S4: Top-k Spreadsheet Style Search for Query Discovery

Fotis Psallidas — Columbia University, New York, NY — fotis@cs.columbia.edu
Bolin Ding, Kaushik Chakrabarti, Surajit Chaudhuri — Microsoft Research, Redmond, WA — bolind, kaushik, surajitc@microsoft.com

Background

An information worker

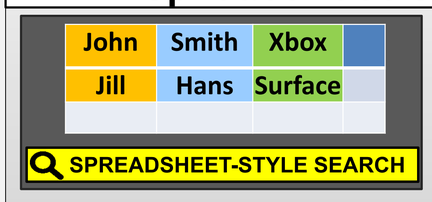
- Has **no exact knowledge** of the database and its underlying schema
- Seeks queries to cover information needs
- **Spends a lot of time** to go over the database schema and manually discover the queries in need

▪ **… hundreds to thousands of tables**
▪ **… tens to thousands of columns per table**
▪ **… and numerous relations**

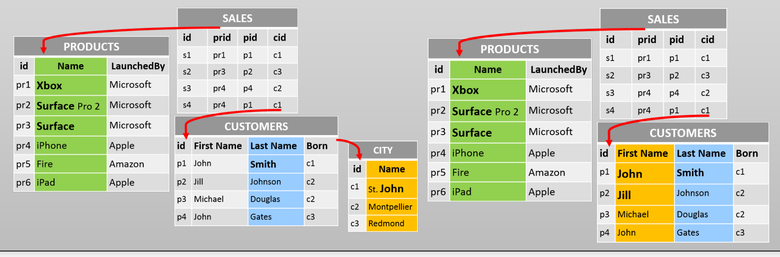
Q: How to help the information worker discover the queries of interest?

Observation: The information worker **knows a few example tuples** that should be present in the output of the queries.

Spreadsheet-Style Search



- Enter example tuples that should be in the output of the desired PJ query
- The system replies with the set of PJ queries relevant to the example tuples

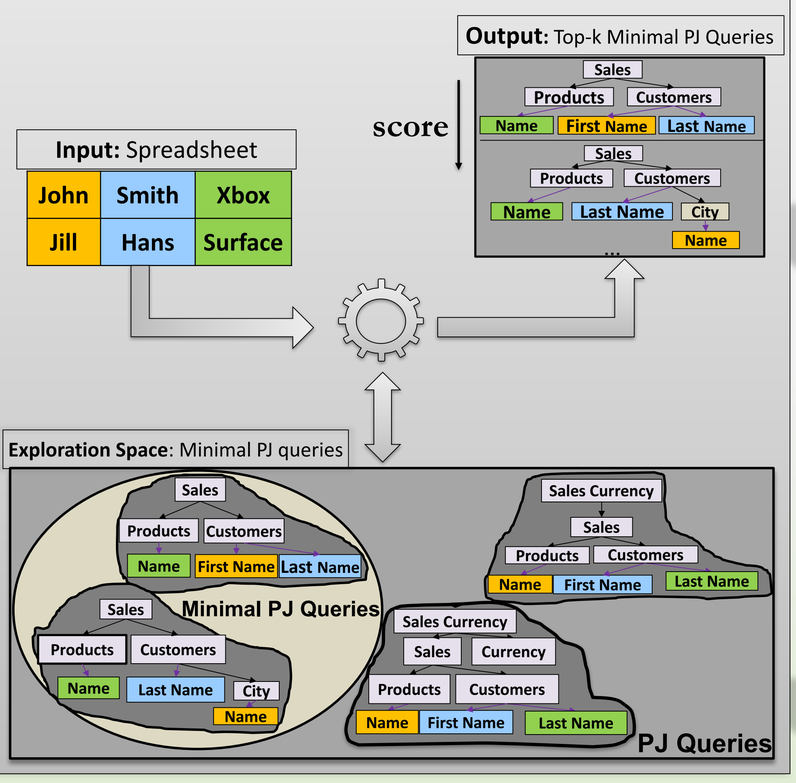


Contributions

- Introducing **top-k spreadsheet style search** and **a novel scoring model** for error-tolerant but efficient PJ query discovery
- Efficiently score the PJ queries using a novel **Enumeration-Evaluation** framework
- Focusing on evaluation using the framework:
  - We introduced **a strong baseline** that evaluates the minimum number of minimal PJ queries to **terminate early**
  - Bypassed the baseline by **sharing sub-PJ queries and terminating early**
  - Introduced optimizations to further increase speedup.
- Extensions of our model and approaches include:
  - **Incremental computation** of the output as soon as the user types in a cell
  - **OR semantics** to allow partial mapping of columns

System Task

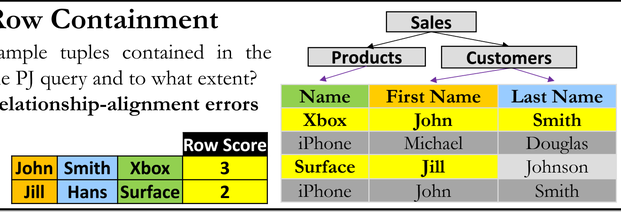
Given a spreadsheet filled with example tuples return the top-k most relevant minimal PJ-queries.



