Human-powered Sorts and Joins

Adam Marcus, Eugene Wu, David Karger, Samuel Madden, Robert Miller
Presenter: Xiaoyi Fan

2016. 10. 12

Outline

- Introduction
- Overview
- Join Operator
- Sort Operator
- Conclusion

Overview of Qurk

- Declarative “workflow” system integrating human intelligence
  - User-defined functions
  - Human Intelligence Tasks (HIT)
  - Capture operations outside relational algebra
  - Typically external API calls

Overview of Qurk

Query Model:

SQL

Overview of Qurk

```
Query =SELECT * FROM photos WHERE
        isFemale(photos.picture);
```

UDF

Overview of Qurk

- Instead of writing code for UDFs, can be described at a high level using Tasks
- Tasks = High level-templates for commonly occurring crowd-operations and or algorithms
- Filter, Generate, Sort, Joins

Overview of Qurk

- Generative Tasks

```
SELECT c.name
FROM celeb AS c
WHERE isFemale(c)
```

isFemale defined as follows:

```
TASK isFemale(field) TYPE Filter:
    Prompt: "<table><tr> \
       <td><img src='%s'></td> \
       <td>Is the person in the image a woman?</td> \
      </tr></table>", tuple[field]
     YesText: "Yes"
     NoText: "No"
     Combiner: MajorityVote
```

Overview of Qurk

- Interface

Yes No

Yes No

Overview of Qurk

- Join Operator

```
SELECT c.name
FROM celeb c JOIN photos p
ON samePerson(c.img,p.img)
```

```
TASK samePerson(f1, f2) TYPE EquiJoin:
    SingluarName: "celebrity"
    PluralName: "celebrities"
    LeftPreview: "<img src='%s' class=smImg>",tuple1[f1]
    LeftNormal: "<img src='%s' class=lgImg>",tuple1[f1]
    RightPreview: "<img src='%s' class=smImg>",tuple2[f2]
    RightNormal: "<img src='%s' class=lgImg>",tuple2[f2]
    Combiner: MajorityVote
```

Overview of Qurk

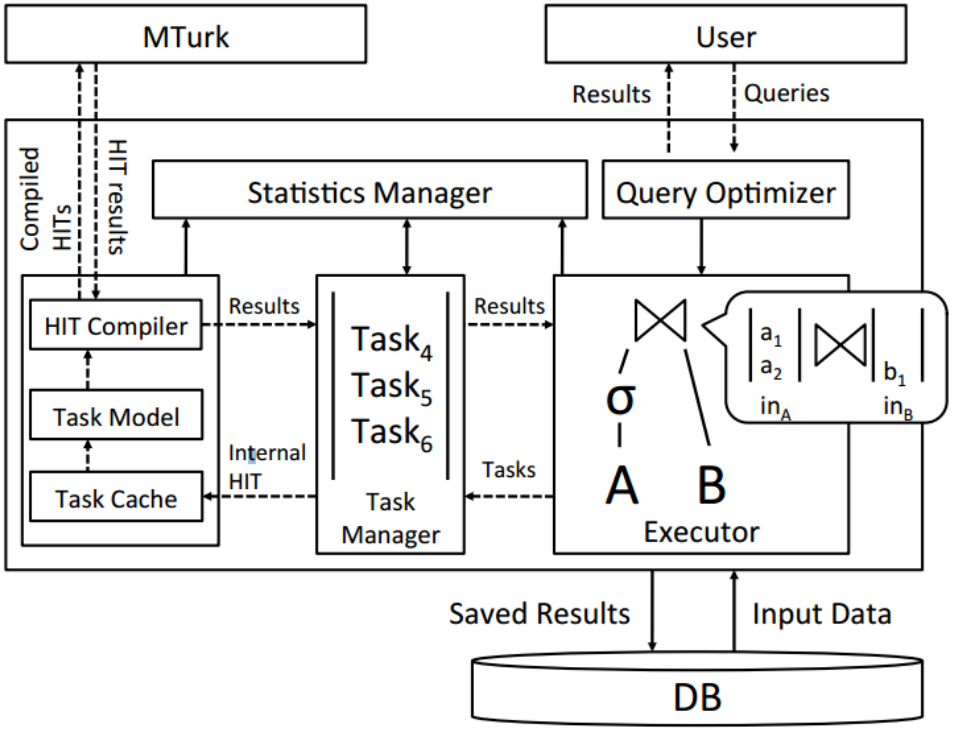
- Sort Operator

```
SELECT squares.label
FROM squares
ORDER BY squareSorter(img)
```

```
TASK squareSorter(field) TYPE Rank:
     SingularName: "square"
     PluralName: "squares"
     OrderDimensionName: "area"
     LeastName: "smallest"
     MostName: "largest"
     Html:  "<img src='%s' class=lgImg>",tuple[field]
```

