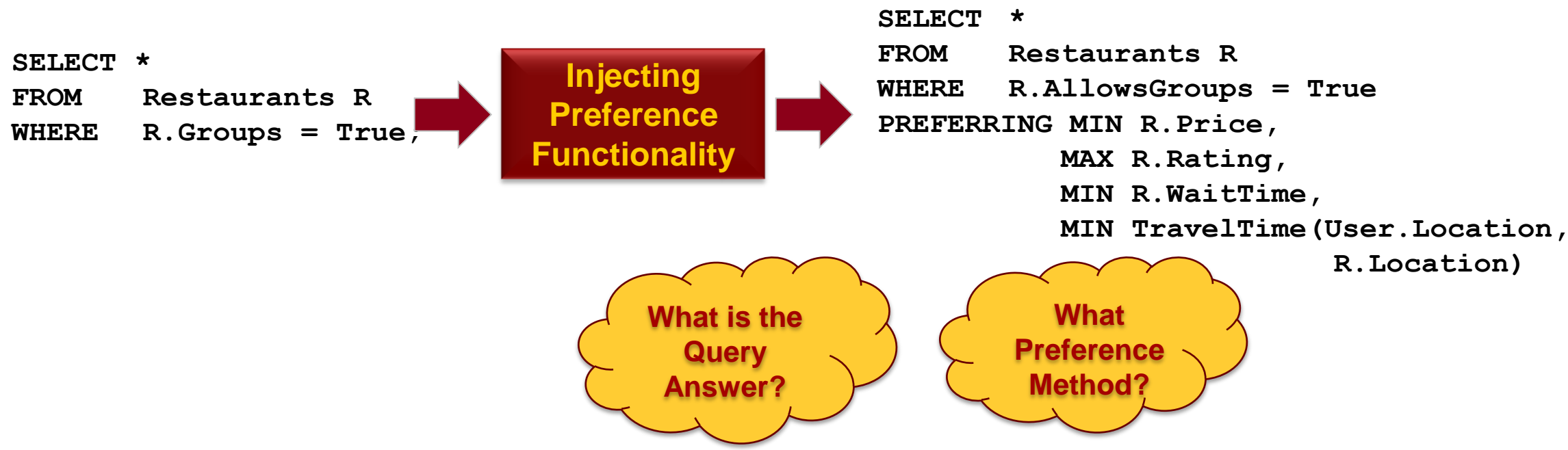# A Demonstration of FlexPref:
Extensible Preference Evaluation Inside the DBMS Engine

Justin J. Levandoski   **Mohamed F. Mokbel**   **Mohamed E. Khalefa**   **Venkateshwar R. Korukanti**

University of Minnesota Department of Computer Science and Engineering

Preference Methods

Need for Preference Funcationality in the DBMS

```
SELECT *
FROM    Restaurants R