Scoring Model // What makes a good score?

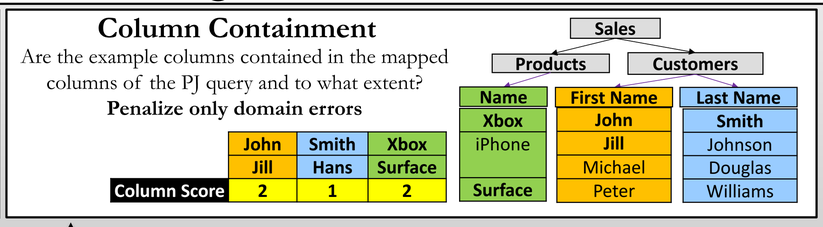
Row Containment

Are the example tuples contained in the output of the PJ query and to what extent?
Penalize relationship-alignment errors



Column Containment

Are the example columns contained in the mapped columns of the PJ query and to what extent?
Penalize only domain errors



Balancing Factor α

- **α=1**: all example tuples exactly contained in the output of the PJ query (strict version)
- **α=0**: no alignment requirements. Ranking based on column containment only

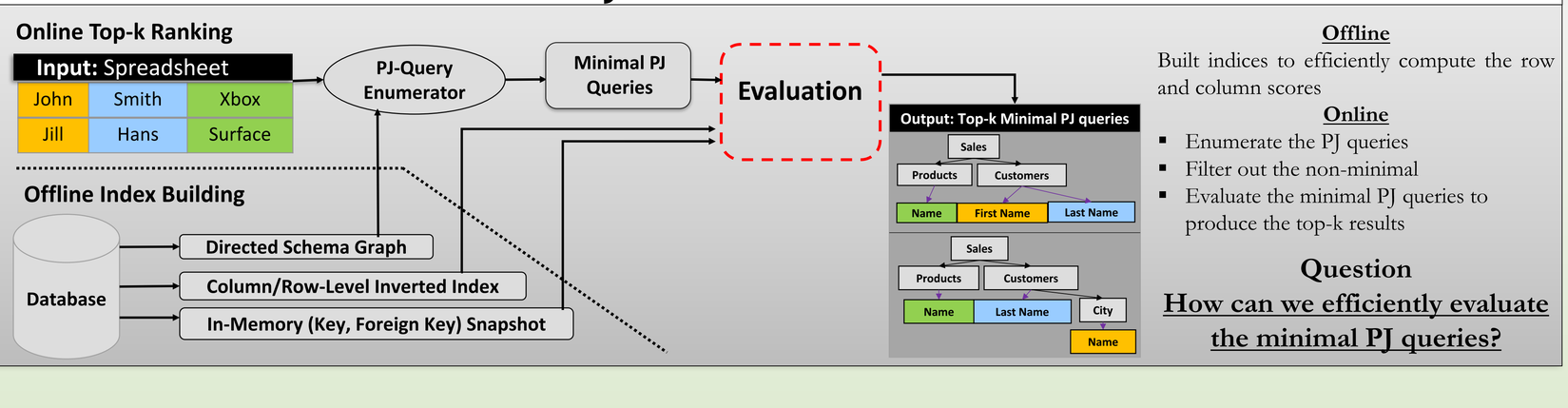
$$a * \text{score}_{row}\left(\text{Sales: Products, Customers; Name, First Name, Last Name} \mid \text{John Smith Xbox; Jill Hans Surface}\right) + (1-a) * \text{score}_{col}\left(\text{Sales: Products, Customers; Name, First Name, Last Name} \mid \text{John Smith Xbox; Jill Hans Surface}\right)$$

$$\text{Cost}\left(\text{Sales: Products, Customers}\right)$$

Cost-based tie breaker

- Two PJ-queries with the same score form a tie we need to break.
- Examples of cost functions include:
  - Height, #nodes, and #edges of the tree

System Architecture



Offline

Built indices to efficiently compute the row and column scores

Online

- Enumerate the PJ queries
- Filter out the non-minimal
- Evaluate the minimal PJ queries to produce the top-k results