Overview of Qurk

- Qurk system architecture

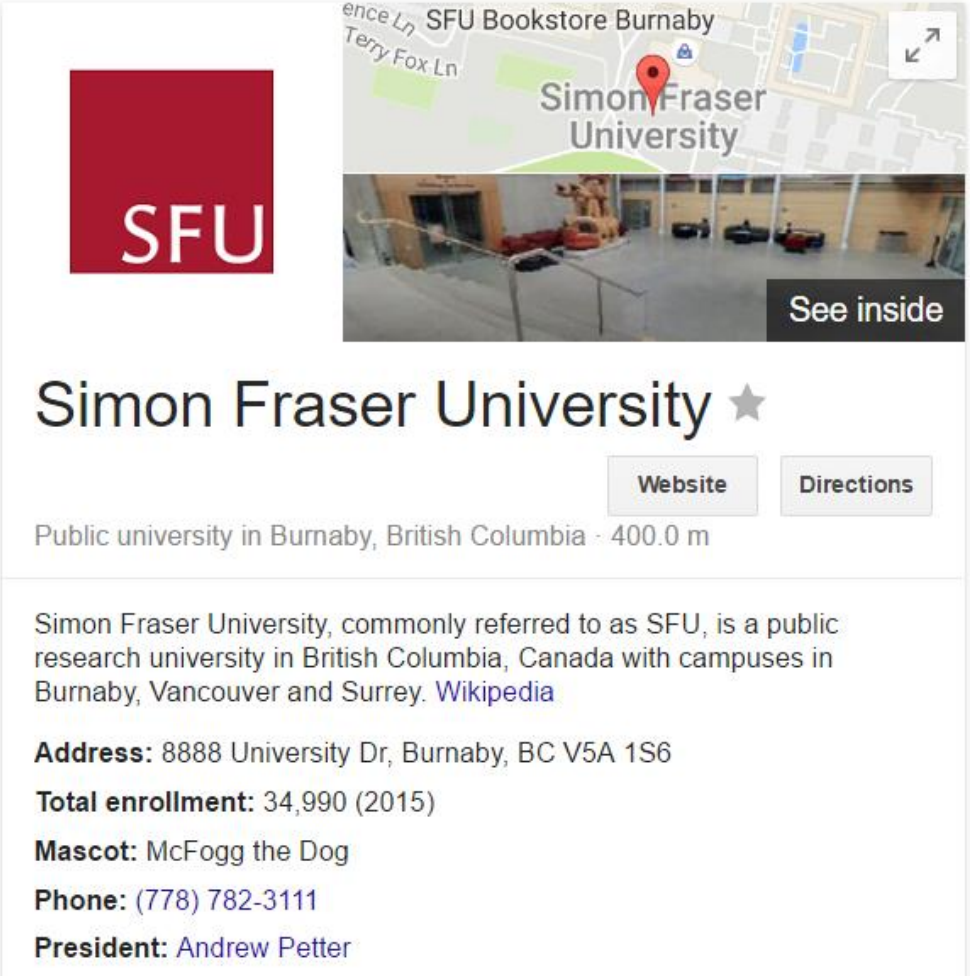


Join Operator

Simon Fraser University

==

SFU



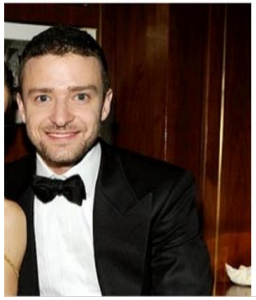
Join Operator

- Matching Celebrities







Join Operator

- Simple Join O(nm)

