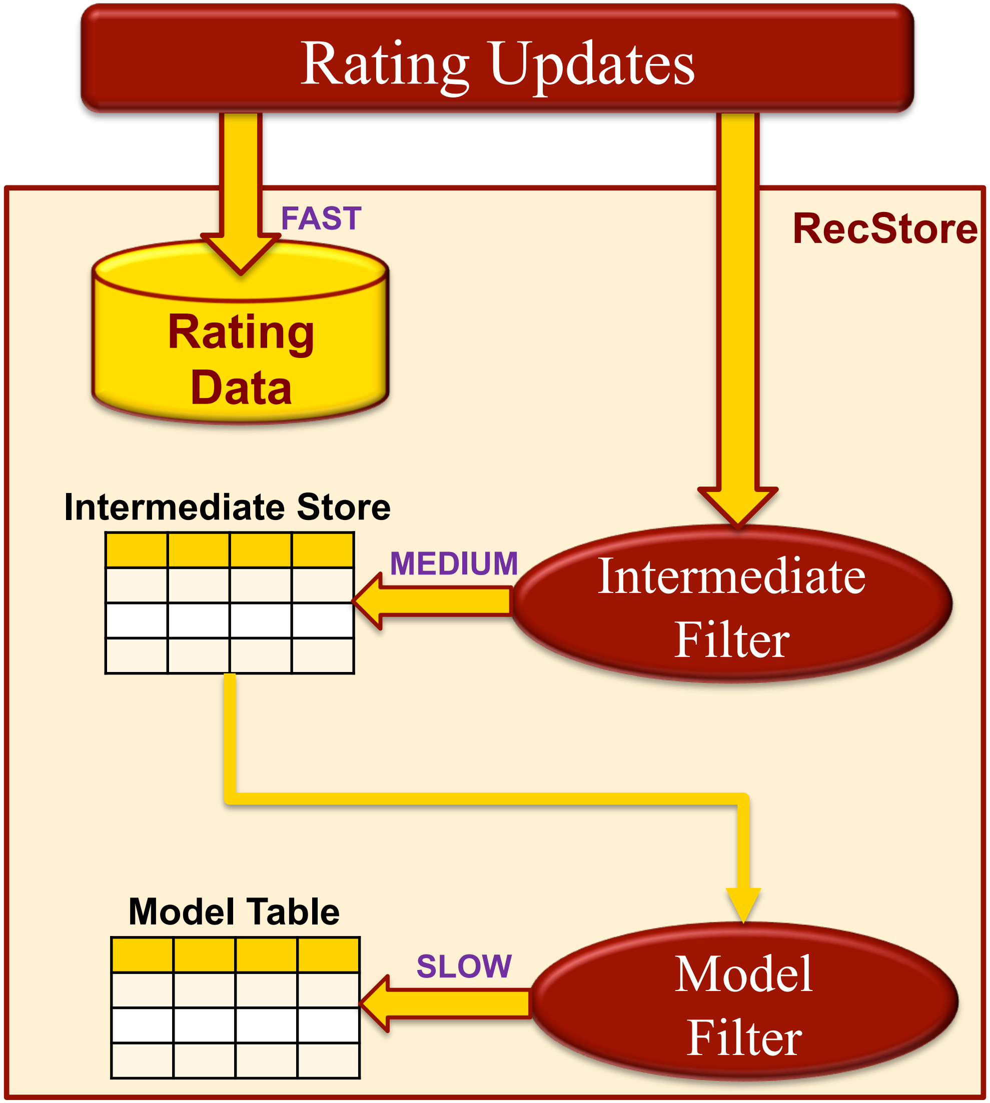
-Middle Ground between Materialize-All and Materialize-None

Intermediate Store / Partial Model Store

-Middle Ground between Materialize-All and Intermediate-Only

Partial Intermediate Store / Partial Model Store

-Lies between Partial Model and Intermediate Only



RecStore: Defining a Recommender Model

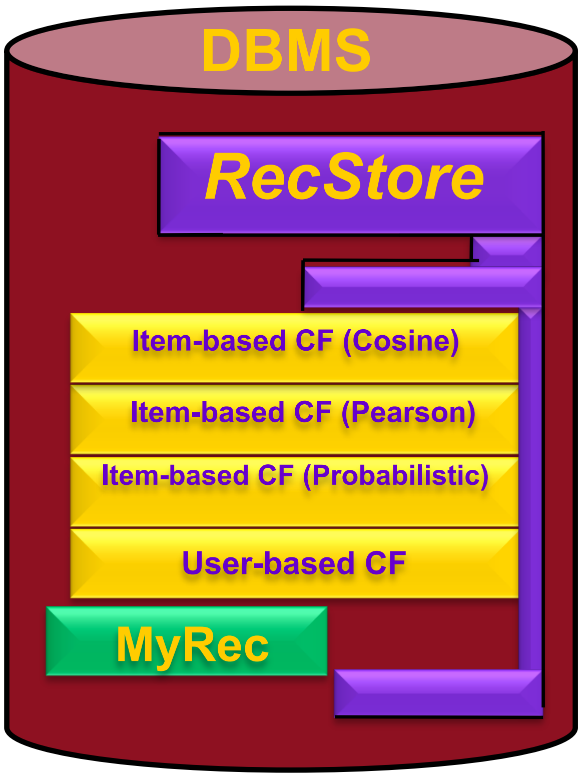
```
DEFINE RECSTORE MODEL ItemItemCosine
FROM        Ratings R1, Ratings R2
WHERE       R1.ItemId <> R2.itemId AND R1.userId = R2.userId

WITH INTERMEDIATE STORE:
            (R1.itemID as item, R2.itemId as rel_itm,
            vector_lenp, vector_lenq, dot_prod, co_rate)

WITH INTERMEDIATE FILTER:
       ALLOW UPDATE WITH My_IntFilterLogic(),
       UPDATE vector_lenp AS vector_lenp + R1.rating * R1.rating,
       UPDATE vector_lenq AS vector_lenp + R2.rating * R2.rating,
       UPDATE dot_prod AS ot_prod + R1.rating * R2.rating,
       UPDATE co_rate AS 1

WITH MODEL STORE:
            (R1.itemId as item, R2.itemId as rel_itm, COMPUTED sim)

WITH MODEL FILTER:
       ALLOW UPDATE WITH My_ModFilterLogic(),
       UPDATE sim AS if (co_rate < 50)
              co_rate * dot_prod / ( 50*sqrt(vector_lenp)
                                                 * sqrt (vector_lenq));
                       else
                         co_rate / sqrt(vector_lenp) *
                                                 sqrt(vector_lenp);
```