Question

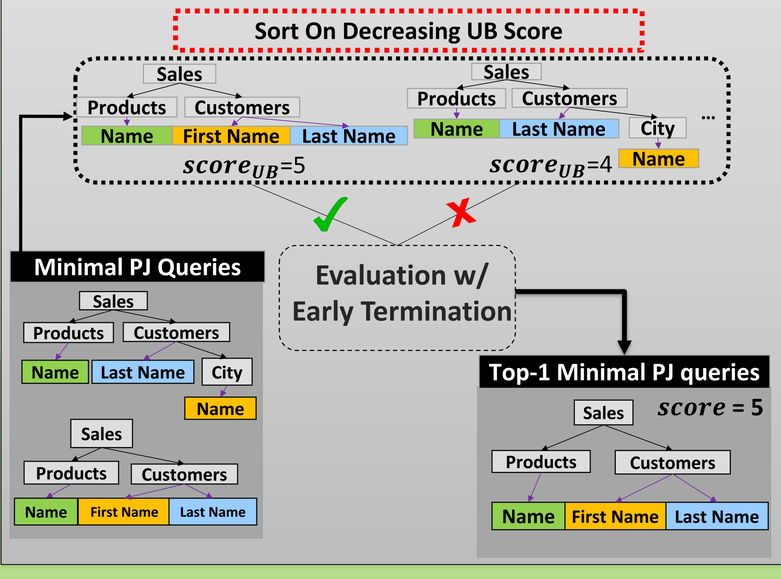
How can we efficiently evaluate the minimal PJ queries?

Evaluation w/ Early Termination

- Theoretical result: The score of a PJ query given a spreadsheet is bounded by the column score
- Column score efficiently computed using the indices built offline!
- **Idea**: Use column score as the upper bound score

Strategy

- **Evaluate PJ queries in the decreasing order of upper bound score**
- **Terminate when current upper bound score less than the k-th ranked evaluated PJ query**
- **Guarantees** minimal evaluation set of PJ queries



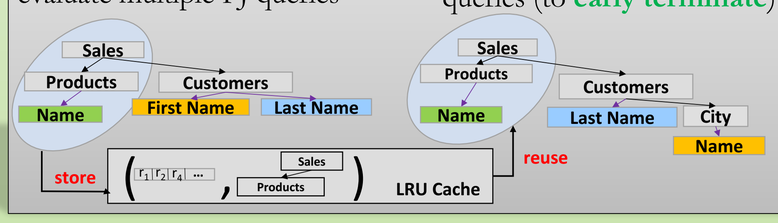
Sharing Sub-PJ Queries

Idea

Reuse previously computed intermediate results for Sub-PJ queries to efficiently evaluate multiple PJ queries

But

- Cache is limited
- We want to evaluate the minimum amount of PJ queries (to **early terminate**)



Caching-Evaluation Scheduling Problem

The two extremes of PJ queries evaluation

One at a time	All at once
▪ Cannot share Sub-PJ queries ▪ Can find the minimal evaluation set	▪ Maximum share of Sub-PJ queries ▪ Far from the minimal evaluation set

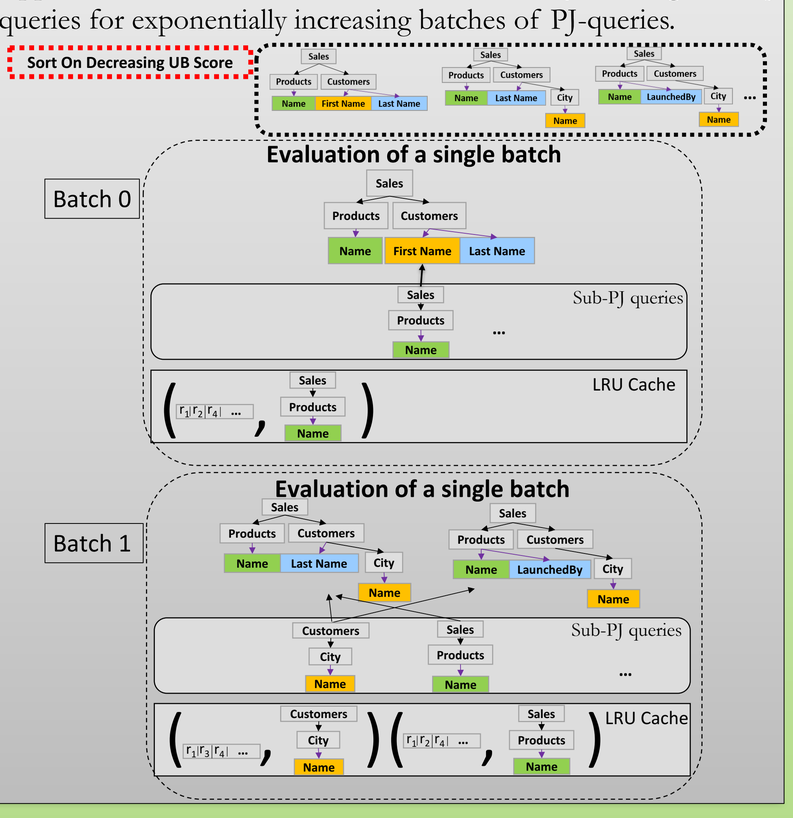
Given a set of PJ queries ordered by their upper bound scores can we minimize the cost of evaluation to produce the top-k output?