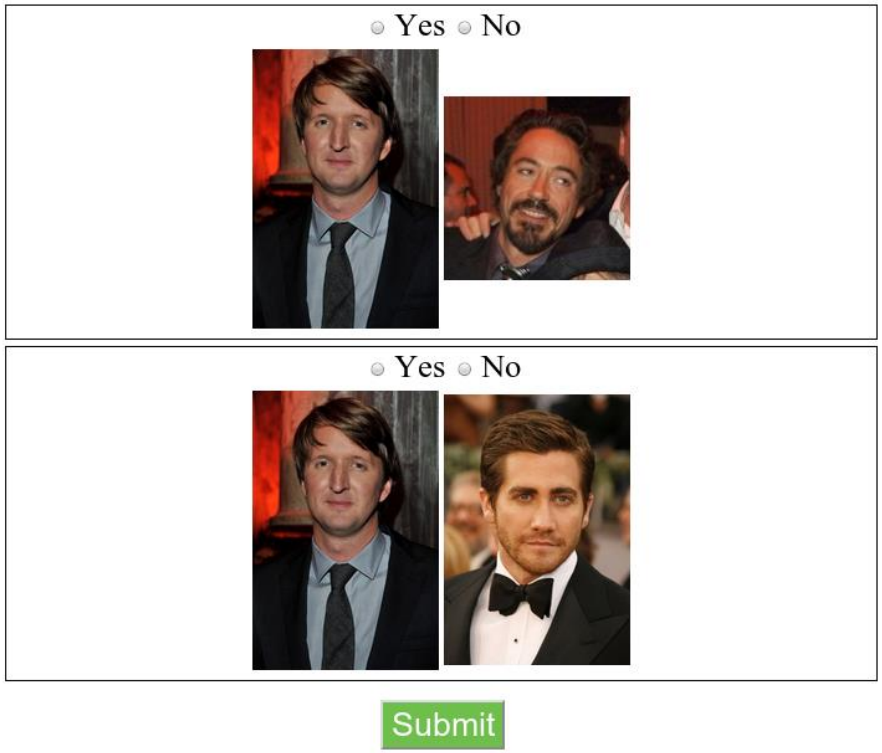
Is the same celebrity in the image on the left and the image on the right?

Yes No

Join Operator

- Naive Batching O(nm/b)

Is the same celebrity in the image on the left and the image on the right?



