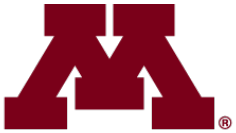
CareDB: A Context and Preference-Aware Database System

Justin Levandoski
Microsoft Research

Mohamed Khalefa
Alexandria University

Mohamed Mokbel
University of Minnesota



UNIVERSITY OF MINNESOTA

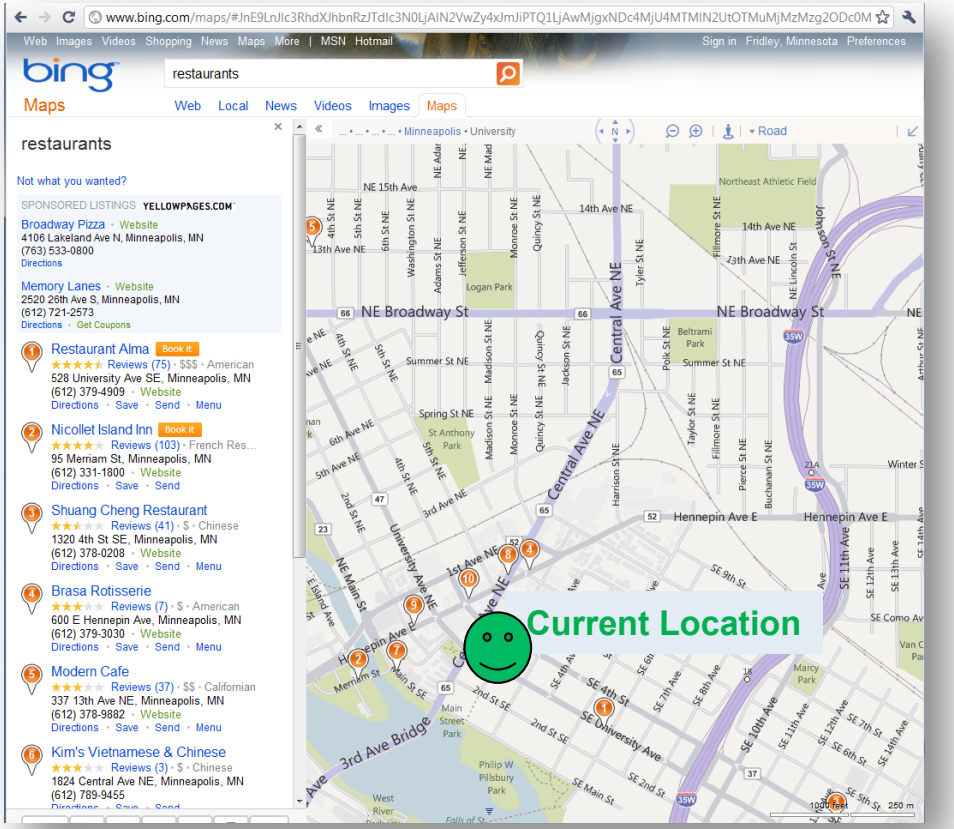
Why a Context and Preference-Aware DBMS?

Consider a location-based restaurant finder

"Find me a restaurant for dinner"

Why a Context and Preference-Aware DBMS?

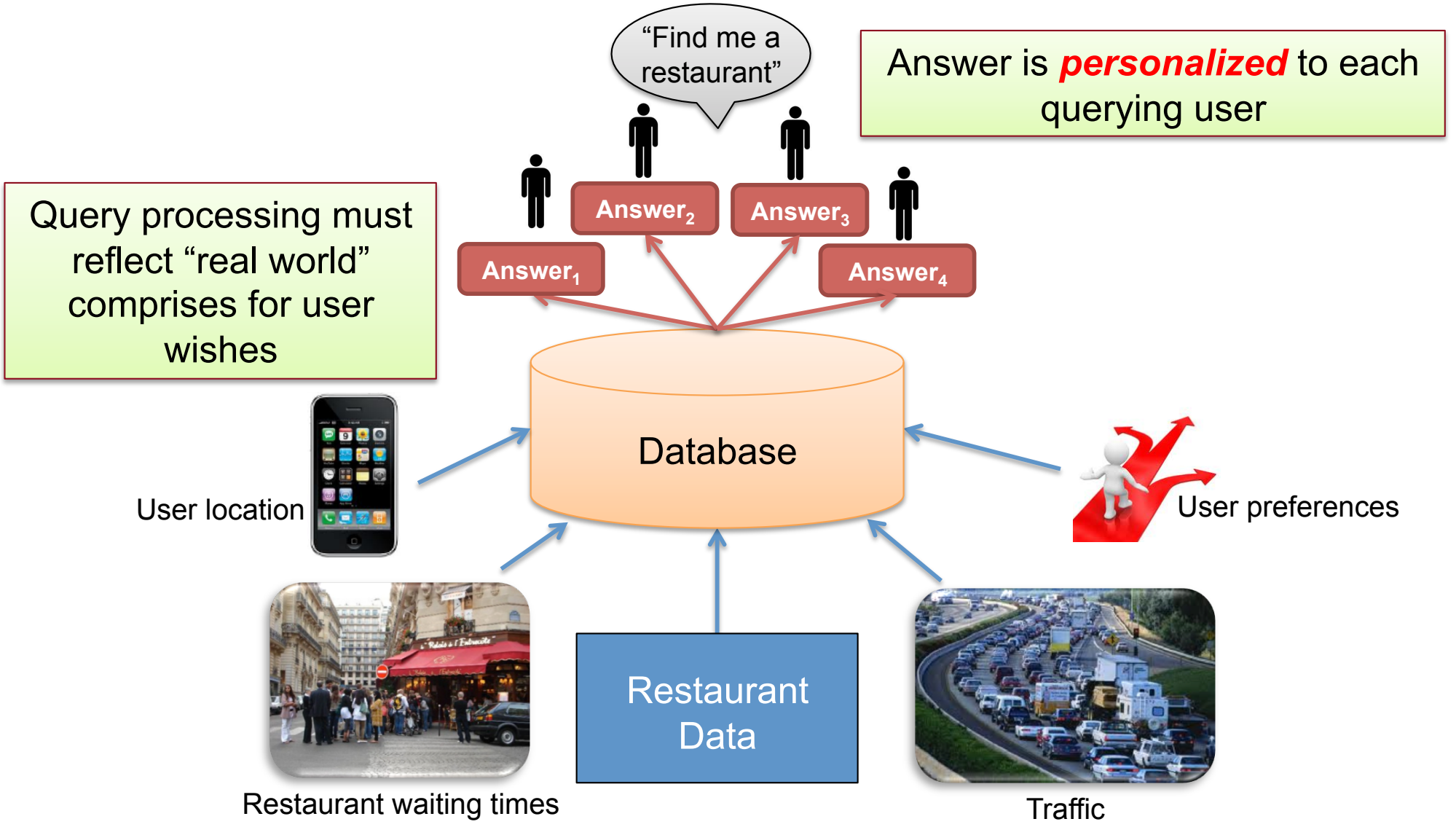
- Existing applications will return *k closest* restaurants
- Consider the following five closest restaurants
  1) *Restaurant 1:*
     - Hour and a half wait
  2) *Restaurant 2:*
     - Does not meet my dietary restrictions
  3) *Restaurant 3:*
     - Way too expensive
  4) *Restaurant 4:*
     - Closed for remodeling
  5) *Restaurant 5:*
     - 30-minute drive time due to accident along route



The answers are NOT useful and detached from:

1. **Personal preferences** (dietary restrictions, budget)
2. **Extra contextual data** (time of day, traffic, waiting times)

A Context and Preference Aware Environment



Input data contains user preferences and dynamic contextual information; each type can uniquely affect the answer, be uncertain, expensive to derive, etc.

CareDB: A Context and Preference-Aware Database

A database that is aware of

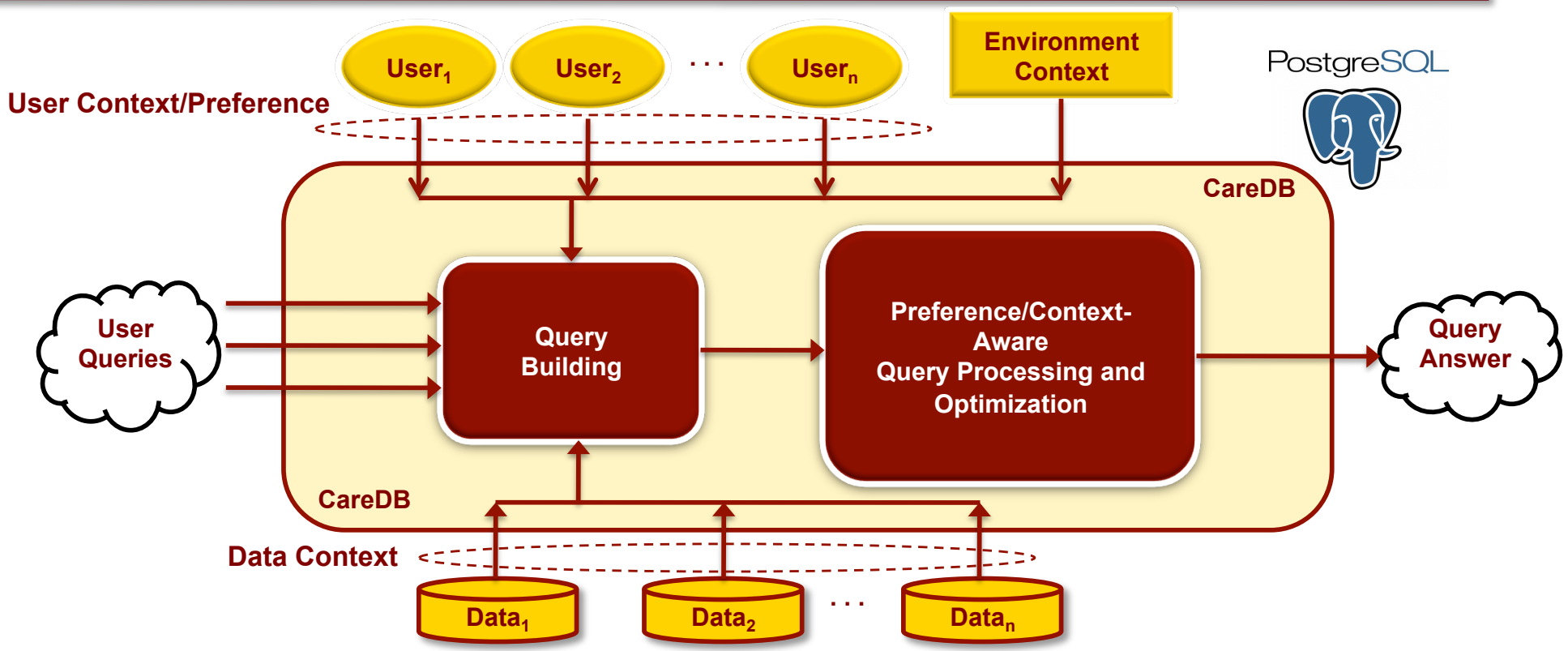
user preferences and

surrounding contextual information,

and uses this information to give

personalized **query answers** to the user.