Simple SQL to Plug-in a new Recommendation Method

RecStore: Querying a Recommender Model

User/item ratings

uid	mid	rating
1	1	3.5
2	1	5
…	…	…

Model

itm	rel_itm	Prediction
1	2	4.0
…	…	…
…	…	…

```
/* Find movies rated by REC_USER_X,
 * store in temp table usrXMovies */
CREATE TEMP TABLE usrXMovies AS
SELECT  R.mid as itemId, R.rating as rating
FROM    ratings R
WHERE   R.uid = REC_USER_X;

/* Generate predictions using weighted sum */
SELECT  M.itm as Candidate Item,
        SUM(M.sim * U.rating)/ SUM(M.sim) as Prediction
FROM    Model M  usrXMovies U
WHERE   M.rel_itm = U.itmId AND
        M.itm NOT IN (select itmId FROM usrXMovies)
GROUP BY M.itm ORDER BY Prediction DESC;
```

Maintain the recommendation *Model* to efficiently answer recommendation queries

Talk Outline

- **Background on Recommender Systems**
- **DBMS for Recommender Systems**
- **RecBench:** *A Benchmark for Recommender System Architectures*
- **RecStore:** *A Storage Engine Support for Recommender Systems*
- **LARS:** *A Location-Aware Recommender System*
- **Recathon:** *A Context-Aware Recommender System*
- **Summary, Commercial Ads, and Acknowledgments**

Mohamed Sarwat, Justin J. Levandoski, Ahmed Eldawy, and Mohamed F. Mokbel. " **LARS*:** A Scalable and Efficient Location-Aware Recommender System". IEEE Transactions on Knowledge and Data Engineering, **TKDE 2013**, *To Appear*.

Justin J. Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F. Mokbel. " **LARS**: A Location-Aware Recommender System". In Proceedings of the IEEE International Conference on Data Engineering, **ICDE 2012**, Washington D.C., April 2012

"Locations" and "Recommendations"

- **Recommender systems rely on the input triple :**

(user, item, rating)

  - Recommender systems completely ignore the spatial aspects of both users and items

Do we need to consider "Locations" ?

- **Adding Location-awareness to Recommender Systems**
  - Recommend movies based on the locations of the ratings
- **Adding Recommendation-awareness to Location-based services**
  - Instead of asking about restaurants in a certain area or closest to me, I can ask a recommender system to suggest few restaurants to me

Location Matters: Netflix Rental Patterns

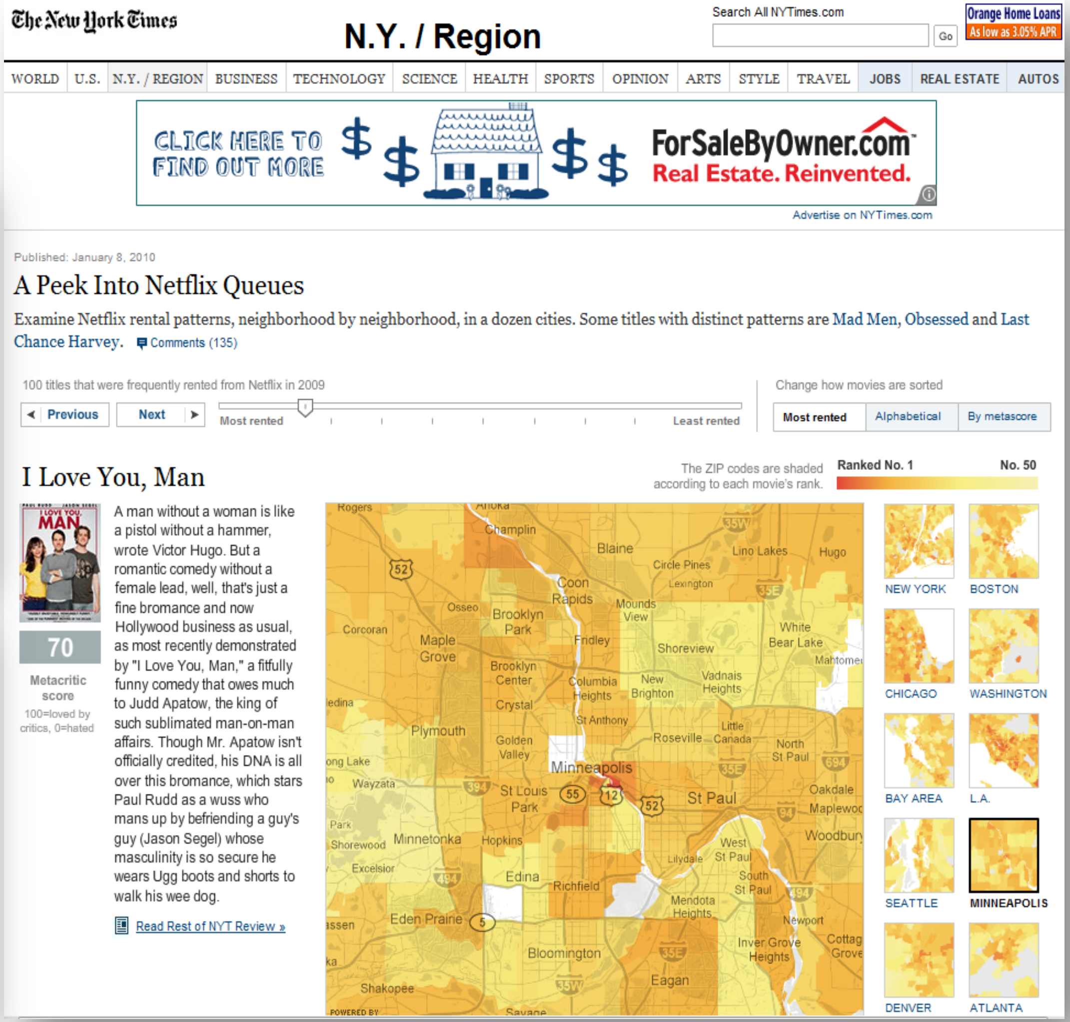
- Movie preferences differ based on the user location (zip code)

Most rented in 55418
1. Milk
2. The Curious Case of Benjamin Button
3. Burn After Reading
4. The Wrestler
5. Slumdog Millionaire
6. Gran Torino
7. Doubt
8. Changeling
9. Rachel Getting Married
10. Twilight

...