Join Operator

- Smart Batching $O(nm/b^2)$

Join Operator

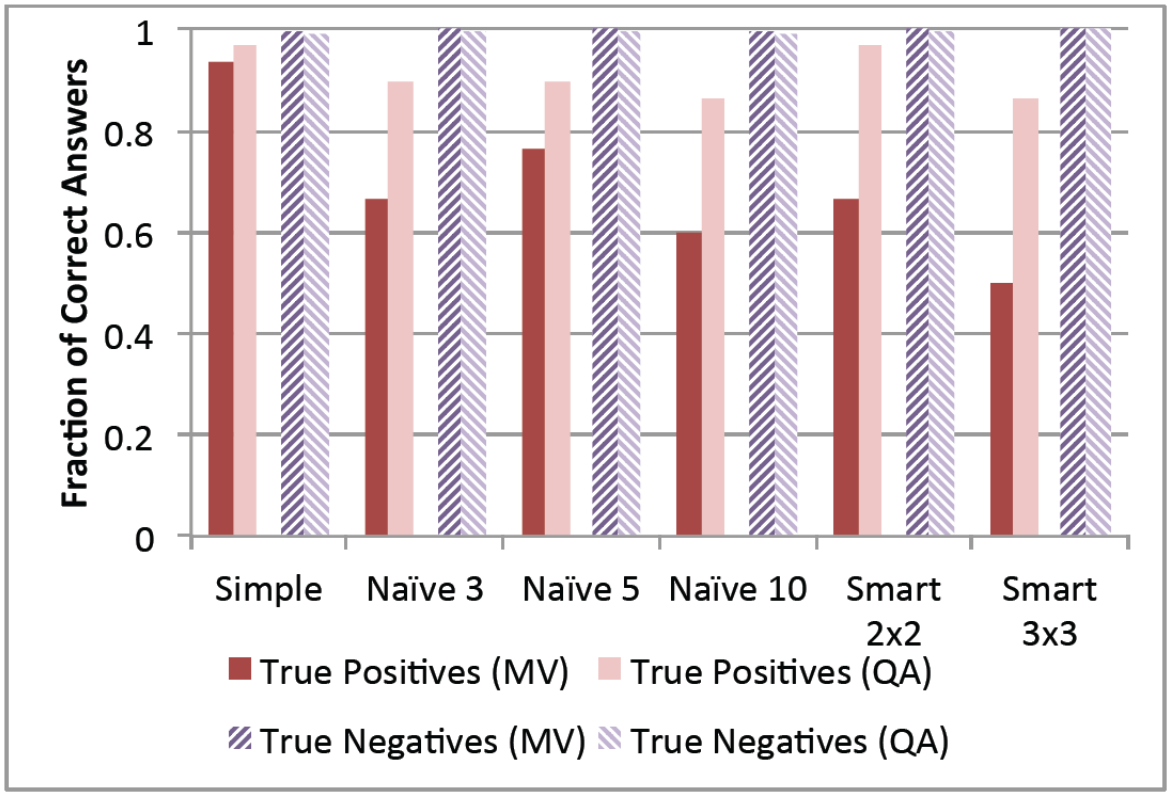
- Experiments setup:
  - Compare *IDEAL*, *Simple*, *Naïve*, and *Smart* schemes
  - Dataset: *celebrity join dataset*

Implementation	True Pos. (MV)	True Pos. (QA)	True Neg (MV)	True Neg (QA)
IDEAL	20	20	380	380
Simple	19	20	379	376
Naive	19	19	380	379
Smart	20	20	380	379