WHERE   R.Groups = True,
```

→ Injecting 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)
```

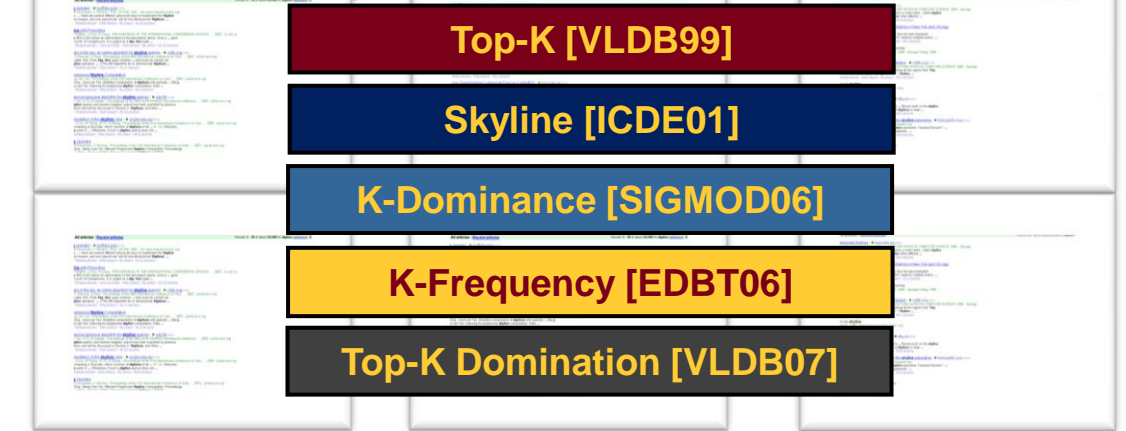*What is the Query Answer?*   *What Preference Method?*

Many Multi-Objective Preference Methods

Quick Exercise
1.   Go to Google Scholar
2.   Search for papers on preference evaluation methods
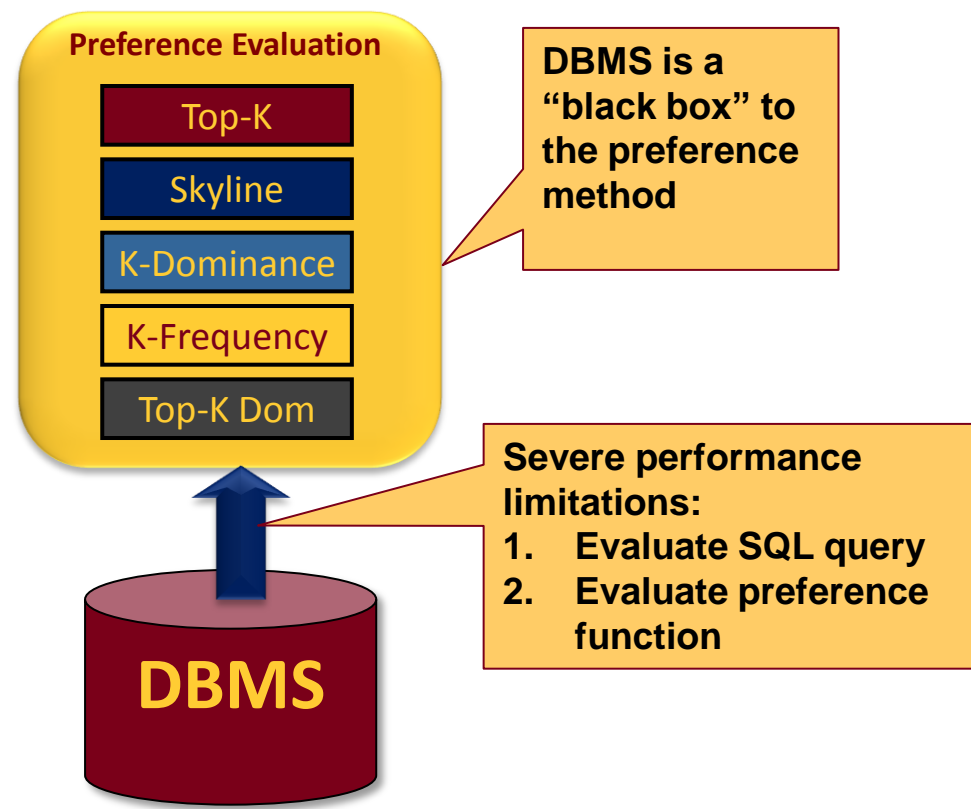3.   How many results do you get back?