16. I Love You, Man

Most rented in 55455
1. I Love You, Man
2. Slumdog Millionaire
3. Adventureland
4. My Best Friend's Girl
5. Nick and Norah's Infinite Playlist
6. Sunshine Cleaning
7. Forgetting Sarah Marshall
8. Away We Go
9. Role Models
10. Confessions of a Shopaholic



Most rented in 55113
1. The Curious Case of Benjamin Button
2. Slumdog Millionaire
3. Gran Torino
4. Doubt
5. Milk
6. Seven Pounds
7. Burn After Reading
8. Changeling
9. The Wrestler
10. New in Town

...

24. I Love You, Man

Most rented in 55404
1. Burn After Reading
2. Milk
3. The Curious Case of Benjamin Button
4. Slumdog Millionaire
5. The Wrestler
6. Twilight
7. Doubt
8. Rachel Getting Married
9. Changeling
10. Gran Torino

...

12. I Love You, Man

Location Matters: Top-3 Check-In Destinations in Foursquare

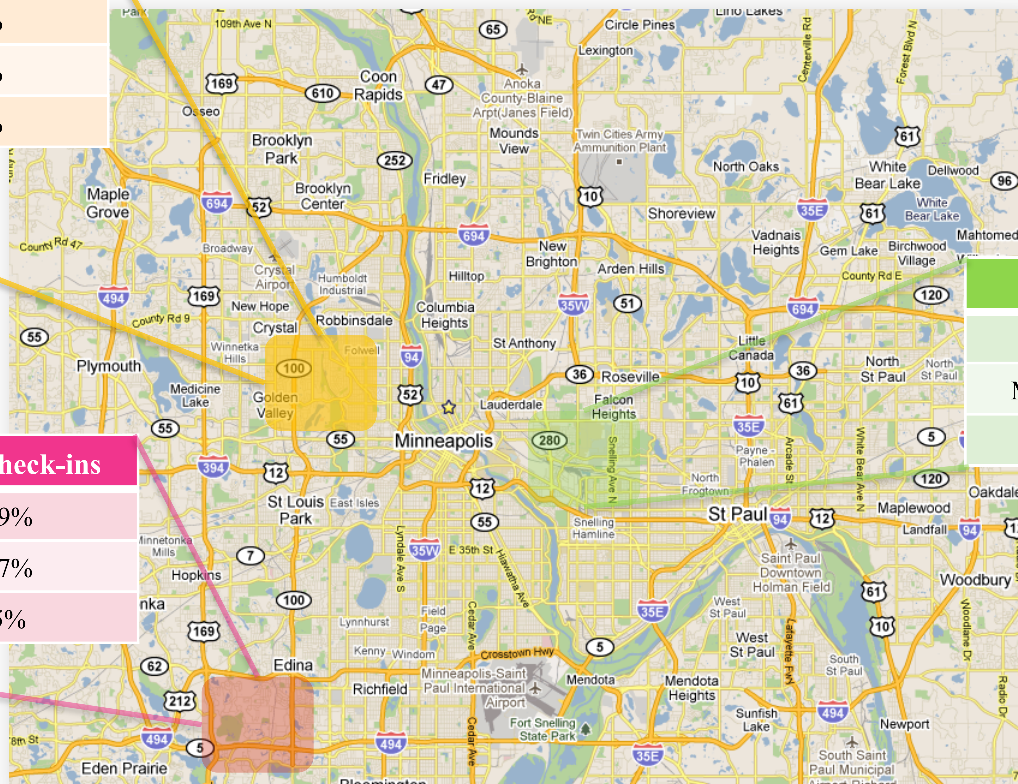
- **Destination preferences differ based on the user location (zip code) and the destination location**

Fousquare users from Robbinsdale tend to visit venues in …

City	% of check-ins
Brooklyn Park	32%
Robbinsdale	20%
Minneapolis	15%

Fousquare users from Edina tend to visit venues in …

City	% of check-ins
Edina	59%
Minneapolis	37%
Edin Prarie	5%



Fousquare users from Falcon Heights tend to visit venues in …

City	% of check-ins
St. Paul	17%
Minneapolis	13
Roseville	10%

LARS: A Location-Aware Recommender System

- **We need to go beyond the traditional rating triple *(user, item, rating)* to include the following taxonomy:**