CareDB Overview



Three Classes of Input to CareDB:

User Context/Preference

User location
User status
User salary
User preferences
……

Environmental Context

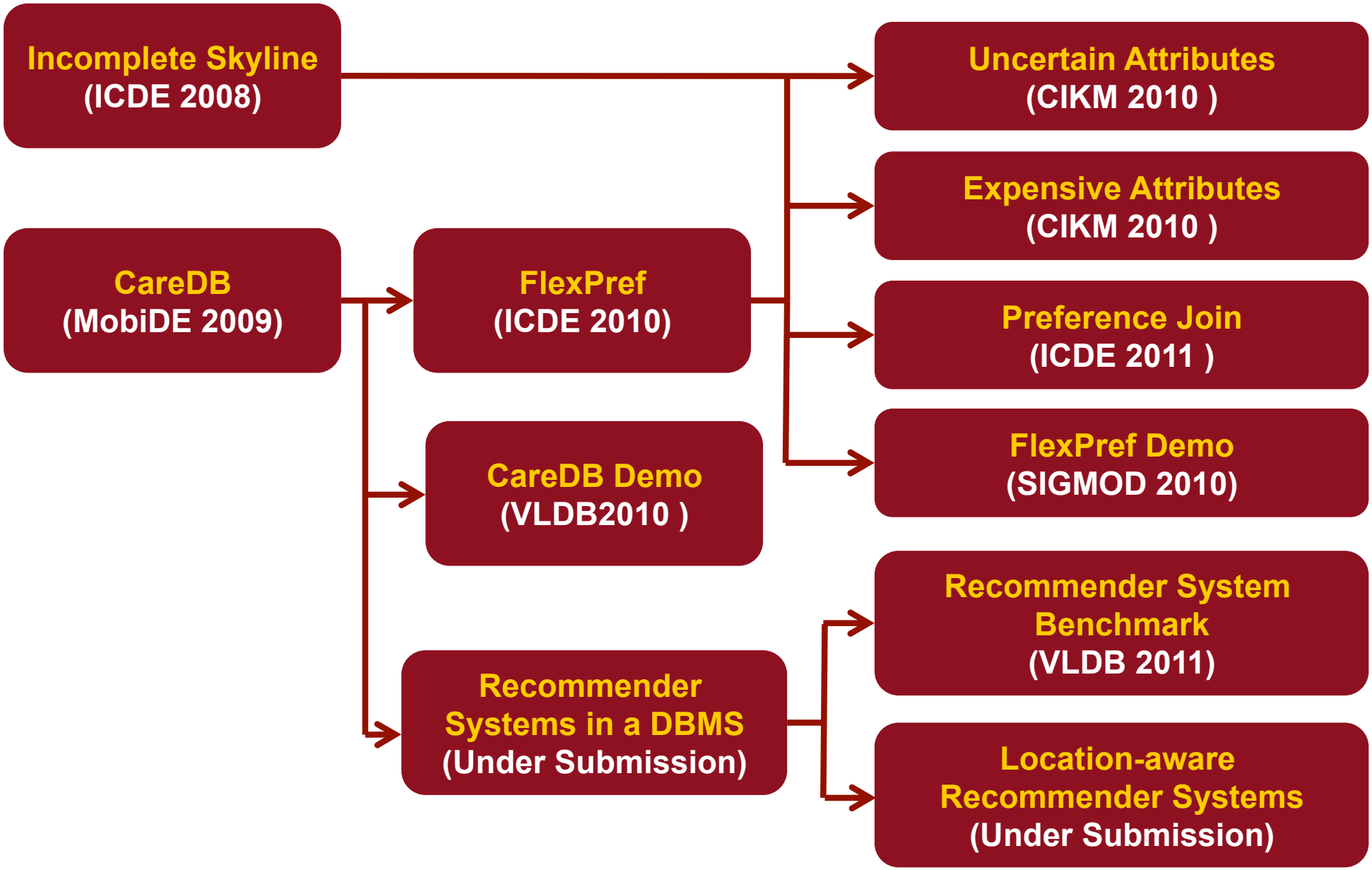
Traffic
Weather
Road Network
Transportation
……

Database Context

Restaurant waiting time
Restaurant location
Restaurant rating
Restaurant price
……

"Towards Context and Preference-Aware Location-Based Database Systems", MobiDE 2009

CareDB Project Summary



Outline

- Background
- The FlexPref Framework
- Handling Contextual Data in CareDB
- Handling Uncertain Data in CareDB
- Demonstration
- Conclusion

Preference Methods

```
SELECT *
FROM   Restaurants R
WHERE  R.AllowsGroups = True;
```

Inject Preference Functionality

```
SELECT       *
FROM         Restaurants R
WHERE        R.AllowsGroups = True
PREFERRING   MIN R.Price,
             MAX R.Rating,
             MIN R.WaitTime,
             MIN TravelTime(User.Location, R.Location)
```

What is the query answer?

What preference method evaluates the PREFERRING clause?

Preference Methods

- **Quick Exercise**
  - Go to Google Scholar
  - Search for papers on preference evaluation methods
  - How many results do you get back?

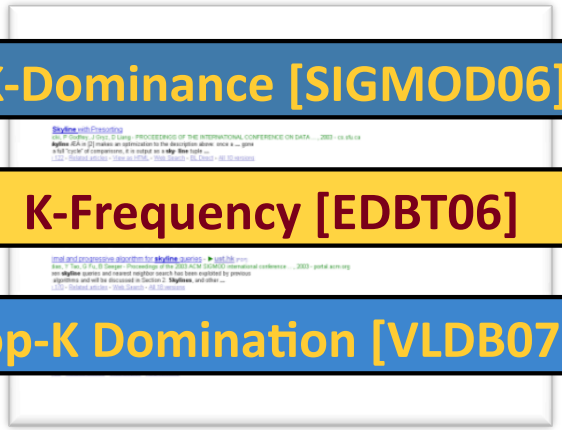
The list goes on and on and on…

Top-K [VLDB99]

Skyline [ICDE01]

K-Dominance [SIGMOD06]

K-Frequency [EDBT06]



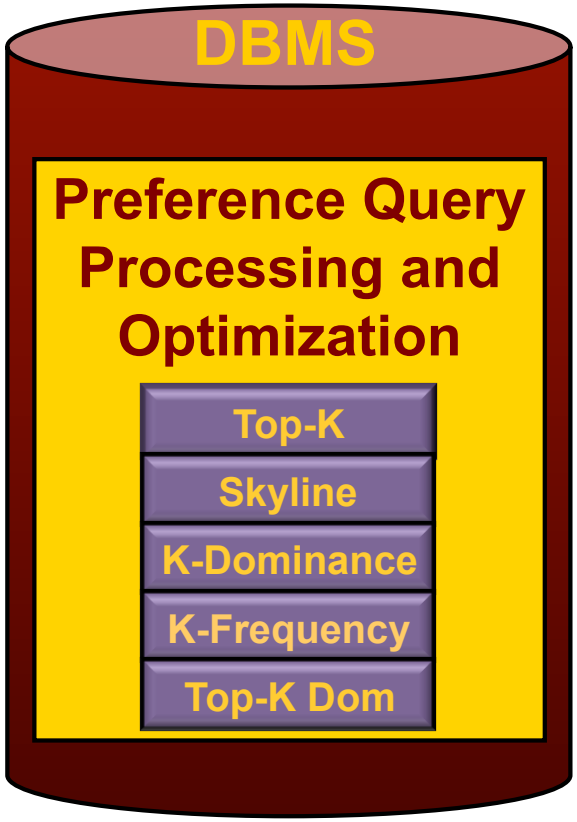
Top-K Domination [VLDB07]

Preference Evaluation in a DBMS

Layered Approach *(The Bad)*

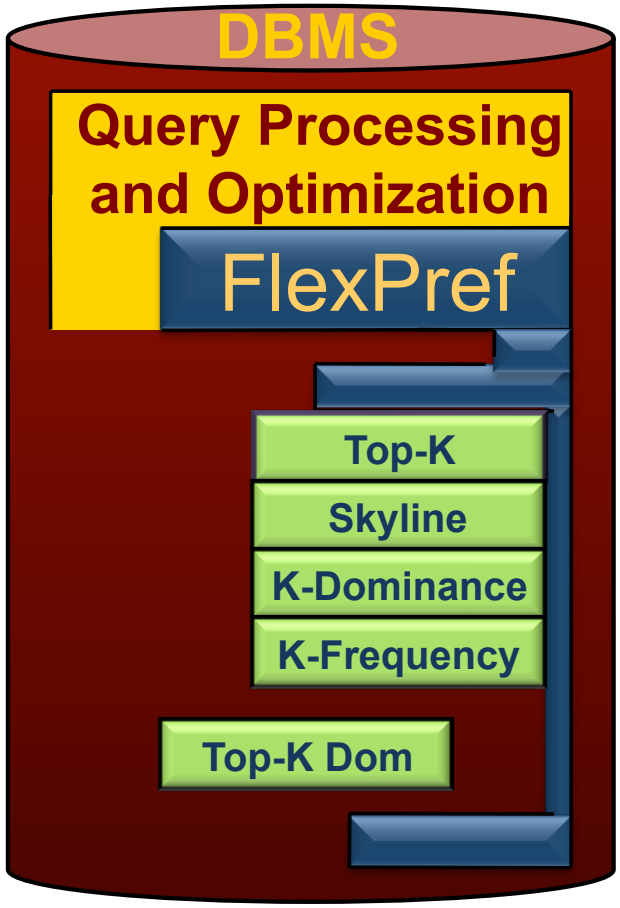
Skyline implementation:
~200 lines of code
Bad Performance