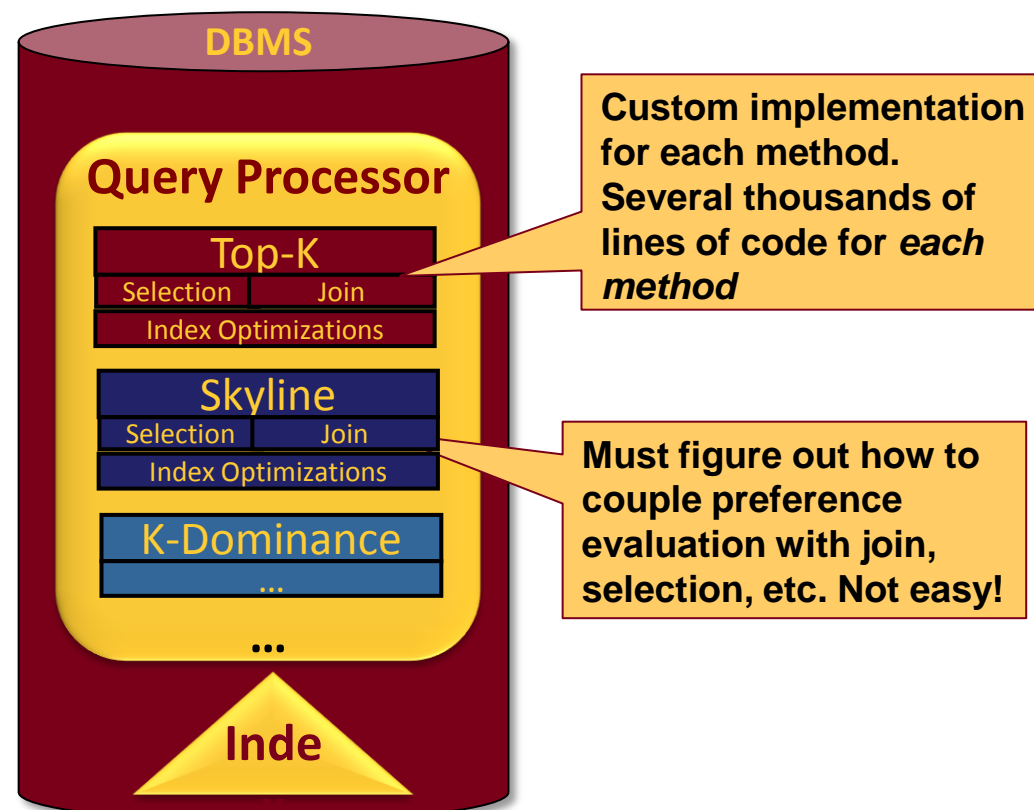
The list goes on and on and on...

| Top-K [VLDB99] |
| Skyline [ICDE01] |
| K-Dominance [SIGMOD06] |
| K-Frequency [EDBT06] |
| Top-K Domination [VLDB07] |

DBMS Implementation Approaches

Existing Approaches

The Layered Approach

Preference Evaluation: Top-K, Skyline, K-Dominance, K-Frequency, Top-K Dom

DBMS is a "black box" to the preference method

DBMS

Severe performance limitations:
1.   Evaluate SQL query
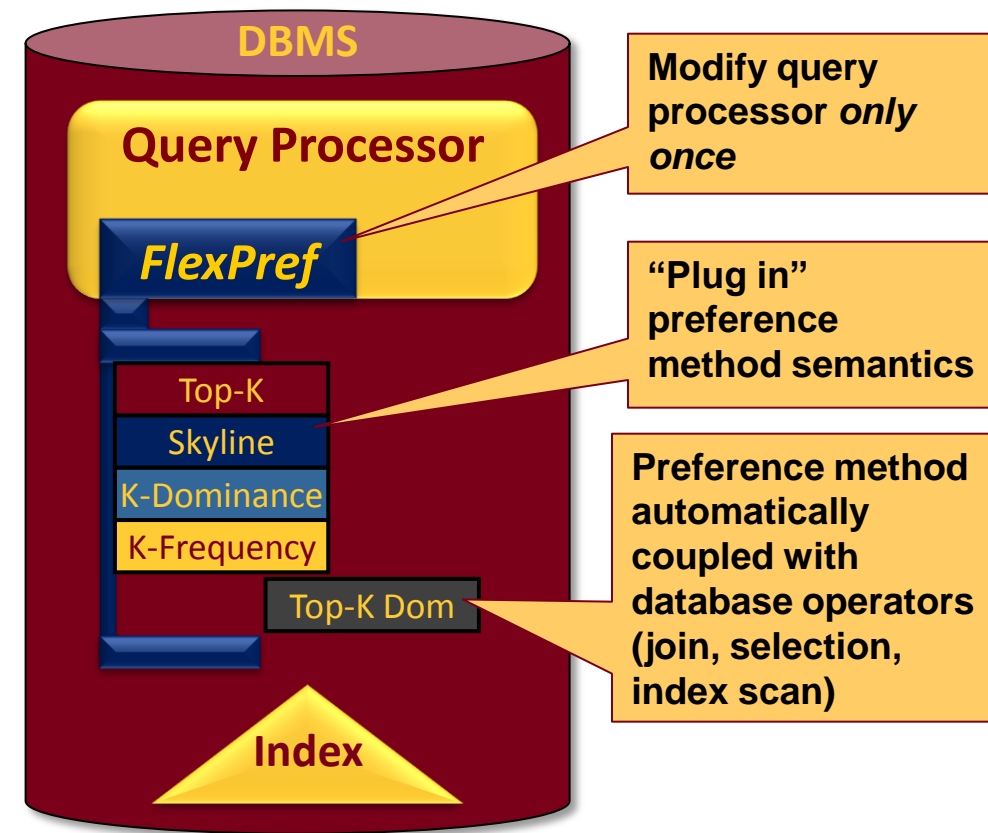2.   Evaluate preference function

✔ **Simplicity**: easy to implement

✘ **Limited Efficiency**: cannot interact with DBMS internals; no query optimization

The Built-In Approach