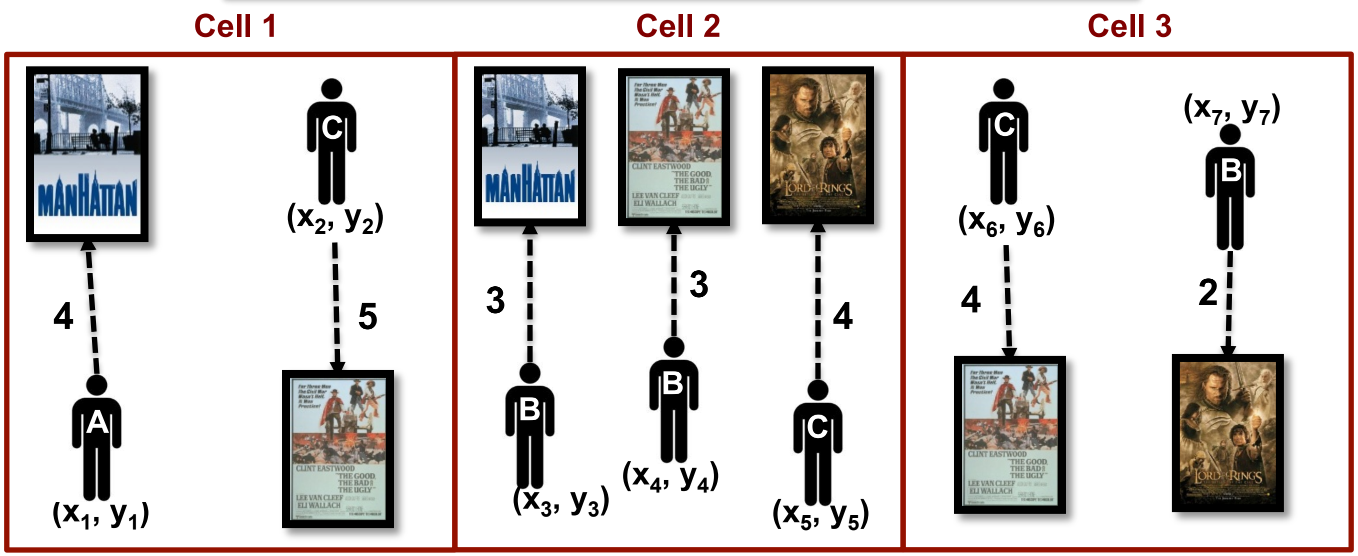
① ***Spatial Rating for Non-spatial Items***
- ***(user_location, user, item, rating)***
- ***Example:*** A user with a certain location is rating a movie
- ***Recommendation:*** Recommend a movie that neighbor users have liked

② ***Non-spatial Rating for Spatial Items***
- ***(user, item_location, item, rating)***
- ***Example:*** A user with unknown location is rating a restaurant
- ***Recommendation:*** Recommend a restaurant within a close vicinity

③ ***Spatial Rating for Spatial Items***
- ***(user_location, location, item_location, item, rating)***
- ***Example:*** A user with a certain location is rating a restaurant

Spatial User Ratings For Non-Spatial Items

1. Partition ratings by user location



2. Build collaborative filtering model for each cell using only ratings contained within the cell

Cell 1

Build Collaborative Filtering Model using:

User	Item	Rating
A		4
C		5

Cell 2

Build Collaborative Filtering Model using:

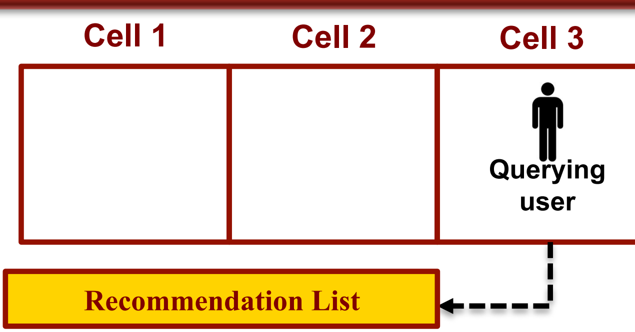
User	Item	Rating
B		3
B		3
C		4

Cell 3

Build Collaborative Filtering Model using:

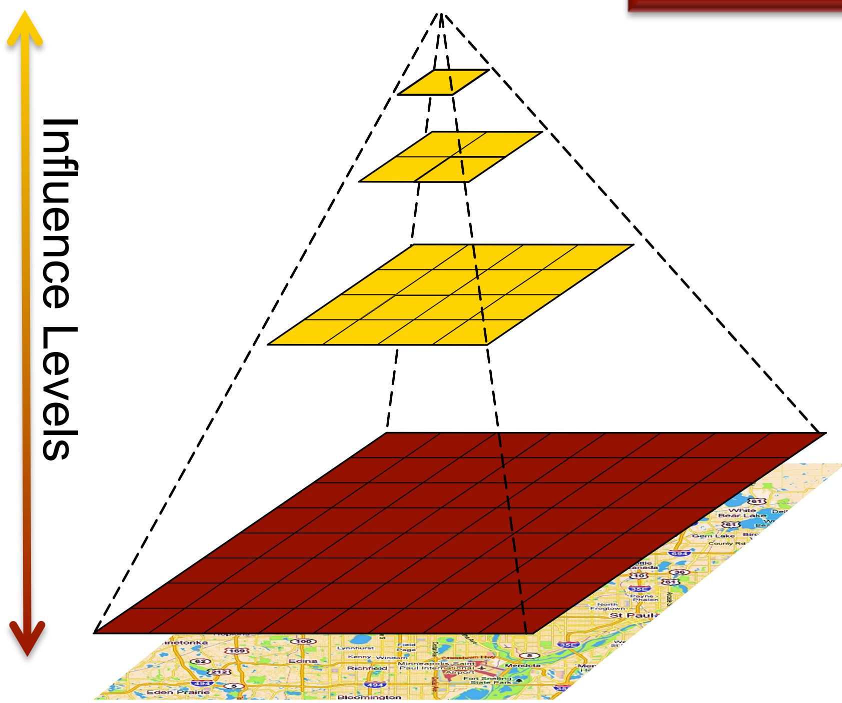
User	Item	Rating
B		4
C		5

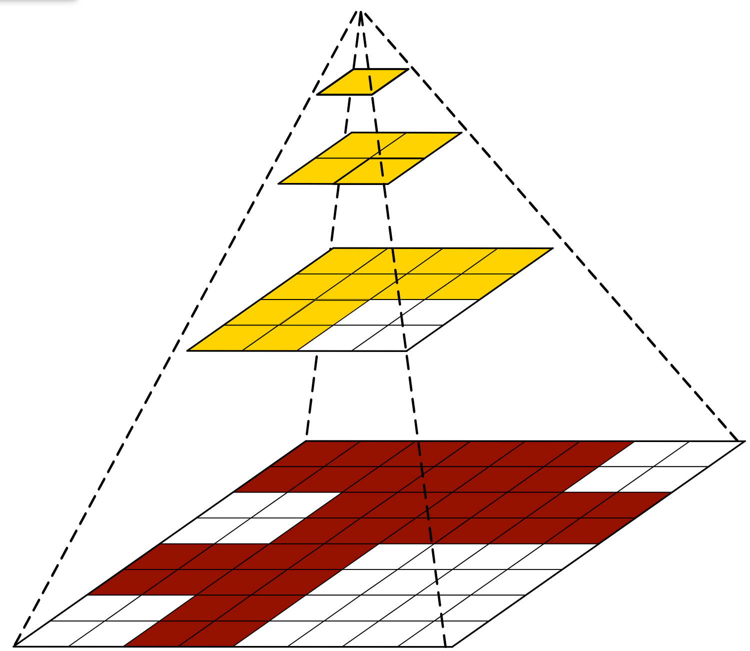
3. Generate recommendations using collaborative filtering using the model of the cell containing querying user