Built-in Approach *(The Ugly)*



Skyline implementation:
~8000 lines of code
Good Performance

FlexPref *(The Good)*

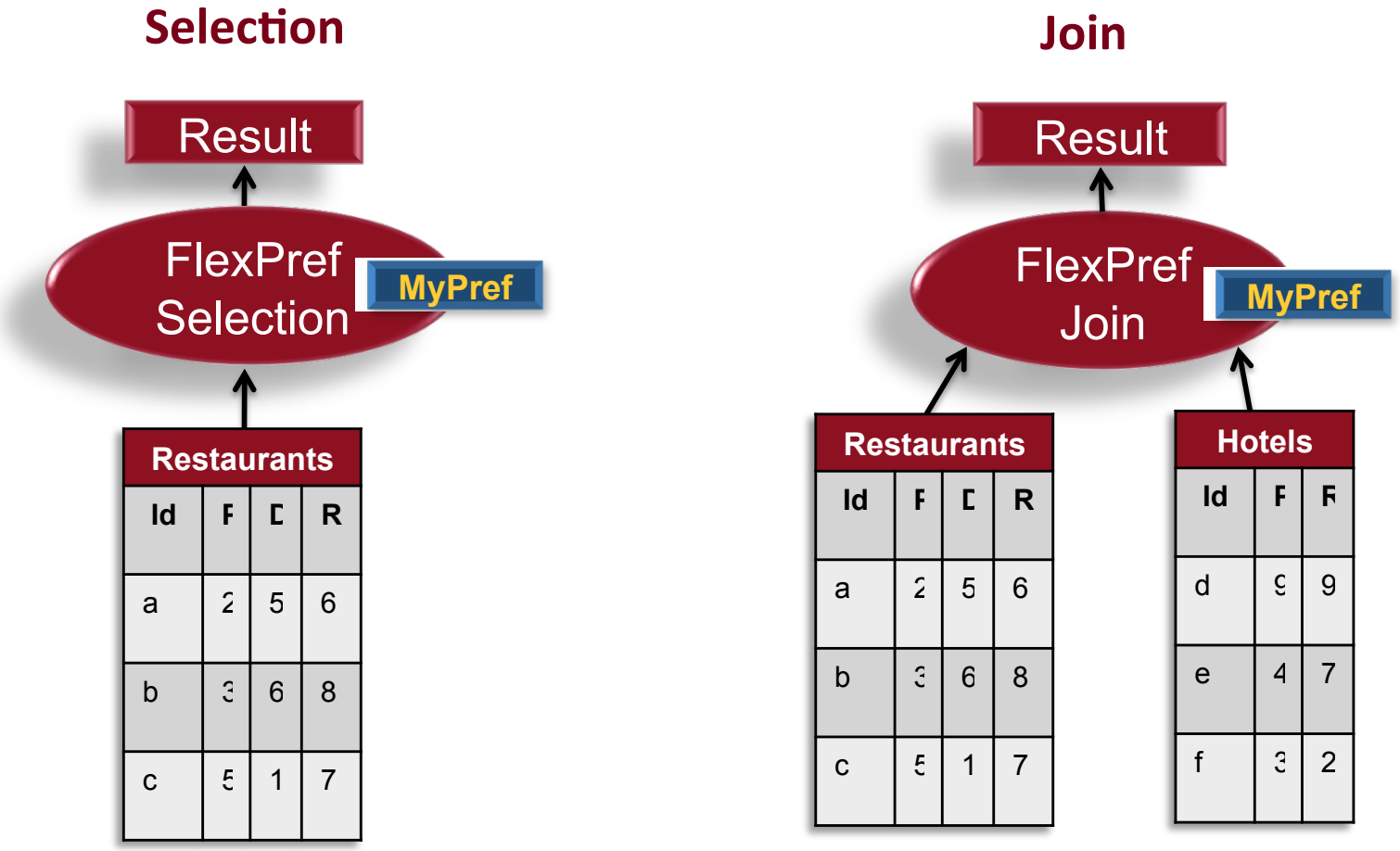


Skyline implementation:
~300 lines of code for selection and join
Good Performance

"FlexPref: A Framework for Extensible Preference Evaluation in Database Systems", ICDE 2010

The FlexPref Architecture

- FlexPref is a set of extensible relational operators
- Examples:



Writing Preference Queries in FlexPref

```
SELECT * FROM Restaurants R WHERE [Where_clause]
PREFERRING [Attribute List]
USING [Pref Method]
OBJECTIVES [Preference Objectives]
```

```
SELECT * FROM Restaurants R
WHERE …
PREFERRING Price P, Distance D, Rating R
USING Skyline OBJECTIVES MIN P, D, MAX R
```

```
SELECT * FROM Restaurants R
WHERE …
PREFERRING Price P, Distance D, Rating R
USING TopKDom WITH K=5 OBJECTIVES MIN P, D, MAX R
```

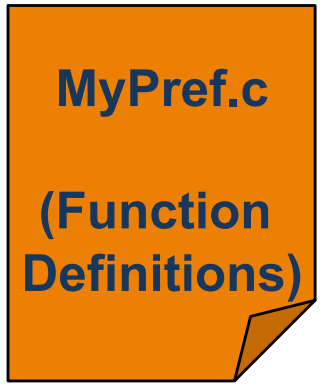
```
SELECT * FROM Restaurants R
WHERE …
PREFERRING Price P, Distance D, Rating R
USING TopK WITH K=5 OBJECTIVES MIN Func(P,D,R)
```

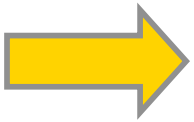

Adding a New Preference Method to FlexPref

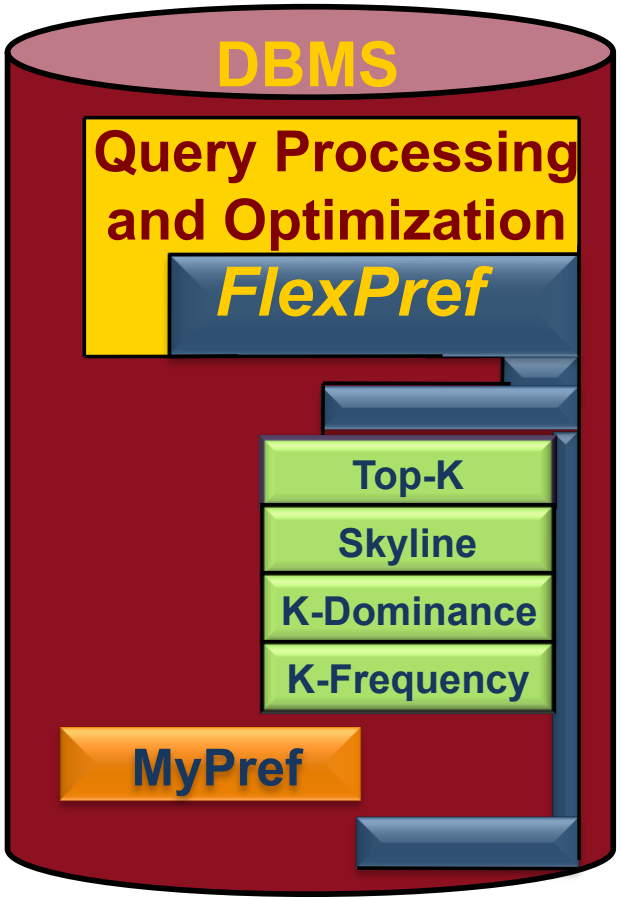
- Adding a preference evaluation method "`MyPref`" to FlexPref requires the implementation of *three functions* and *two macros* in a separate file "`MyPref.c`" outside the DB engine.

- Once implemented, the preference method is registered using a **`DefinePreference`** command

```
DefinePreference MyPref with MyPref.c
```







FlexPref Generic Functions & Macros

FlexPref Macros

#define DefaultScore
Default score assigned to each object
#define IsTransitive
Whether preference function is transitive or not

PairwiseCompare(Object P, Object Q)

INPUT: *Two objects P and Q*
ACTION: *Update the score of P*
RETURN: **1** *if Q can never be a preferred object*
-1 *if P can never be a preferred object*
0 *otherwise*

IsPreferredObject(Object P, PreferenceSet S)

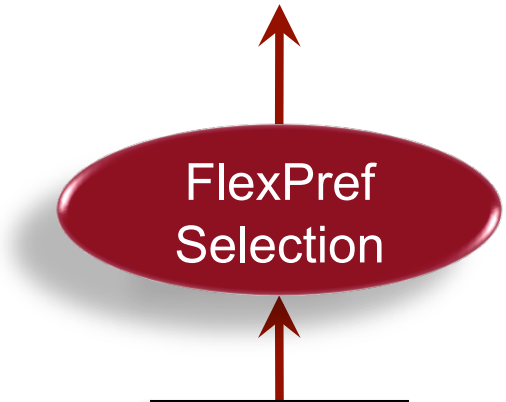
INPUT: *A data object P and a set of preferred objects S*
RETURN: ***True*** *if P is a preferred object and can be added to S*
False *otherwise*

AddPreferredToSet(Object P, PreferenceSet S)

INPUT: *A data object P and a set of preferred objects S*
ACTION: *Add P to S and remove or rearrange objects from S*

FlexPref Operator Implementation

- FlexPref operators written in terms of the three generic functions and two macros:



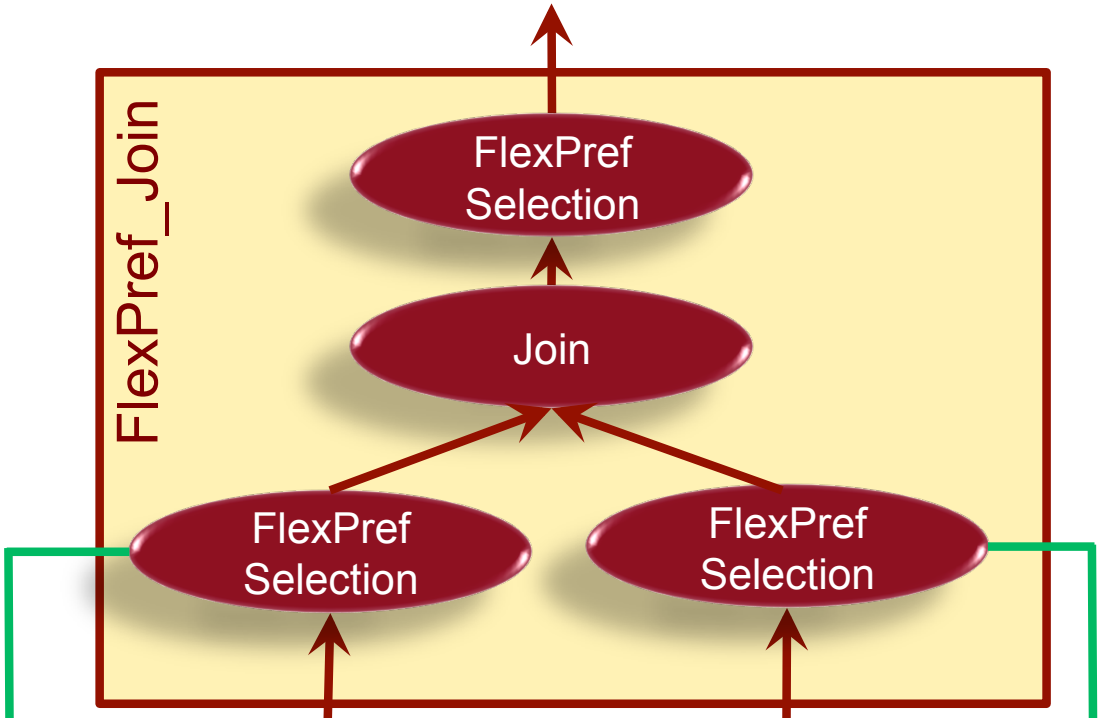
Restaurants			
Id	F	D	R
a	2	5	6
b	3	6	8
c	5	1	7

```
SELECT * FROM Restaurants
PREFERRING ... R
USING Skyline OBJECTIVES
MIN P, MIN D, MAX R
```

```
Input: Single Table T
Output Preference set S

Preference Set S ← NULL
For each object P in T
   P.score = #DefaultScore
   for each Object Q in T
     cmp ← PairwiseCompare(P,Q)
     if (cmp ==1)
        if Q is in S then remove Q
        if #isTransitive
           then discard Q from T
     if (cmp == -1)
        if #isTransitive
           then discard P from T
        read next object P
     if (IsPreferredObject(P,S))
        then AddToPreferredSet(P,S)
Return S
```