DBMS
Query Processor
Top-K: Selection, Join, Index Optimizations
Skyline: Selection, Join, Index Optimizations
K-Dominance ...
Inde

Custom implementation for each method. Several thousands of lines of code for *each method*

Must figure out how to couple preference evaluation with join, selection, etc. Not easy!

✔ **Efficient**: methods tightly coupled with DBMS

✘ **Infeasible**: cannot provide custom Implementation for *every* preference method

The FlexPref Approach

FlexPref Architecture

DBMS
Query Processor
FlexPref
Top-K
Skyline
K-Dominance
K-Frequency
Top-K Dom
Index

Modify query processor *only once*

"Plug in" preference method semantics

Preference method automatically coupled with database operators (join, selection, index scan)

✔ **Simplicity of the layered approach**: easy to implement

✔ **Efficiency of the built-in approach**: methods tightly coupled with DBMS
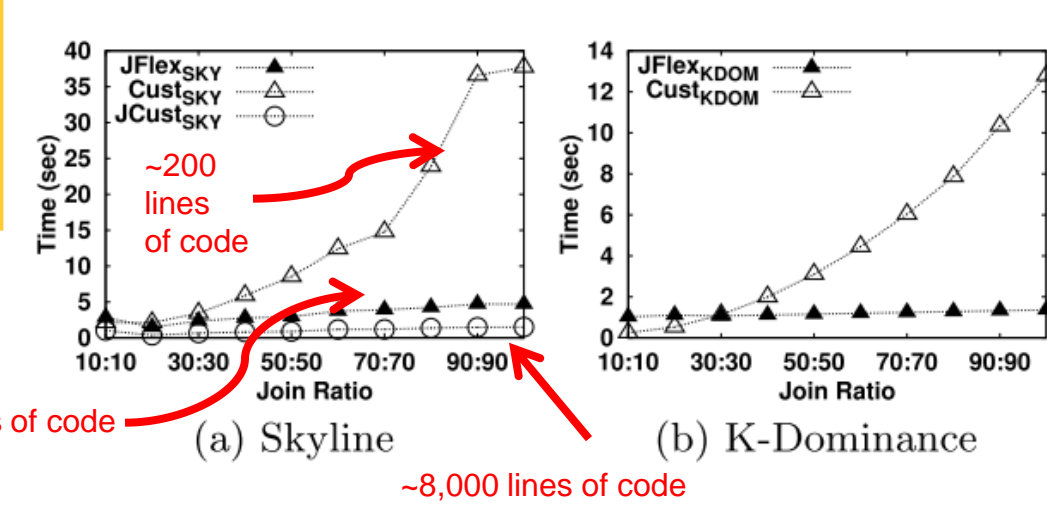
Writing Queries

```
SELECT *
FROM Restaurants R
WHERE [Where_clause]
PREFERENCE OF [Attribute/Objectives]
WITH PREF METHOD MyPref
```
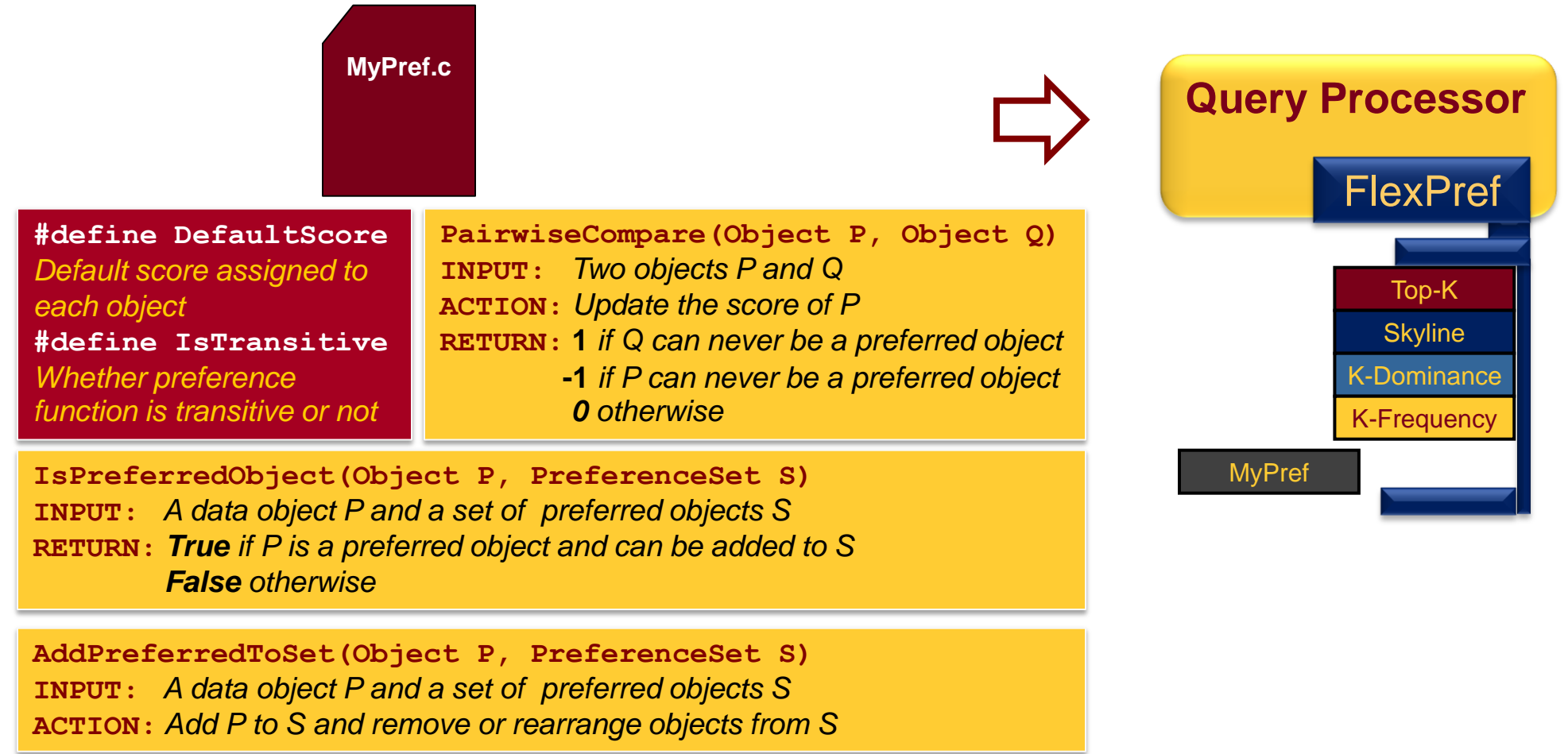
Experimental Analysis

Preference evaluation with binary join
FlexPref Join v. Built-in Selection v. Built-in Join

~200 lines of code
~300 lines of code
~8,000 lines of code

(a) Skyline   (b) K-Dominance

Preference Implementation