Spatial User Ratings For Non-Spatial Items

User Partitioning





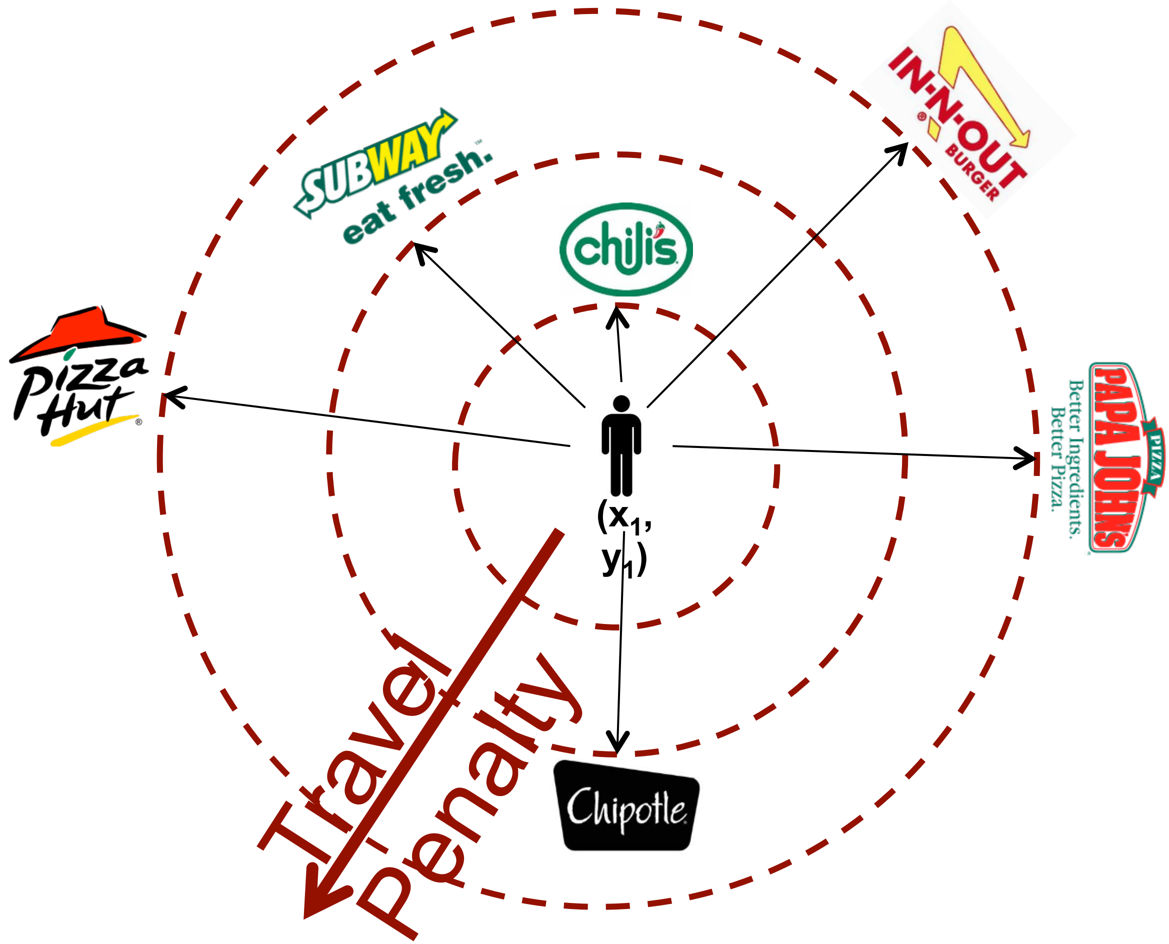
- Smaller cells ➔ More "localized" answer
- Each user can select a personalized localized level
- Scalability problem in terms of maintaining large numbers of recommendation models

- No need to maintain all cells
- If four cells will end up giving the same recommendations, merge them.
- If ratings inside a cell are diverse, split it
- Merging and splitting balance between localization and storage/maintenance

Non-Spatial User Ratings For Spatial Items

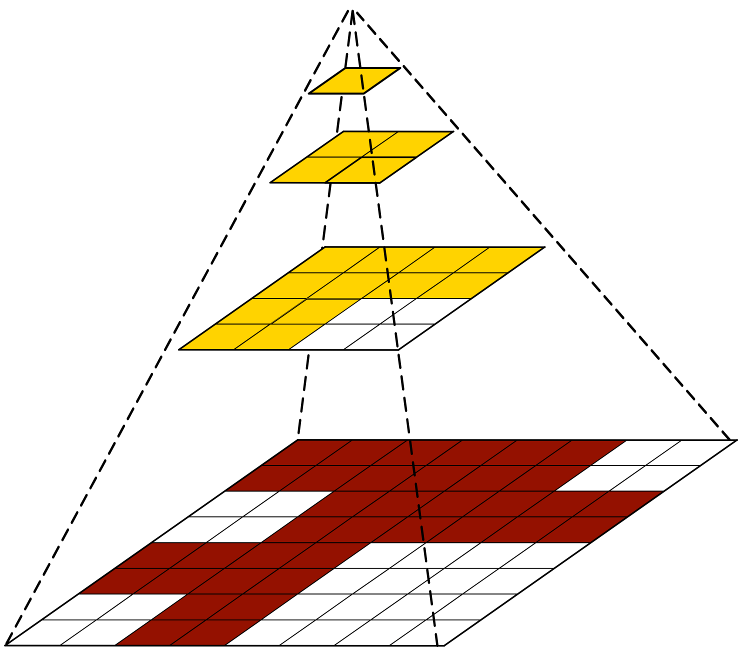
Travel Penalty

- Penalize items based on their distance from the user.
- Distance from the user is normalized to the ratings scale to get the Travel Penalty.
- Use a ranking function that combines the recommendation score and travel penalty
- Incrementally, retrieve items based on travel penalty, and calculate the ranking score on an ad-hoc basis
- Employ an early stopping condition to minimize the list of accessed items to get the K recommended items