FlexPref Join Operation



Restaurants

Id	F	D	F	city
a	2	5	6	Miami
b	3	6	8	Miami
c	5	1	7	Seattle
d	7	2	4	Seattle

Hotels

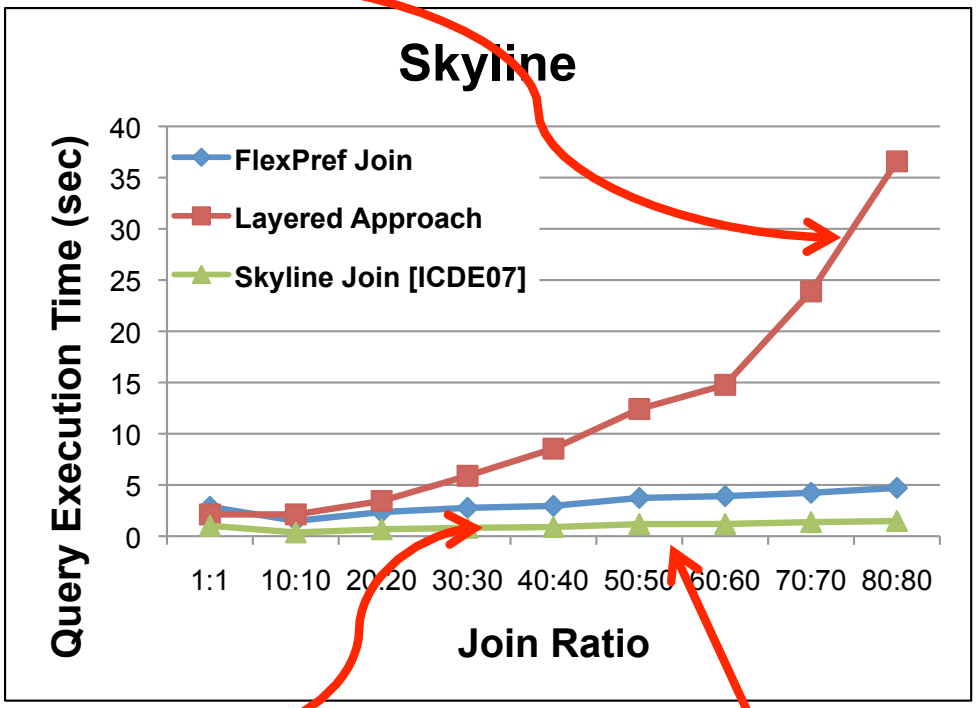
Id	F	R	city
d	9	9	Miami
e	4	7	Miami
f	3	2	Seattle
h	2	6	Seattle

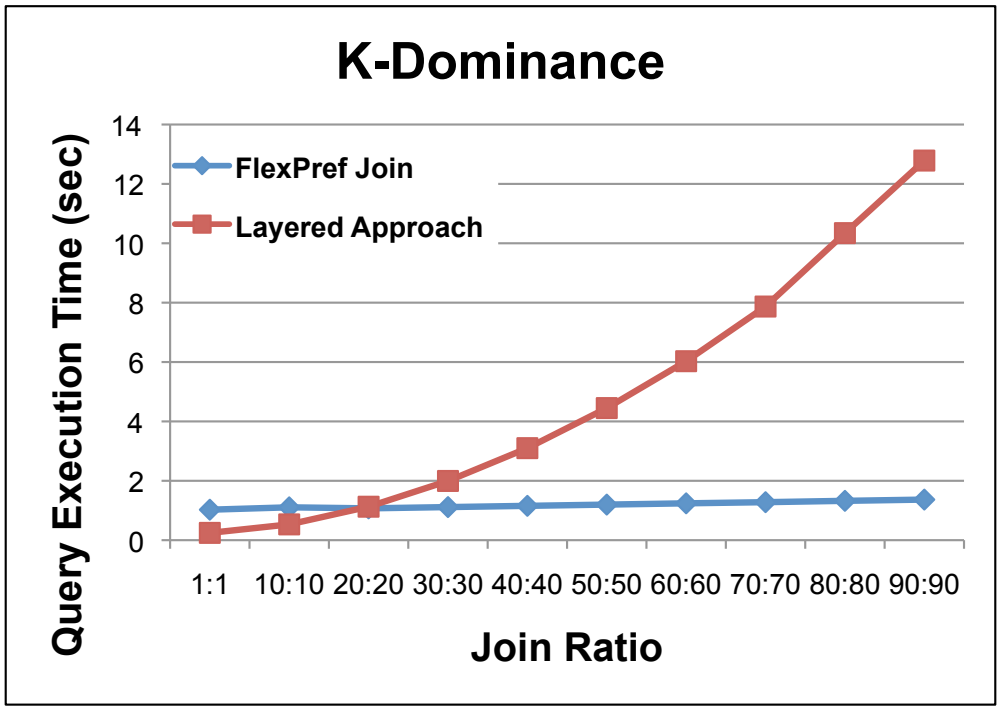
```
SELE                                    nts
WHER                                    =
PREF                    R.D                 ting R
USING Skyline OBJECTIVES MIN P, MIN D, MAX R
```

Performance Analysis

- Comparison of preference-aware join operator

~200 lines of code





~300 lines of code

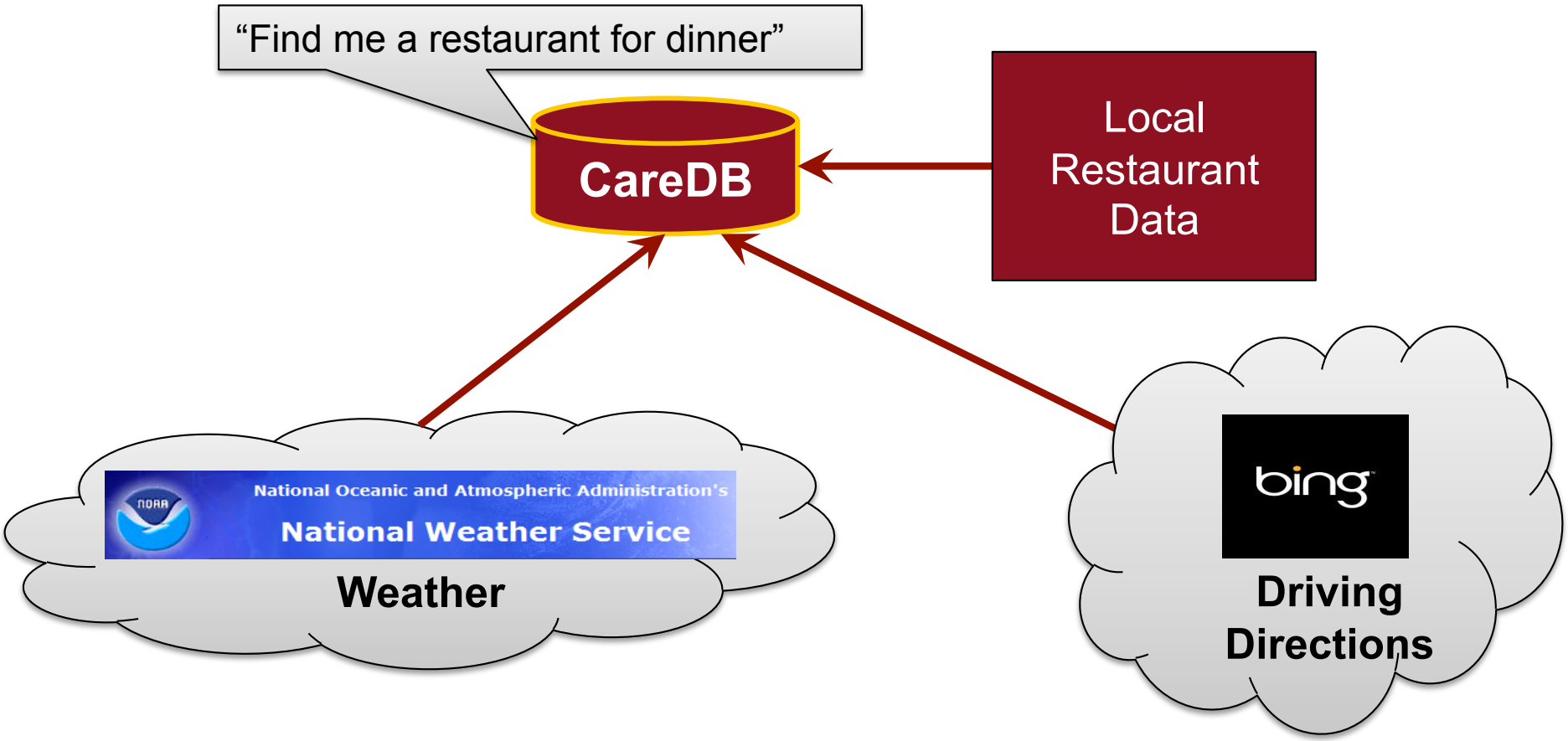
~8,000 lines of code

Outline

- Background
- The FlexPref Framework
- Handling Contextual Data in CareDB
- Handling Uncertain Data in CareDB
- Demonstration
- Conclusion

Contextual Data

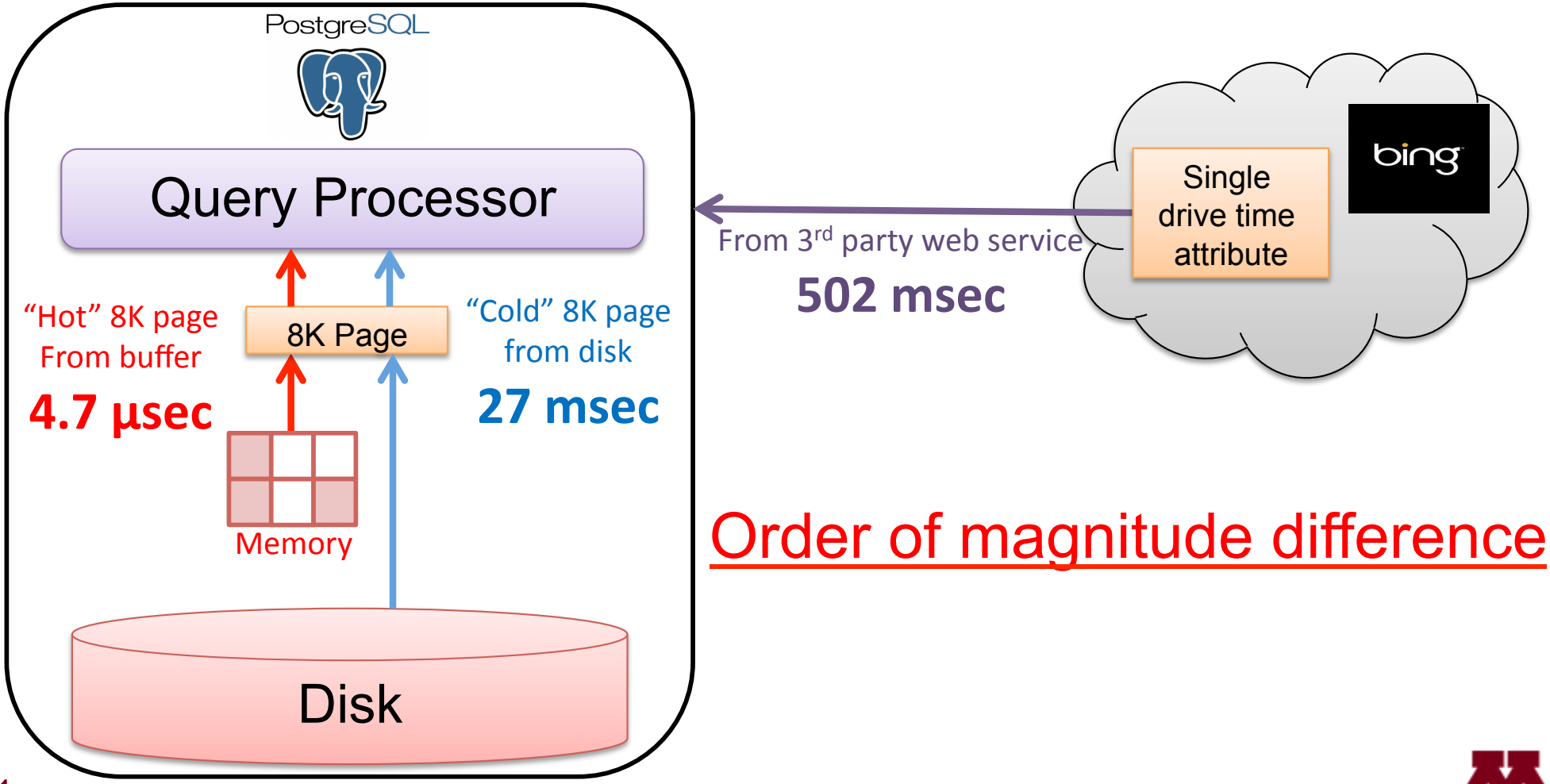
- Data retrieved in “real time” during preference query
- Likely retrieved from third party (e.g., web, “cloud”)