Baseline comparison of three join algorithms with no batching enabled

Join Operator

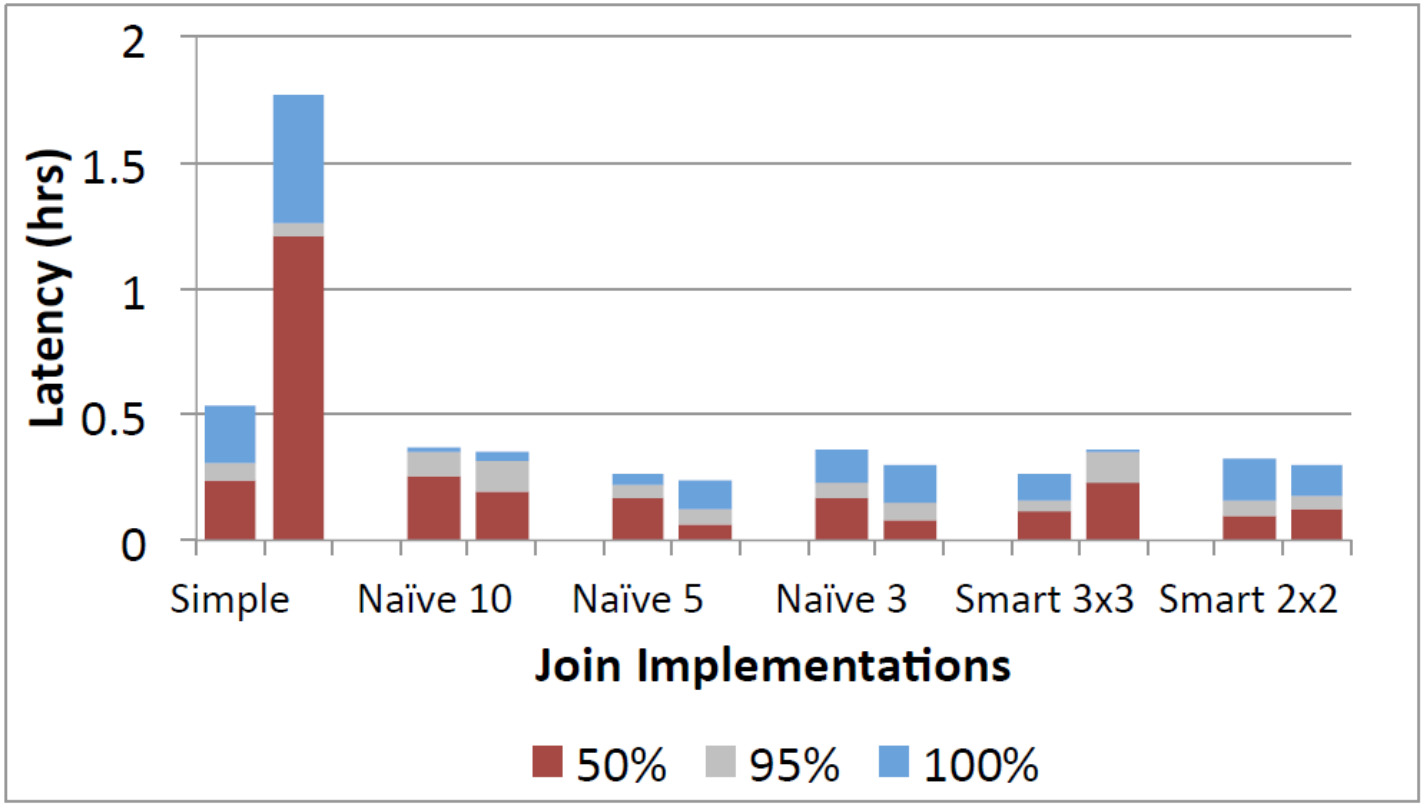
- Experiments



Fraction of correct answers on celebrity join for different batching approaches

Join Operator

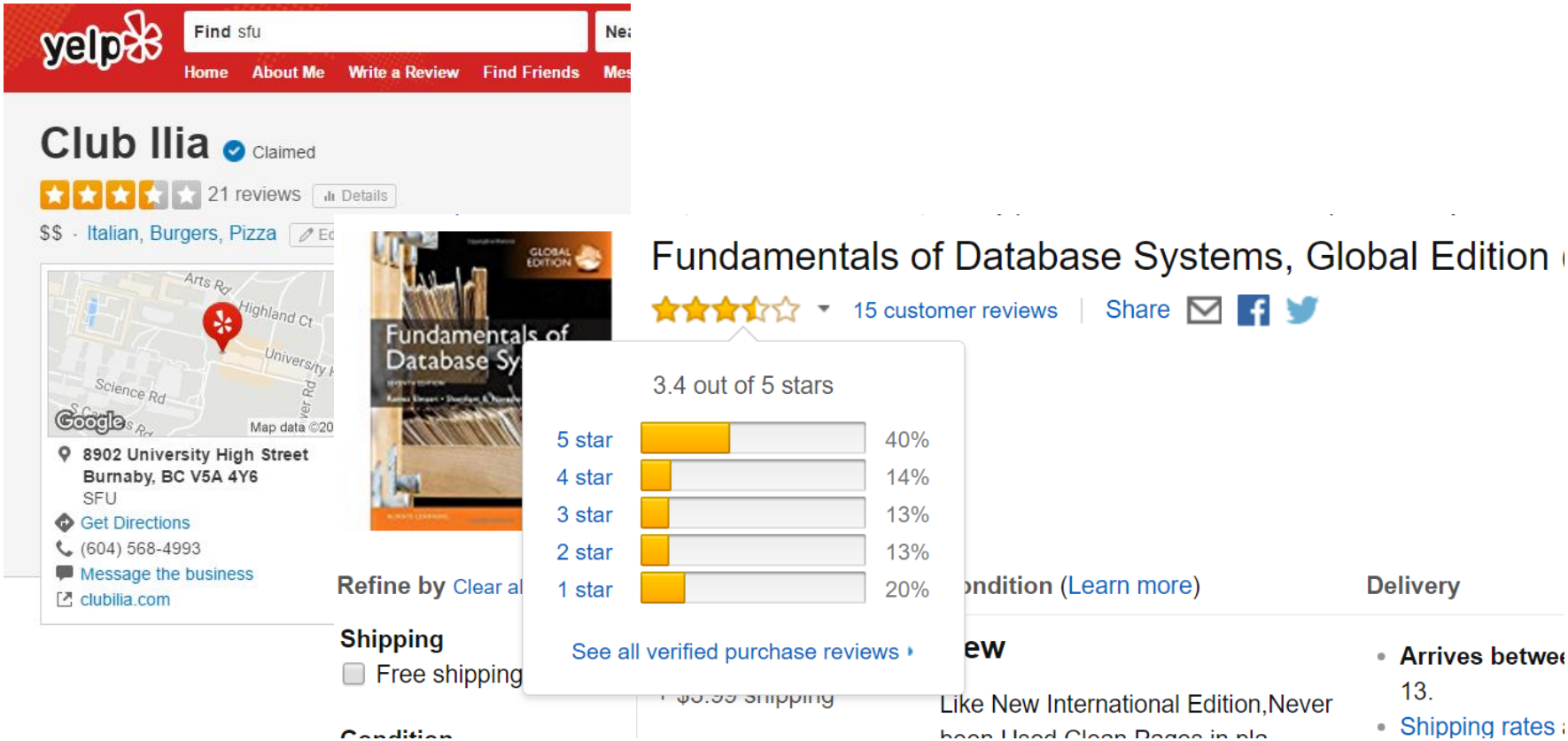
- Experiments



Completion time in hours for variants of celebrity join on two tables

Sort Operator

- Sort application with human intelligence

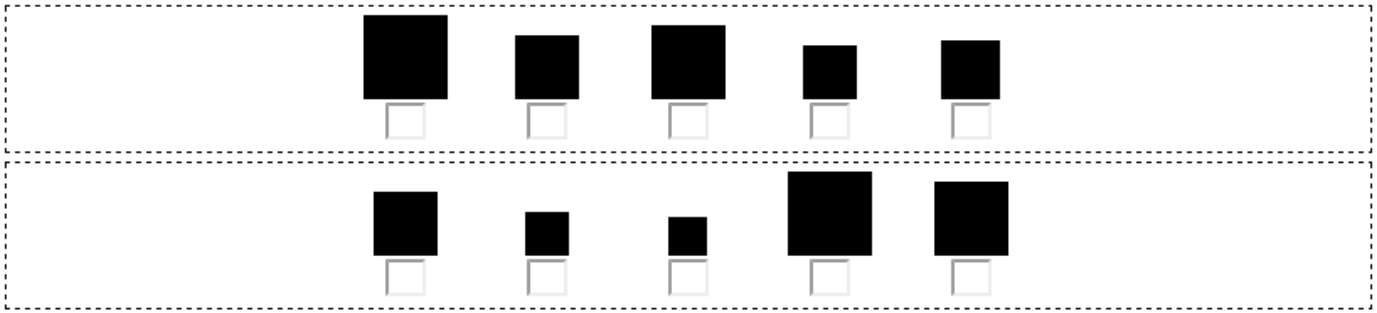


Sort Operator

- Comparison Sort

There are 2 groups of squares. We want to order the squares in each group from smallest to largest.

- Each group is surrounded by a dotted line. Only compare the squares within a group.
- Within each group, assign a number from 1 to 7 to each square, so that:
  - 1 represents the smallest square, and 7 represents the largest.
  - We do not care about the specific value of each square, only the relative order of the squares.
  - Some groups may have less than 7 squares. That is OK: use less than 7 numbers, and make sure they are ordered according to size.
  - If two squares in a group are the same size, you should assign them the same number.