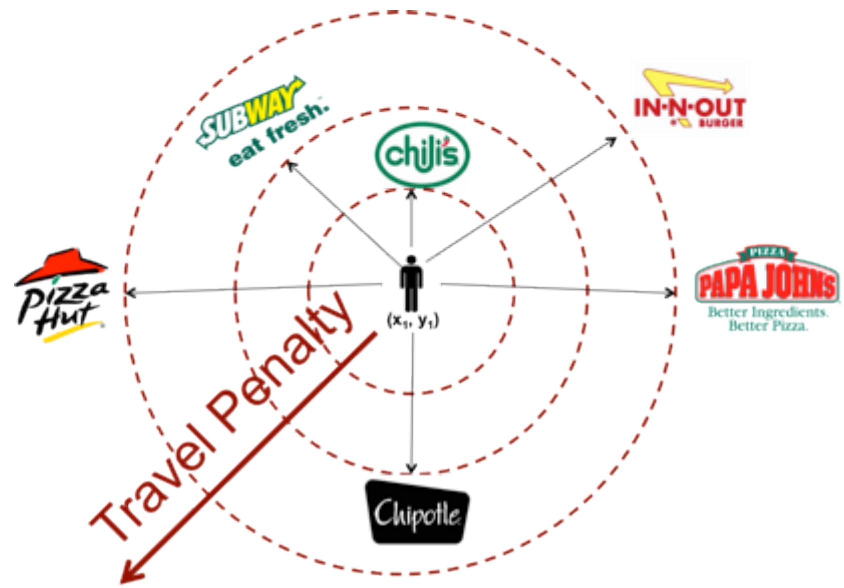
Spatial User Ratings For Spatial Items

User Partitioning



\+

Travel Penalty



Talk Outline

- **Background on Recommender Systems**
- **DBMS for Recommender Systems**
- **RecBench:** *A Benchmark for Recommender System Architectures*
- **RecStore:** *A Storage Engine Support for Recommender Systems*
- **LARS:** *A Location-Aware Recommender System*
- **Recathon:** *A Context-Aware Recommender System*
- **Summary, Commercial Ads, and Acknowledgments**

Mohamed Sarwat, James Avery, and Mohamed F. Mokbel. " **Recathon**: A Unified Architecture for Personalized Recommendation in Database Systems". Under Preperation.

Context-Aware Recommendation: Why?

- *"Recommend me five books"* ✓
- *"Recommend me five books, based on my location"* ✓
- *"Recommend me five books, based on my age"* ?
- *"Recommend me five books, based on my job"* ?
- *"Recommend me five books, based on my budget"* ?

A need for a system that generates context-aware recommendations

Recathon:
A Context-Aware Recommender System

Main Idea: Treat recommender systems in the same way as indexing in databases

- **Same as Indexing:**
  - A recommender can be built on one (or more) attribute(s)
  - A recommender can be dropped anytime
  - A recommender is maintained with inserting new items
  - There are different methods of building a recommender
- **Different from Indexing**
  - A query needs to explicitly specify which recommender model to use
  - Recommenders are maintained differently based on query and transaction workload
  - Recommenders can be maintained partially to provide part of the final answer or fully to directly give the final answer

Recathon: Creating a Recommender

CREATE RECOMMENDER *RecommenderName*
USERS FROM *User_Table_Name*
ITEMS FROM *Items_Table_Name*
RATINGS FROM *Rating_Table_Name*
ATTRIBUTES *User_Attributes*
USING *RecommenderMethod*

CREATE RECOMMENDER *AgeRec*
USERS FROM *MovieUsers*
ITEMS FROM *MovieTable*
RATINGS FROM *MovieRating*
ATTRIBUTES *Age*
USING *ItemBasedCF*

CREATE RECOMMENDER *AgeCityRec*
USERS FROM *MovieUsers*
ITEMS FROM *MovieTable*
RATINGS FROM *MovieRating*
ATTRIBUTES *Age, City*
USING *SVD*

UserID	Age	City	Salary
1	20	Minneapolis	3K
2	35	Saint Paul	4K
3	18	Falcon Heights	3.5K
4	23	Edina	5K
5	31	Minnetonka	10K

MovieUsers

ItemID	Movie
1	Lord of the Rings
2	Manhattan
3	The Good, the Bad, and the Ugly

MovieTable

UserID	ItemID	Rating
1	1	3.5
1	3	4.5
3	2	1.5
4	1	5.0
5	1	3.0

MovieRating

Recathon: Querying a Recommender

- Once a recommender is created, a set of intermediate tables and views are created
- Tables and views are continuously maintained, based on an adaptive maintenance technique
- Recommenders are exposed to Recathon users as views that can be queried with standard SQL

Recommend me a movie based on my Age

```
SELECT          ItemID
FROM            AgeRec R1
RECOMMEND(10)   R1.uid = 1 AND
                R1.age = 20
```

Recommend me a movie based on my Age & City

```
SELECT          ItemID
FROM            AgeRec R1
RECOMMEND(10)   R1.uid = 1 AND
                R1.age = 20 AND
                City = ‘Edina’
```