“Preference Query Processing over Expensive Attributes”, CIKM 2010

Contextual Data: Expensive Attributes

- Contextual data is expensive to retrieve relative to local data
- Experiment implemented in PostgreSQL prototype

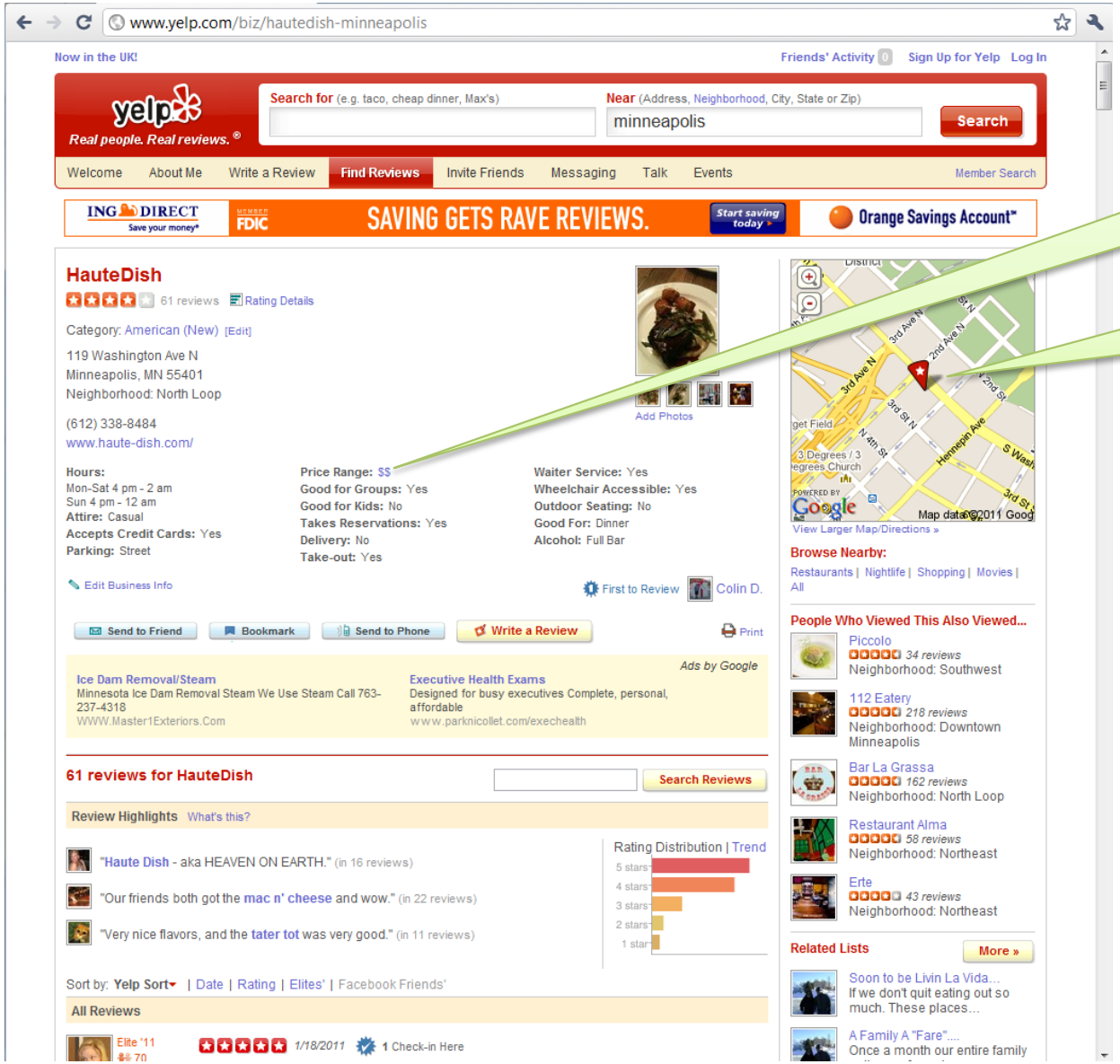


Order of magnitude difference

Outline

- Background
- The FlexPref Framework
- Handling Contextual Data in CareDB
- Handling Uncertain Data in CareDB
- Demonstration
- Conclusion

Contextual Data: Uncertain Data



Third party restaurant prices reported as range, e.g., $15-$20

Driving time reported with error, e.g., 3 minutes [+/- 1 min]

Contextual Data: Handling Uncertain Data

- Some attributes represented as range

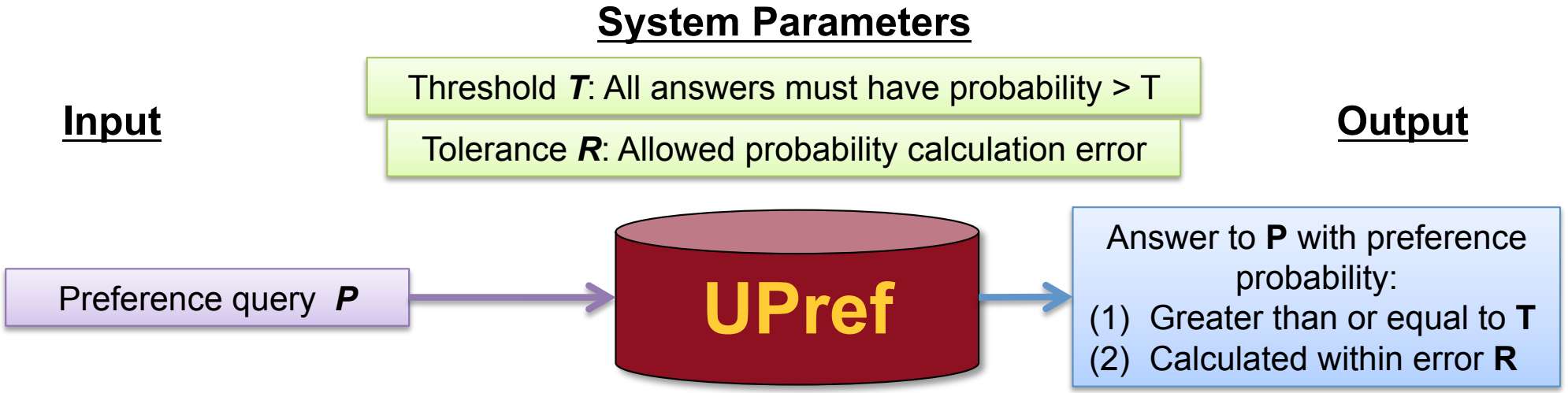
Restaurants			
Id	Price	Dist	Rating
a	5-10	5	6
b	7-15	6	8
c	20-30	1	7
d	15-25	2	4

- Answers are probabilistic

```
SELECT * FROM Restaurants
PREFERRING P, D, R
USING Skyline OBJECTIVES MIN
Price, MIN Dist, MAX Rating
```

Answers	
Id	Preference Probability
a	75%
b	50%

- The **UPref** framework:

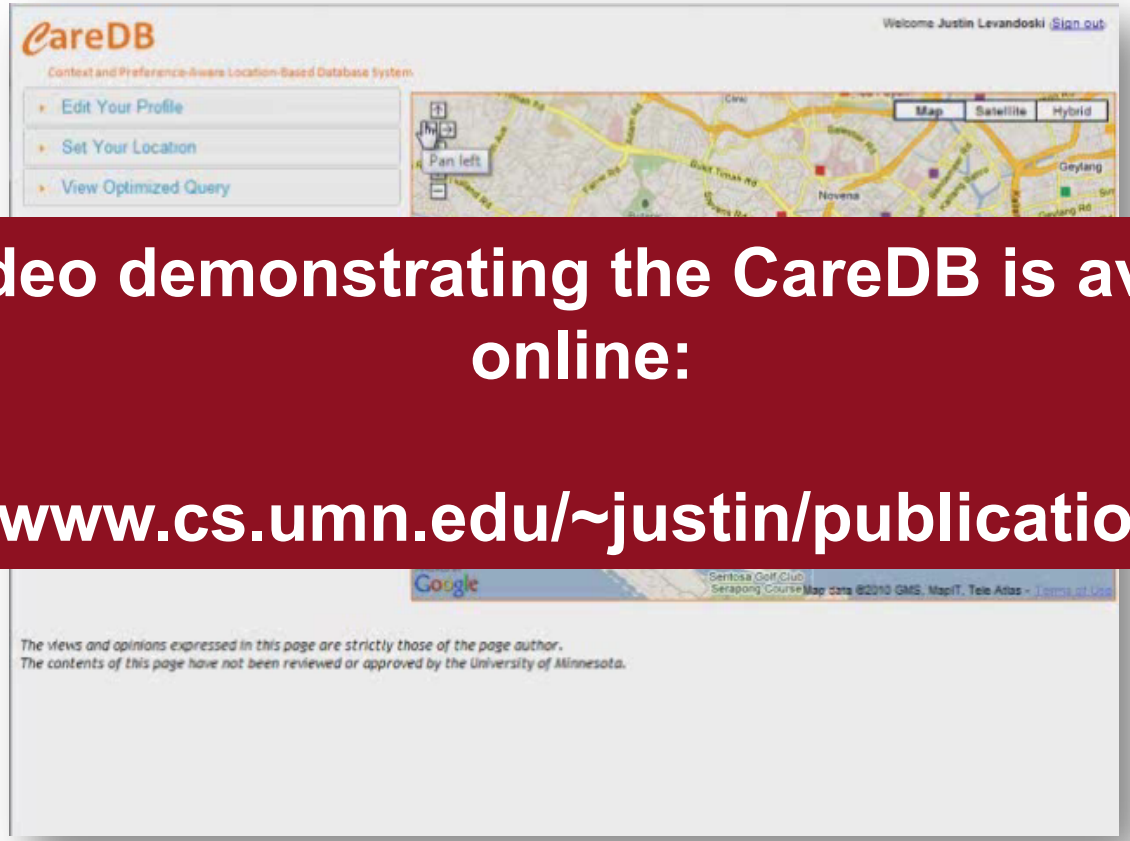


"Preference Query Processing for Uncertain Data", CIKM 2010 / TKDE (Under Submission)