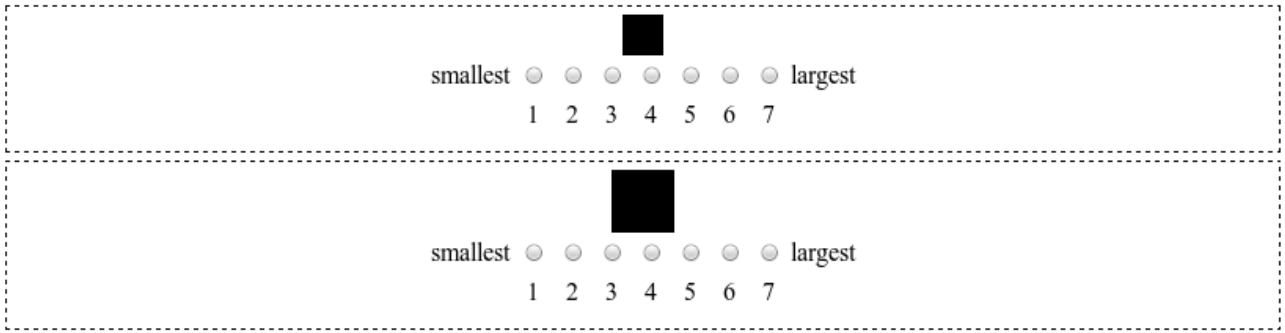
Submit

Sort Operator

- Rating Sort

There are 2 squares below. We want to rate squares by their size.

- For each square, assign it a number from 1 (smallest) to 7 (largest) indicating its size.
- For perspective, here is a small number of other randomly picked squares:



Submit

Hybrid Schemes

- Initially use the rating-based sort
- Use comparisons, in one of three flavors:
  - Random: pick S items, compare
  - Confidence-based: pick most confusing “window”, compare
  - Sliding-window: for all windows, compare

Sort Operator

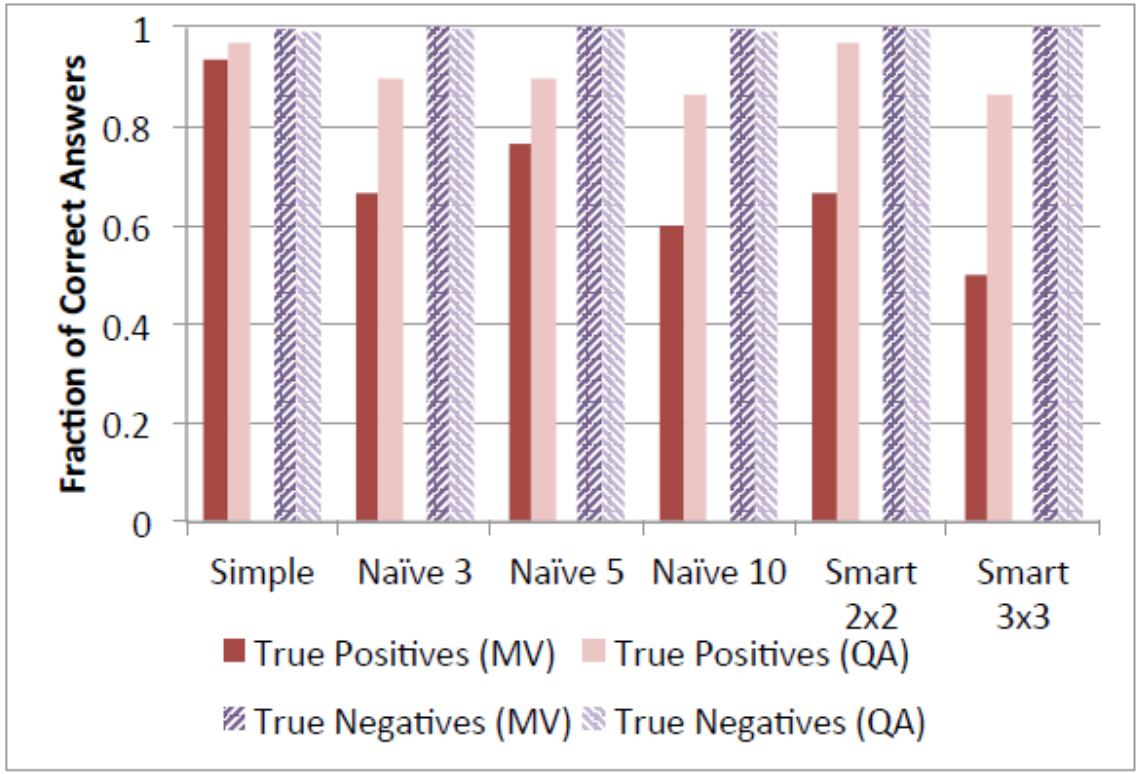
- Experiments setup:
  - Compare *IDEAL*, *Simple*, *Naive* and *Smart* schemes
  - Dataset: *celebrity join dataset*