Talk Outline

- **Background on Recommender Systems**
- **DBMS for Recommender Systems**
- **RecBench:** *A Benchmark for Recommender System Architectures*
- **RecStore:** *A Storage Engine Support for Recommender Systems*
- **LARS:** *A Context-Aware Recommender System*
- **Recathon:** *A Context-Aware Recommender System*
- **Summary, Commercial Ads, and Acknowledgments**

Summary

Rating Updates

- *"Recommend me five books"*
- *"Recommend me five books, based on my location"*
- *"Recommend me five books, based on my age"*
- *"Recommend me five books, based on my job"*
- *"Recommend me five books, based on my budget"*

Related Publications

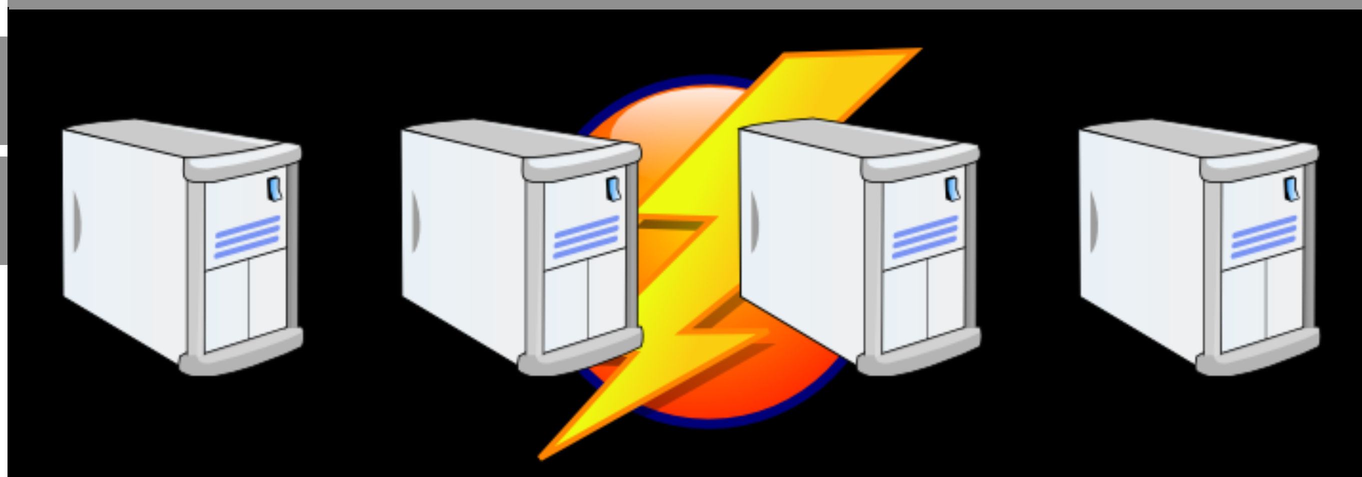
- **Papers**
  - Mohamed Sarwat, Justin J. Levandoski, Ahmed Eldawy, and Mohamed F. Mokbel. "***LARS*: A Scalable and Efficient Location-Aware Recommender System***". IEEE Transactions on Knowledge and Data Engineering, **TKDE 2013**, To Appear.
  - Justin J. Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F. Mokbel. " ***LARS: A Location-Aware Recommender System***". In Proceedings of the IEEE International Conference on Data Engineering, **ICDE 2012**, Washington D.C., April 2012
  - Justin J. Levandoski, Mohamed Sarwat, Mohamed F. Mokbel, and Michael D. Ekstrand. " ***RecStore: An Extensible and Adaptive Framework for Online Recommender Queries inside the Database Engine***". In Proceedings of the International Conference on Extending Database Technology, **EDBT 2012**, Berlin, Germany, March 2012
  - Jie Bao, Yu Zheng and Mohamed Mokbel. "***Location-based and Preference-Aware Recommendation Using Sparse Geo-Social Networking Data***". In Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, **SIGSPATIAL GIS 2012**, Redondo Beach, California, November 2012
  - Justin J. Levandoski, Michael D. Ekstrand, Michael J. Ludwig, Ahmed Eldawy, Mohamed F. Mokbel and John T. Riedl. " ***RecBench: Benchmarks for Evaluating Performance of Recommender System Architectures***". In Proceedings of the International Conference on Very Lage Databases, **VLDB 2011**, Seattle, WA, September 2011
  - Mohamed Sarwat, James Avery, and Mohamed F. Mokbel. " ***Recathon: A Unified Architecture for Personalized Recommendation in Database Systems***". Under Preparation.
- **Demos**
  - Mohamed Sarwat, Jie Bao, Ahmed Eldawy, Justin J. Levandoski, Amr Magdy, and Mohamed F. Mokbel. "***Sindbad: A Location-based Social Networking System***". In Proceedings of ACM SIGMOD Conference on Management of Data, **ACM SIGMOD 2012**, Scottsdale, AZ, May, 2012.
  - Badrish Chandramouli, Justin J. Levandoski, Ahmed Eldawy and Mohamed F. Mokbel. "***StreamRec: A Real-Time Recommender System***". In Proceedings of ACM SIGMOD Conference on Management of Data, **ACM SIGMOD 2011**, Athenes, Greece, Jun., 2011.
  - Mohamed Sarwat, James Avery, and Mohamed F. Mokbel. " ***Recathon: A Unified Architecture for Personalized Recommendation in Database Systems***". Under Preparation.