Outline

- Background
- The FlexPref Framework
- Handling Contextual Data in CareDB
- Handling Uncertain Data in CareDB
- Demonstration
- Conclusion

CareDB Prototype Demonstration



Full video demonstrating the CareDB is available online:

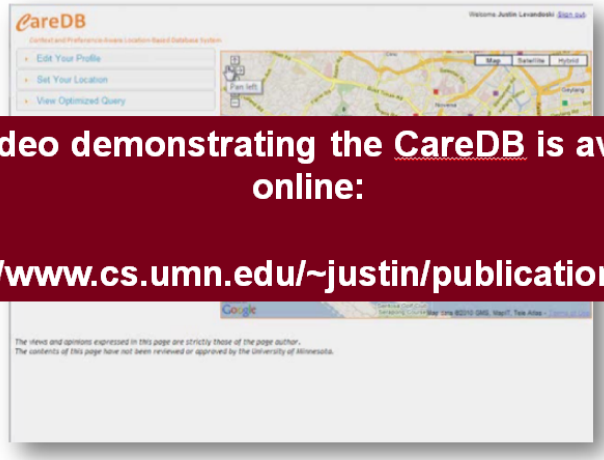
http://www.cs.umn.edu/~justin/publications.html

Outline

- Background
- The FlexPref Framework
- Handling Contextual Data in CareDB
- Handling Uncertain Data in CareDB
- Demonstration
- Conclusion

Summary

CareDB Prototype Demonstration



Full video demonstrating the CareDB is available online:

http://www.cs.umn.edu/~justin/publications.html

30

"CareDB: A Context and Preference-Aware Location-Based Database System", VLDB 2010

Thank You

Questions

Extra Slides

Definition of Context

"Any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves." **[Key, 2001]**

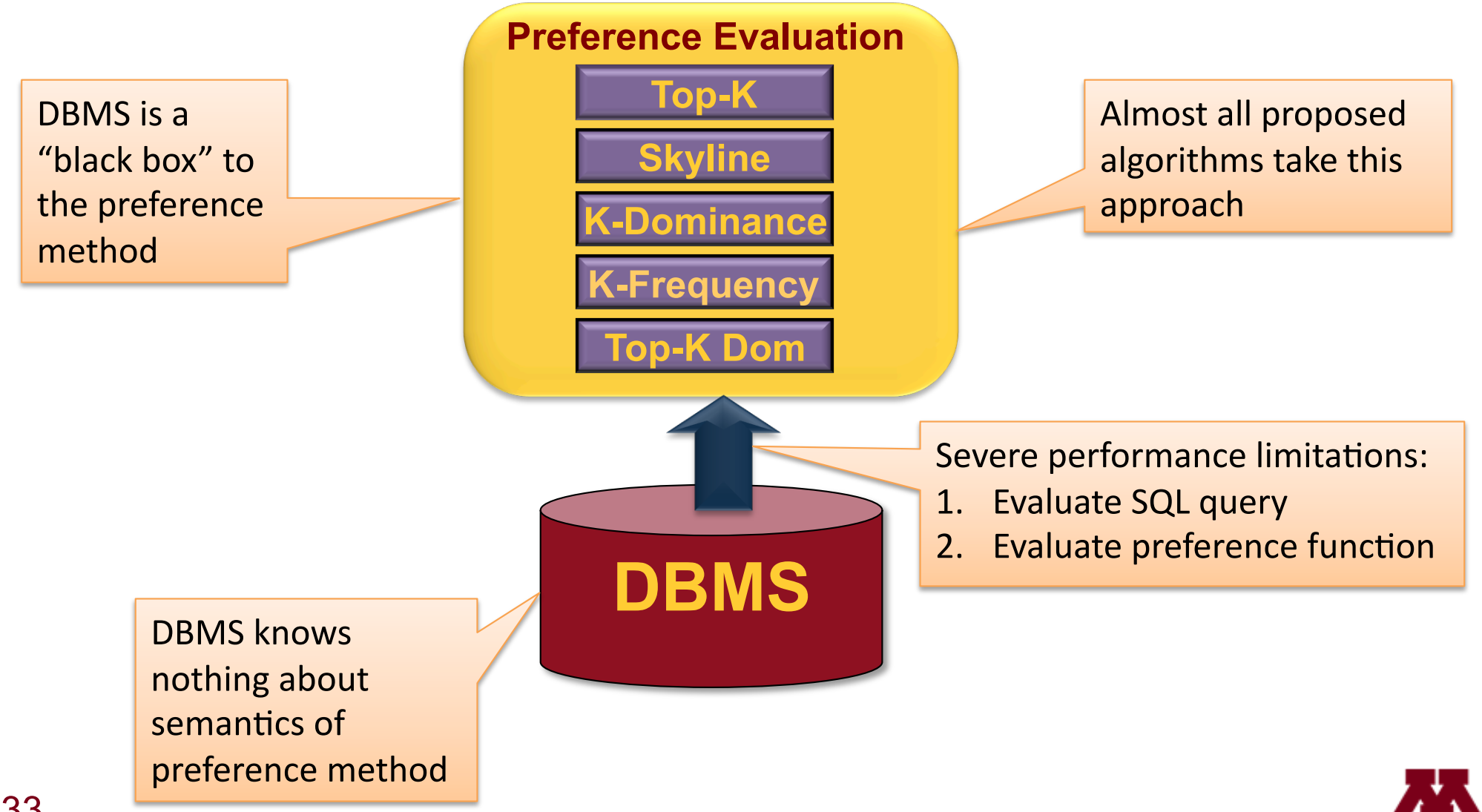
Embedding Preference in a DBMS

- **Notion of "preference" is subjective**
  - Each method challenges notion of "preferred"
  - No limit to the number of proposed methods!
  - We would like to support all methods in CareDB

- **How do we embed these methods in a DBMS?**
  - Must handle arbitrary queries
    - Selection over single table
    - Data may reside in multiple tables
    - Data may be sorted
  - How do we do this efficiently and in a scalable manner?

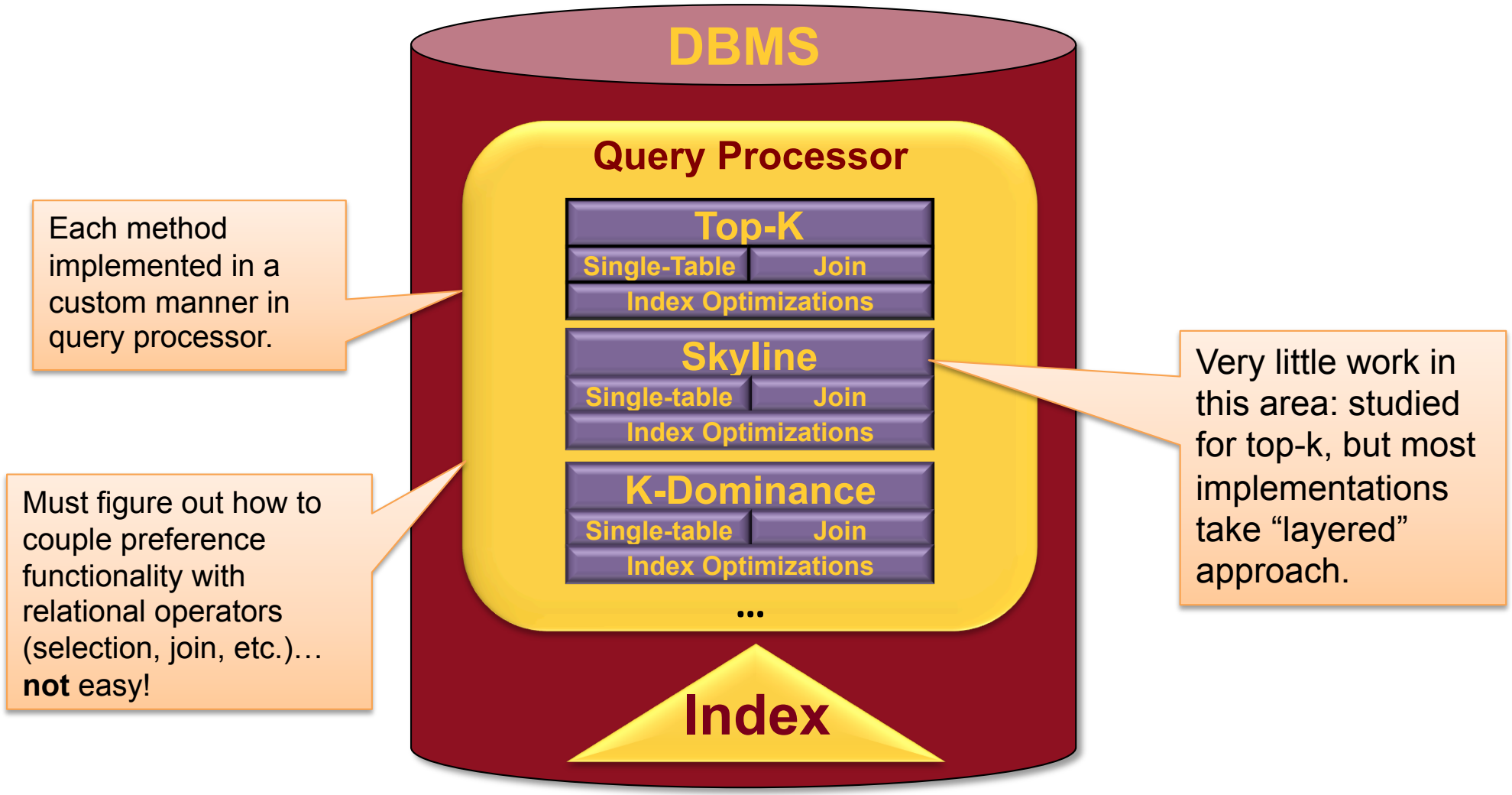
Implementing Preference Methods in a DBMS