1.   Define *two macros* and *three functions* in separate file outside DBMS/FlexPref

2. Compile into FlexPref using command:
`DefinePreference MyPref with MyPref.c`

MyPref.c

Query Processor
FlexPref
Top-K
Skyline
K-Dominance
K-Frequency
MyPref

```
#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
```
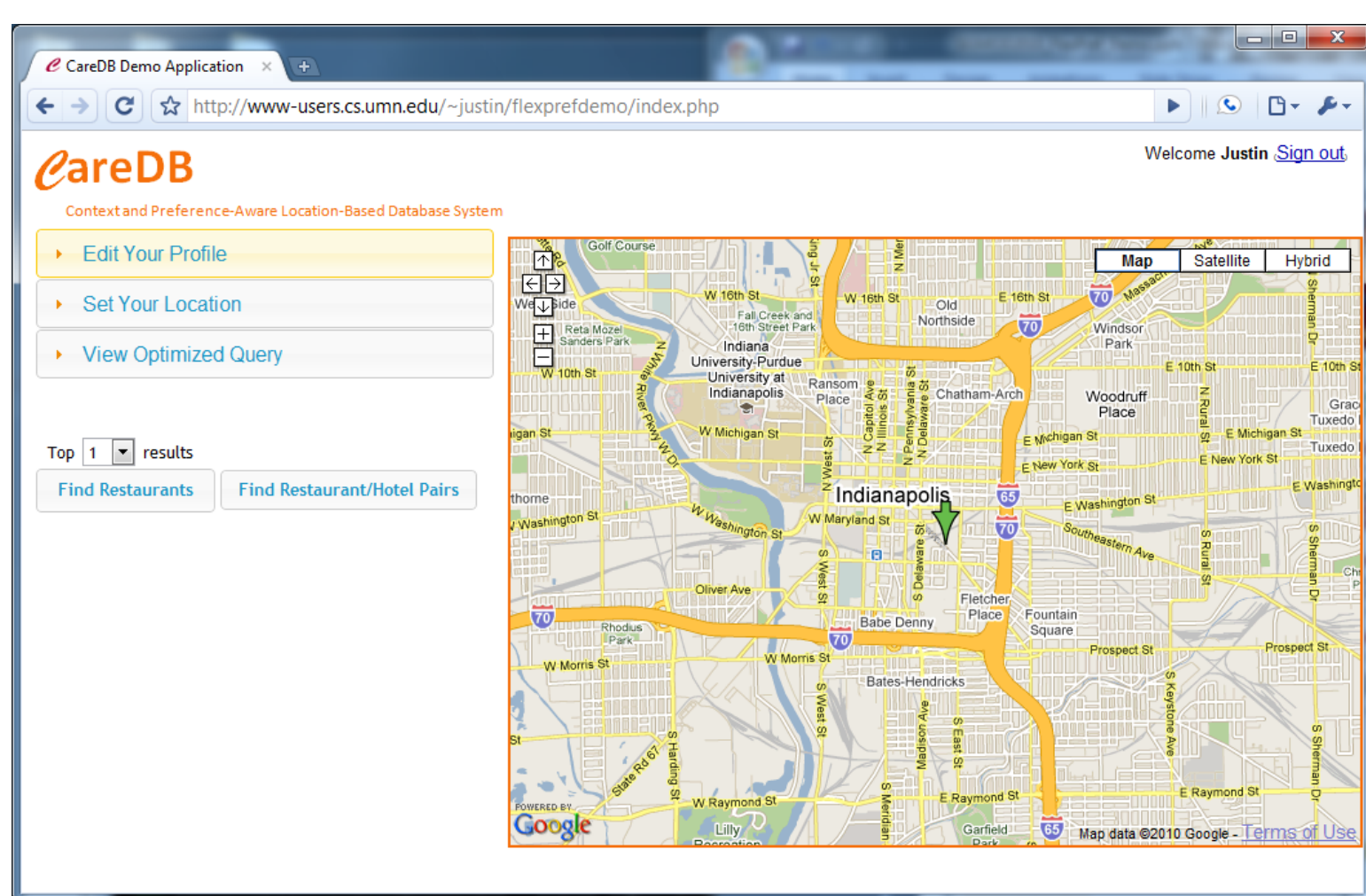
Query Processing

FlexPref offers generic query processing support for the following operators, each operator written in generic fashion with reference to the specific macros and functions for specific preference method semantics

```
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
```

Answer
FlexPref Selection