Commercial Ads

Spatial Hadoop

Analyze your spatial data efficiently

Efficient Spatial Operations

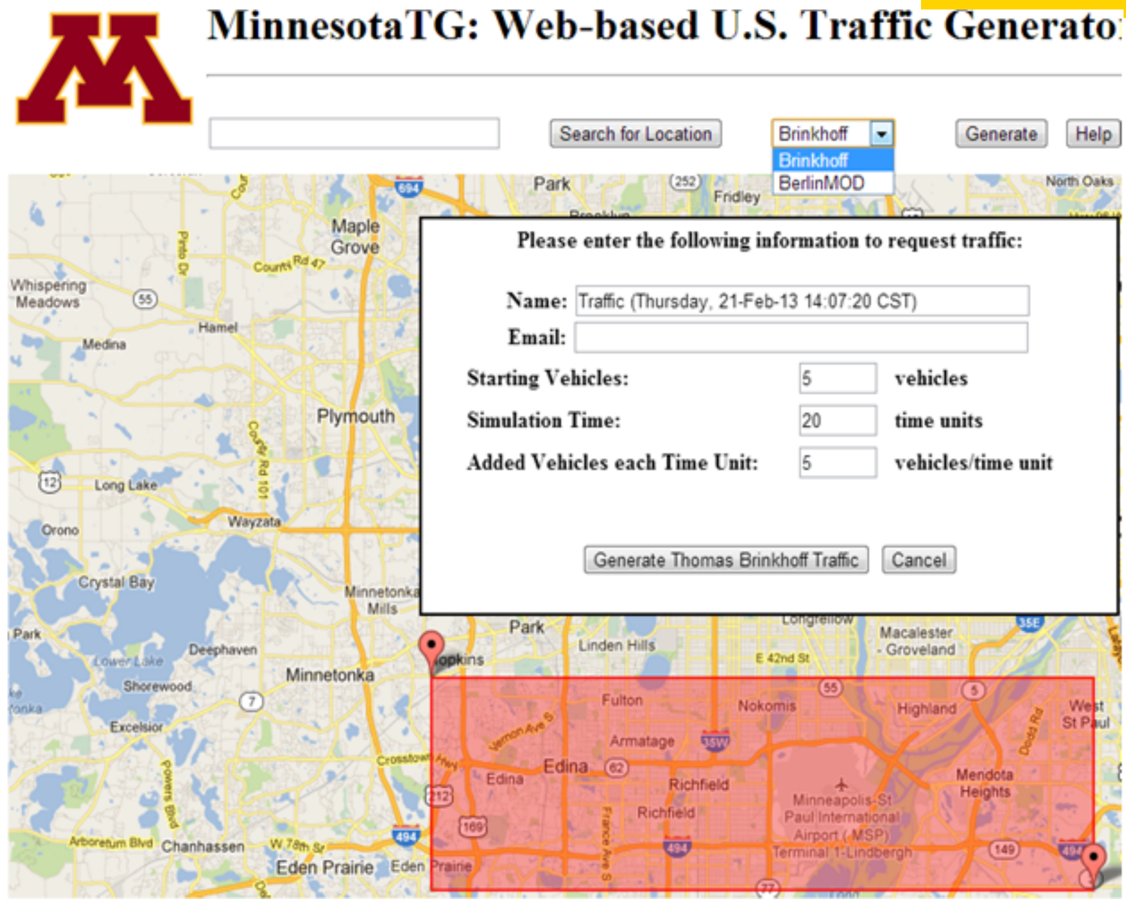


Analyze your data on large clusters with built-in spatial

Website: *http://spatialhadoop.cs.umn.edu/*
Download source code, binary distribution, and instructions

MNTG: Web-based Traffic Generator

- **Easy-to-generate traffic data for road networks**
  - No need to do the installation/configuration
  - Very easy to get the data, just clicks
  - Works for road networks in US
  - Dedicated server for data generation
  - Email notifications
  - Visualization tools



Website: *http://mntg.cs.umn.edu*
Video: *http://www.youtube.com/watch?v=dVP4oc0k9nU*

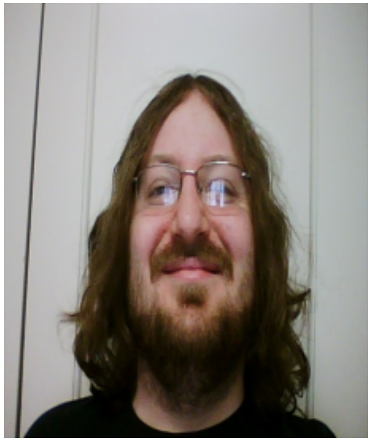
Acknowledgment

The RecBench, RecStore, LARS, and Recathon Team

- **Ph.D. Alumni**

Justin Levandoski
(PhD, 2011)
Researcher at Microsoft Research (MSR) -- Database Group,
Redmond, USA

- **Current Members: Ph.D. Students**



James Avery

Ahmed Eldawy

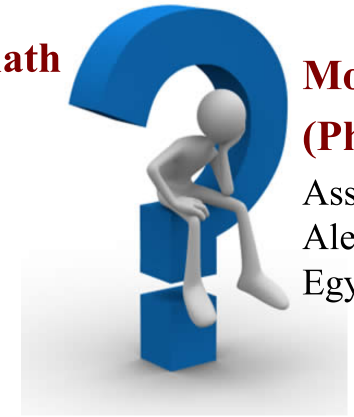
Mohamed Sarwat

Other Group Members

- ## Ph.D. Alumni

Chi-Yin Chow
(PhD, 2010)
Assistant Professor at City University of Hong Kong, Hong Kong

Biplob Debnath
(PhD, 2011)
Researcher, NEC Labs, NJ, USA



Mohamed Khalefa
(PhD, 2011)
Assistant Professor, Alexandria University, Egypt

- ## Current Members: Ph.D. Students

Louai Al-Arabi

Jie Bao

Abdeltawab Hendawi

Amr Magdy

Acknowledgment: Funding

- **NSF-IIS:** Towards Spatial Database Management Systems for Flash Memory Storage. 2012 -2015
- **Microsoft Research.** *Microsoft Unrestricted Gift, October, 2010*
- **NSF- CAREER:** Extensible Personalization of Spatial and Spatio-temporal Database Management Systems. 2010 -2015
- **Microsoft Research.** *Microsoft Unrestricted Gift, January, 2010*
- **Microsoft Research.** *Microsoft Unrestricted Gift, April, 2009*
- **NSF- IIS:** Towards Ubiquitous Location Services: Scalability and Privacy of Location-based Continuous Queries. 2008 -2012
- **NSF- IIS:** Preference- And Context-Aware Query Processing for Location-based Data-based servers. 2008 -2012
- **NSF- CNS:** Infrastructure for Research in Spatio-Temporal and Context-Aware Systems and Applications. 2007 - 2012

Thanks