NP-hard even when the minimum evaluation set is given by an oracle

Sharing Sub-PJ Queries & Early Termination

Idea

Approximate the minimum evaluation set by sharing sub-PJ queries for exponentially increasing batches of PJ-queries.



Experiments // Settings

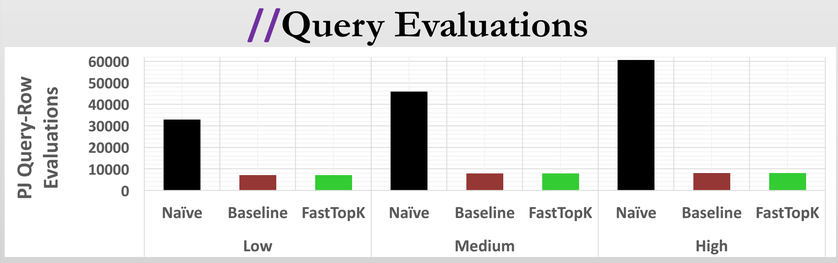
DATABASE					
	Inv. Index (MiB)	(key,fk) snapshot (MiB)	#Relations	#Text Columns	#Edges
CSUPP	4759.7	1237.4	105	821	63

EVALUATION STRATEGIES	
Naïve	Evaluation of ALL minimal PJ-Queries
Baseline	Evaluation w/ **early termination strategy**
FastTopK	Evaluation w/ **sharing sub-PJ queries and early termination**

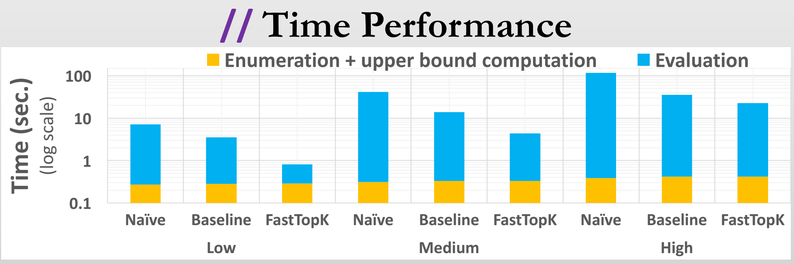
SPREADSHEET GENERATION
50 spreadsheets generated from meaningful PJ-Queries (added **2 alignment errors**) bucketed under **Low**, **Medium**, **High** cost classes

Experiments // Evaluation

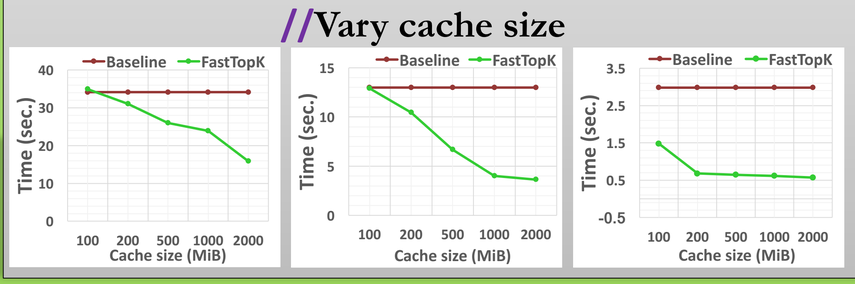
//Query Evaluations



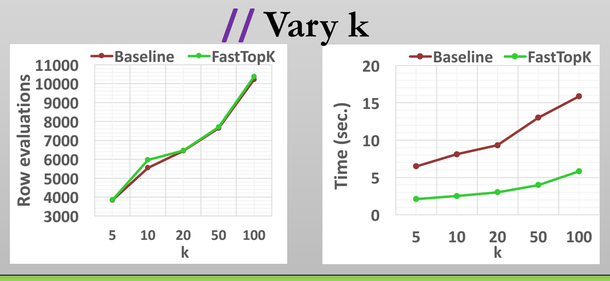
// Time Performance



//Vary cache size



// Vary k



Microsoft Research