Restaurants

Answer
FlexPref Join
Restaurants   Hotels

FlexPref in Action

Environment

o   **Web-based restaurant and hotel finder application implemented in Google Maps with real Indianapolis restaurant/hotel data**

o   **Data stored in underlying PostgreSQL database.**

o   **FlexPref implemented inside query processor of PostgreSQL.**

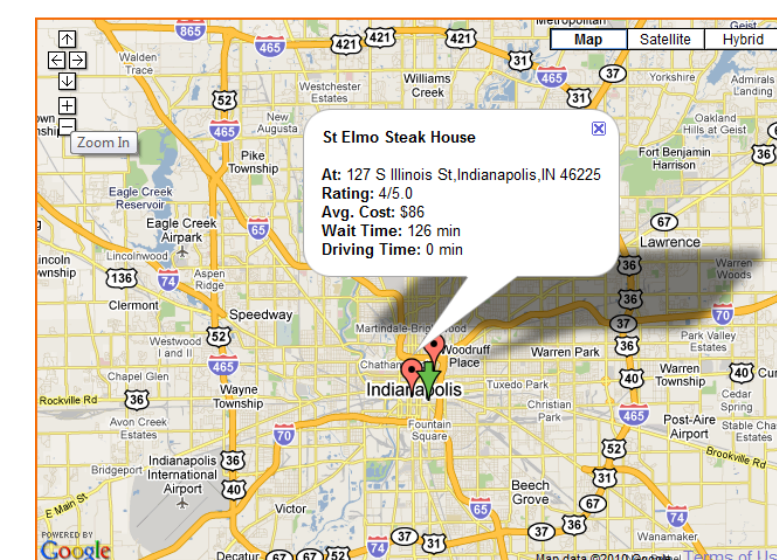### Define Preference Method
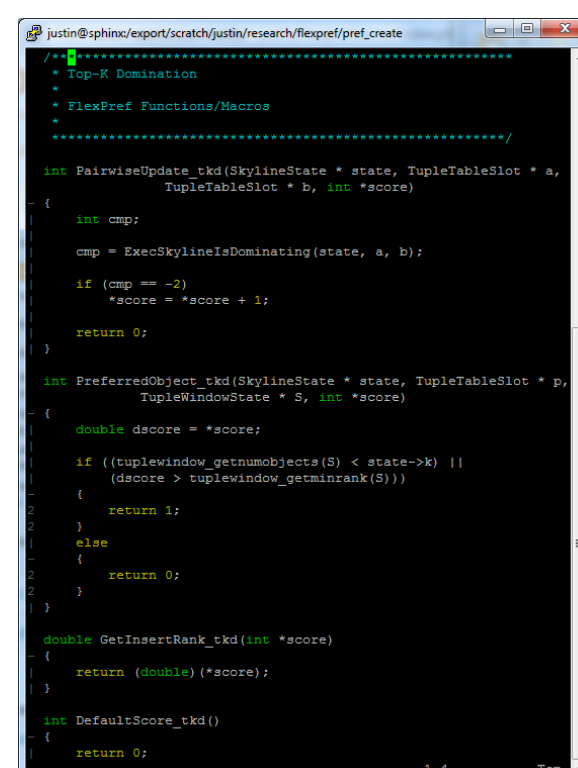
o   **We demonstrate how to implement the Top-K Domination preference method using FlexPref functions and macros**

o   **Functions implemented in file *outside* DBMS codebase and *compiled* into FlexPref**