The Layered Approach



Implementing Preference Methods in a DBMS

The Built-In Approach



Implementing Preference Methods in a DBMS

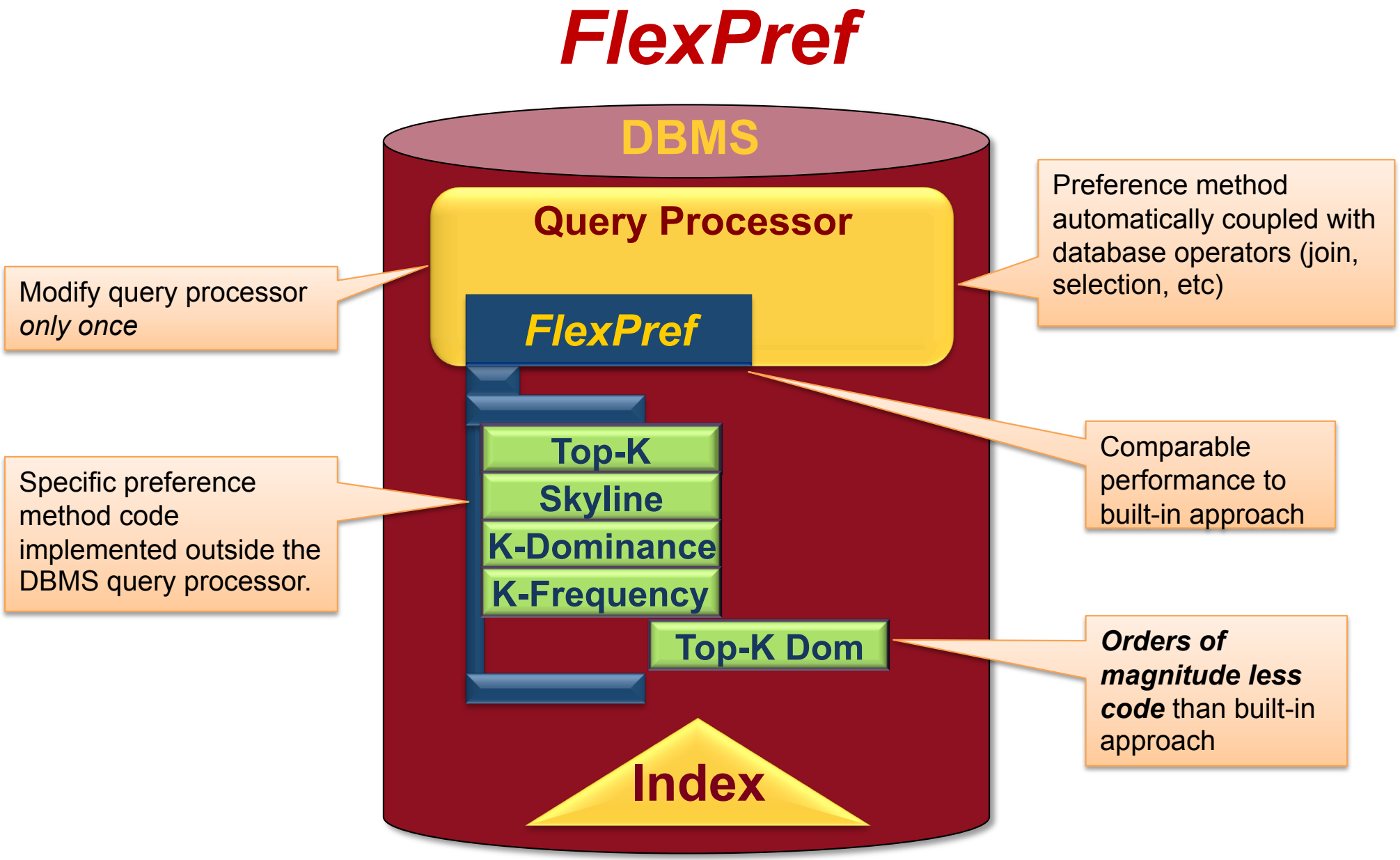
The Layered Approach

- ✓ ***Simplicity***: easy to implement
- ✗ ***Limited Efficiency***: cannot interact with DBMS internals, thus no query optimization

The Built-In Approach

- ✓ ***Efficient***: methods tightly coupled with DBMS
- ✗ ***Infeasible***: cannot provide custom implementation for *every* preference method

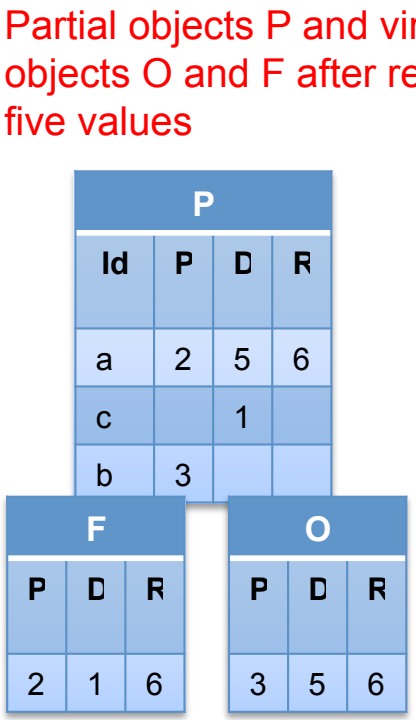
The FlexPref Architecture



"FlexPref: A Framework for Extensible Preference Evaluation in Database Systems", ICDE 2010

FlexPref Sorted List Operator

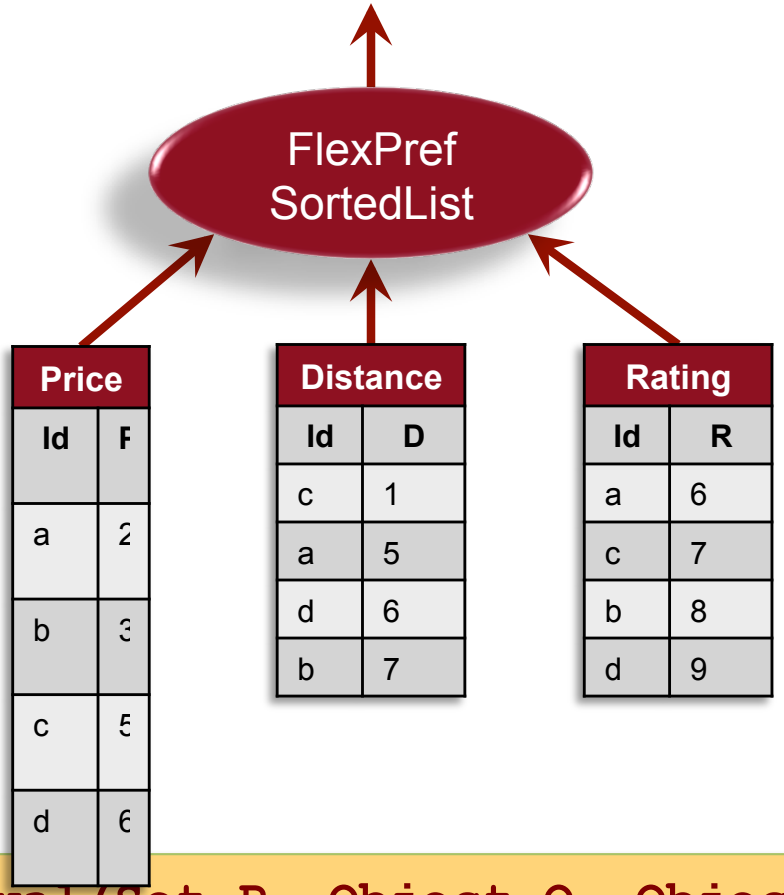
Partial objects P and virtual objects O and F after reading five values



P			
Id	P	D	R
a	2	5	6
c		1	
b	3		

F		
P	D	R
2	1	6

O		
P	D	R
3	5	6



Price	
Id	P
a	2
b	3
c	5
d	6

Distance	
Id	D
c	1
a	5
d	6
b	7

Rating	
Id	R
a	6
c	7
b	8
d	9

Generic Steps
Average case: DN/2 = O(N)

StopSortedEval(Set P, Object O, Object F)

INPUT: *A set of partial objects P and two virtual objects O and F*

ACTION: *Update objects in P, until it is sufficient to perform preference evaluation*

FlexPref: Query Optimization

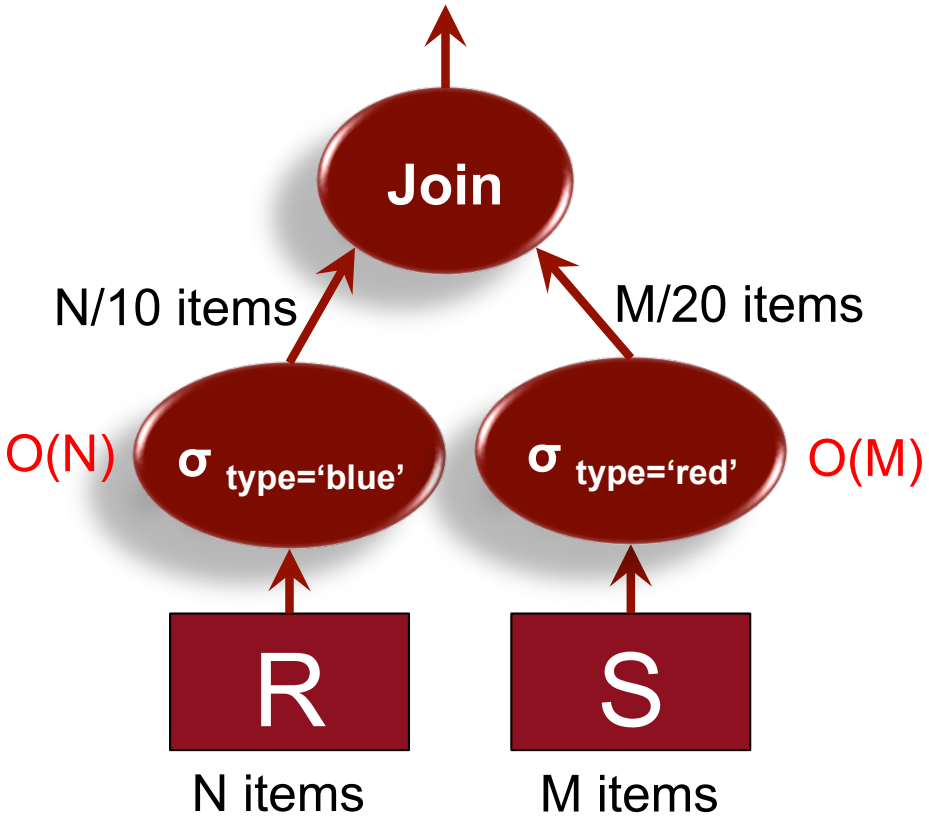
- Why optimize?