Implementation	True Pos. (MV)	True Pos. (QA)	True Neg (MV)	True Neg (QA)
IDEAL	20	20	380	380
Simple	19	20	379	376
Naive	19	19	380	379
Smart	20	20	380	379

Baseline comparison of three join algorithms with no batching enabled

Sort Operator

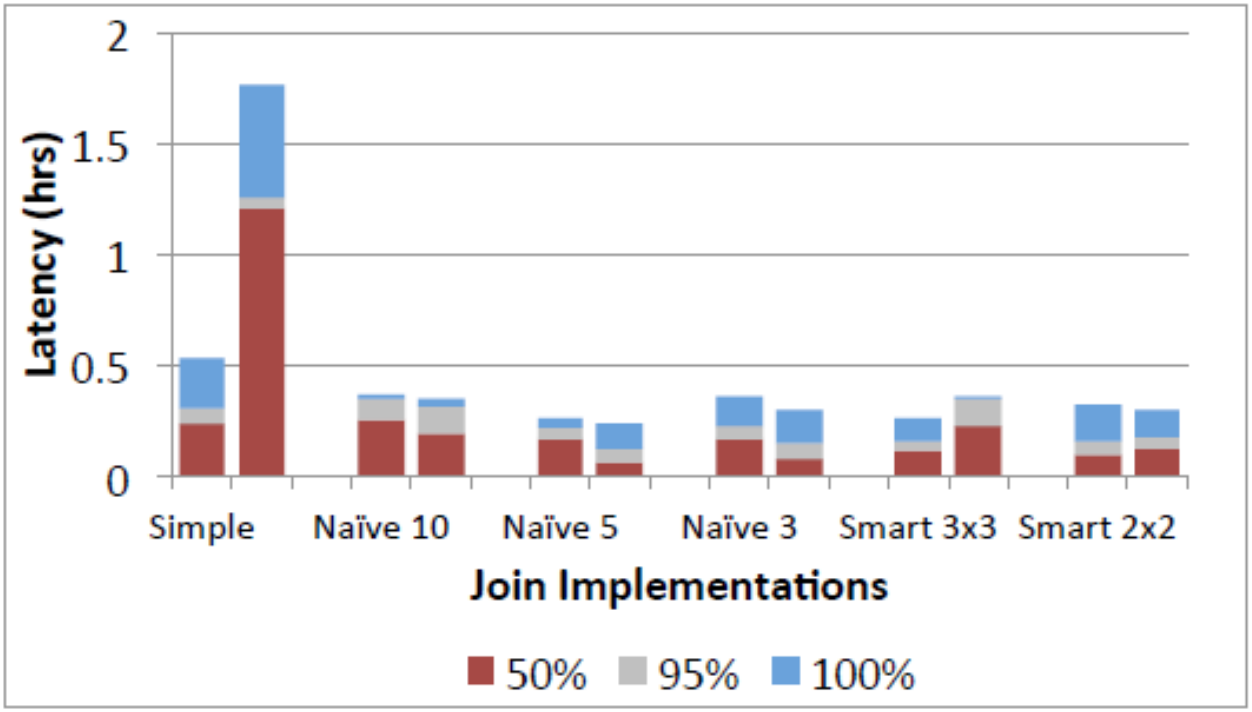
- Experiments



Fraction of correct answers on celebrity join for different batching approaches

Sort Operator

- Experiments



Completion time in hours for variants of celebrity join on two tables

Conclusion

- A declarative workflow engine, Qurk
- Integrated crowd intelligence
- Sorting and joining operation
- Optimization in task batching
- Reduce the overall cost from $67 to $3

Questions and Comments?

Thank you!