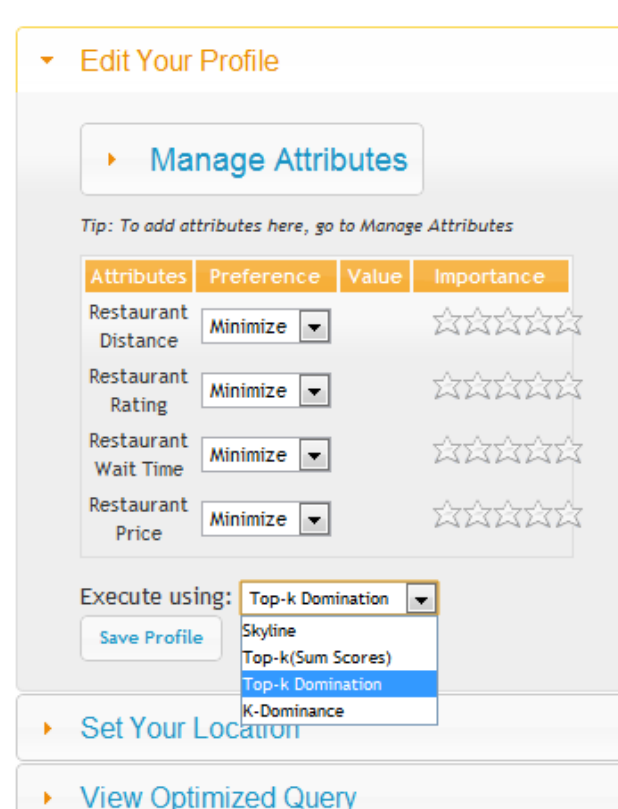
o   **Once compiled, preference method is available for use in defining preference profile in web-based application**
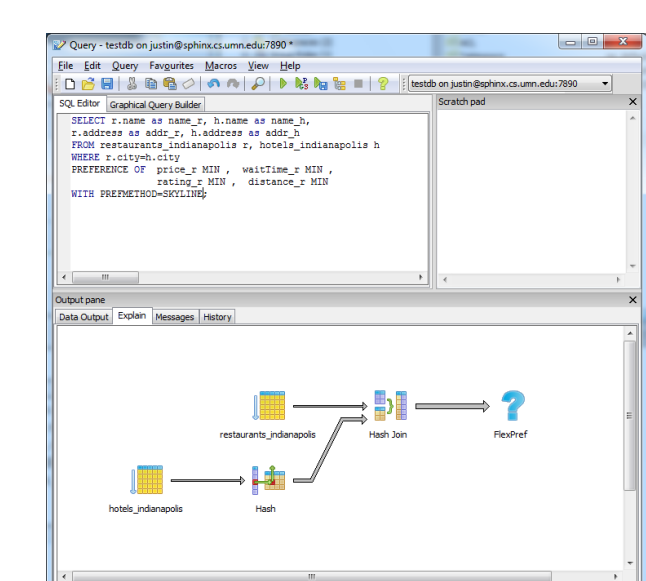
Querying with FlexPref

o   **Users press "find restaurant/hotel" button.**
o   **Answers displayed on Google Maps.**
o   **Answers generated by FlexPref executing in PostgreSQL using chosen preference method**
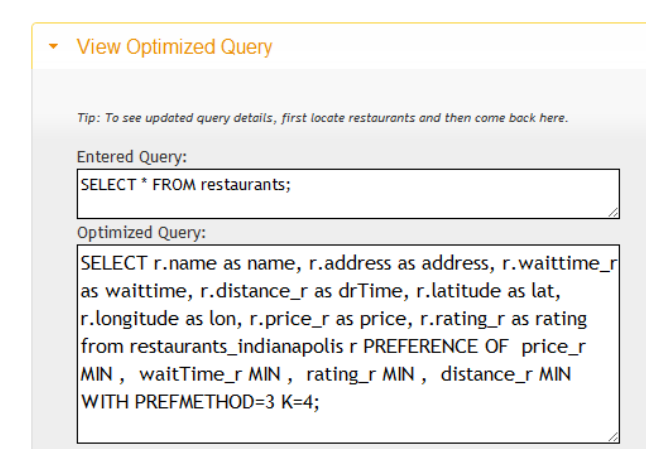
o   **Users can view SQL query extended with preference syntax that was executed in underlying PostgreSQL database**

o   **Users can also issue ad-hoc FlexPref queries, and view query plans, using graphical client connected to PostgreSQL**