We study the three fundamental properties of FlexPref necessary for query optimization:

Algebraic properties

Cost

Cardinality estimation




```
SELECT * FROM R, S
WHERE R.id = S.id AND
      R.type = 'blue' AND
      S.type = 'red';
```

FlexPref Theoretical Framework Examples

- Example 1: Symmetric dominance

Methods exhibiting symmetric dominance not allowed

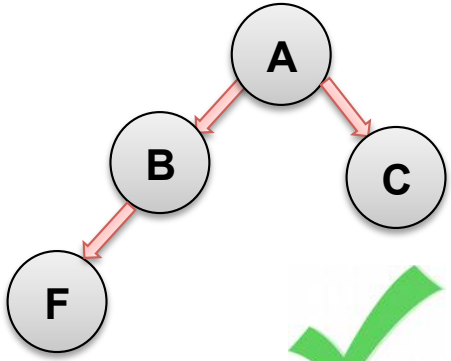
- Example 2: Conditional Preferences

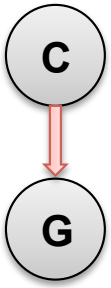
"I prefer cheap Mexican restaurants when it is sunny"

"I prefer expensive French restaurants when rainy"

Flexpref framework does not support semantics of conditional preferences in its PREFERRING clause

- Example 3: Irreflexive, asymmetric, transitive dominance





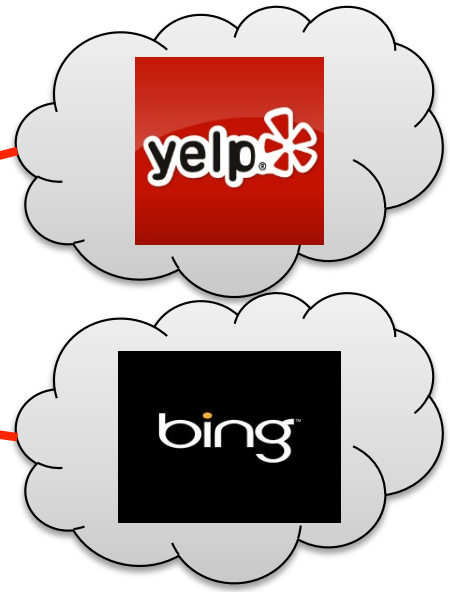
Flexpref supports preference methods that generate strict partial order

Preference Queries Over Expensive Attributes

- Preference queries with a mix of “local” and “expensive” attributes
- Goal: retrieve the least amount of expensive attributes

```
SELECT *
FROM    Restaurants R
PREFERRING MIN R.Price, MIN R.Distance, MAX R.Rating
```

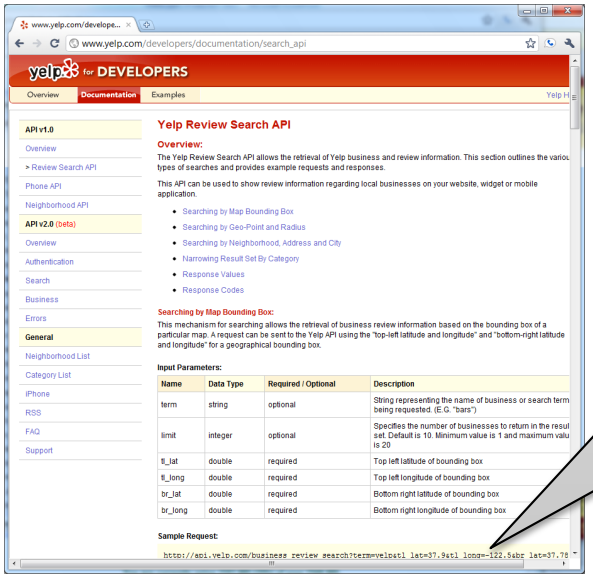
Restaurants			
Id	**Price**	**Distance**	**Rating**
a	2		
b	3		
c	5		



Stored locally in DBMS

Contextual Data

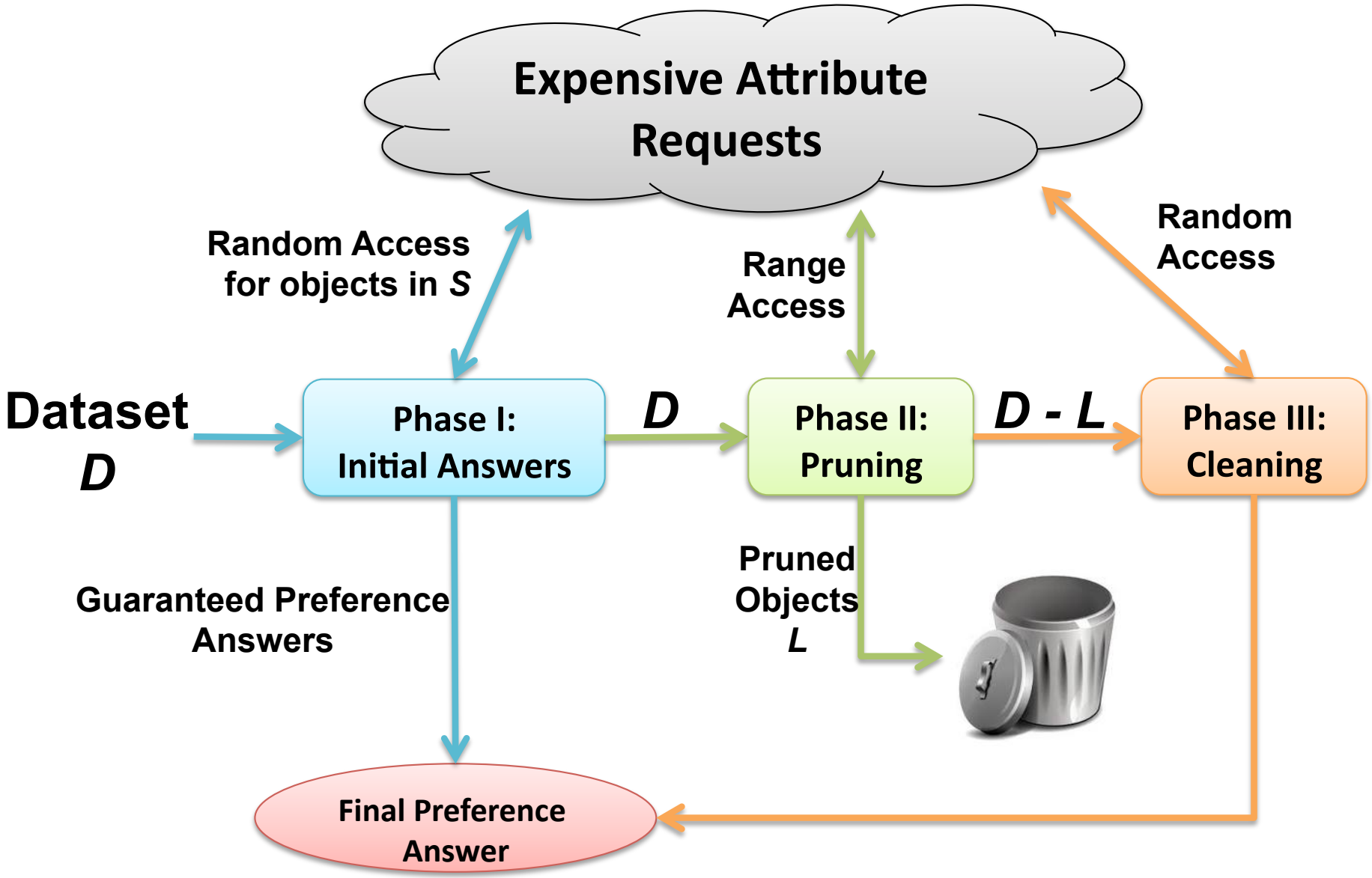
- You cannot “download” this data
- Data may change from query to query
- Legal reasons - from yelp API terms:



“You may not **cache**, **store**, **analyze** or **otherwise use** Yelp content except for real-time use.”

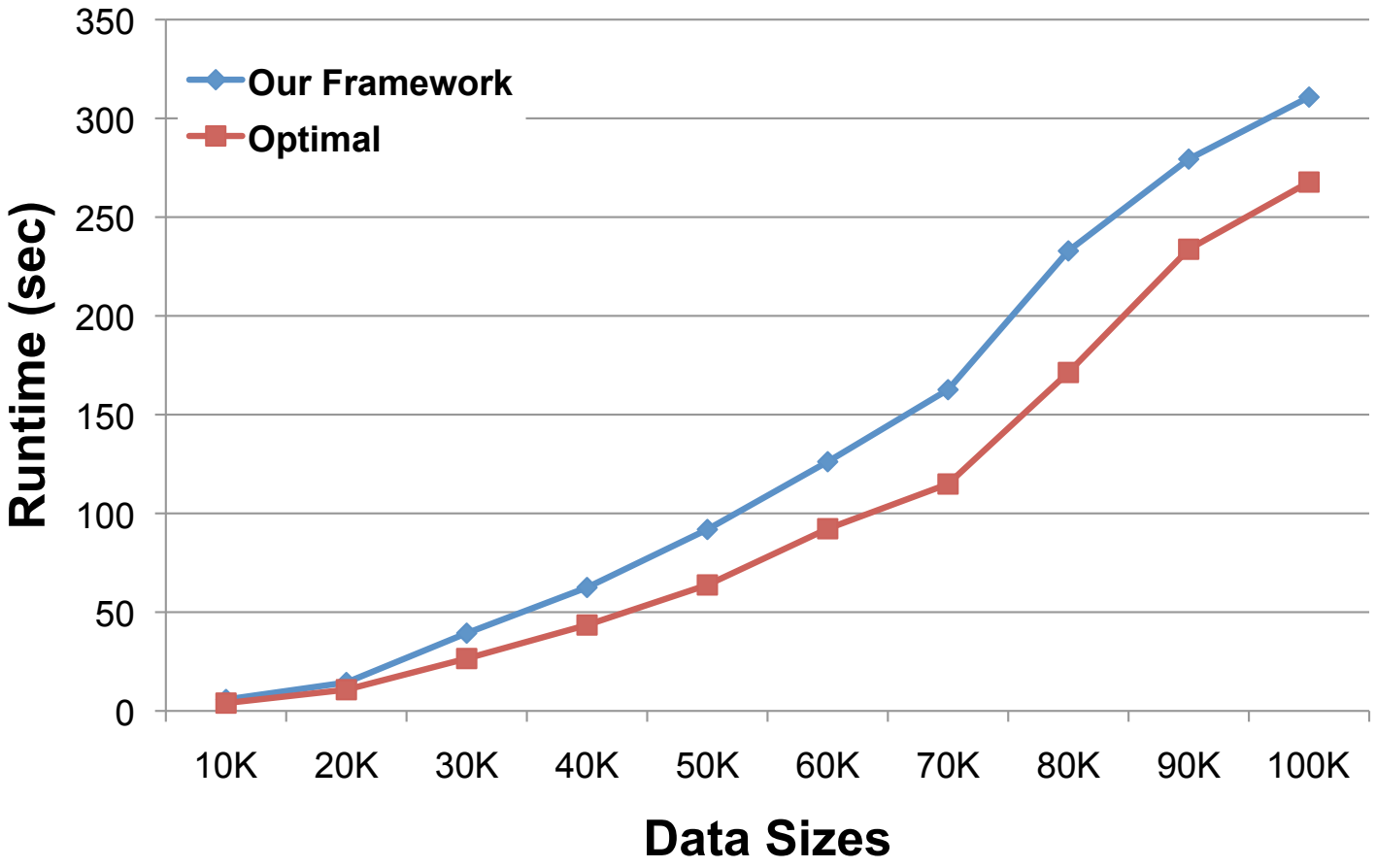
http://www.yelp.com/developers/documentation/faq

Preference Query Processing Over Expensive Attributes



"Preference Query Evaluation over Expensive Attributes", CIKM 2010

Multi-Objective Query Experiment



Future Research Directions – Long Term

- **Geo-social systems (Application)**
  - Build a scalable framework to share and analyze user-generated and geo-tagged multimedia data in social networking environments

- **Real-time recommender systems (Application)**
  - Adapt existing, high-quality recommendation techniques to new dynamic environments where “recommendations” change rapidly (e.g., social network data, online news/blog posts).

- **Data management anywhere (Platform)**
  - Explore system architectures to integrate and perform query processing over existing and emerging disparate data sources (e.g., web services, Amazon Mechanical Turk, local data).