Overlap Interval Partition Join

University of Zurich[UZH]

Anton Dignös, Michael H. Böhlen, and Johann Gamper

{adignoes, boehlen}@ifi.uzh.ch, gamper@inf.unibz.it

Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

Goal and Approach

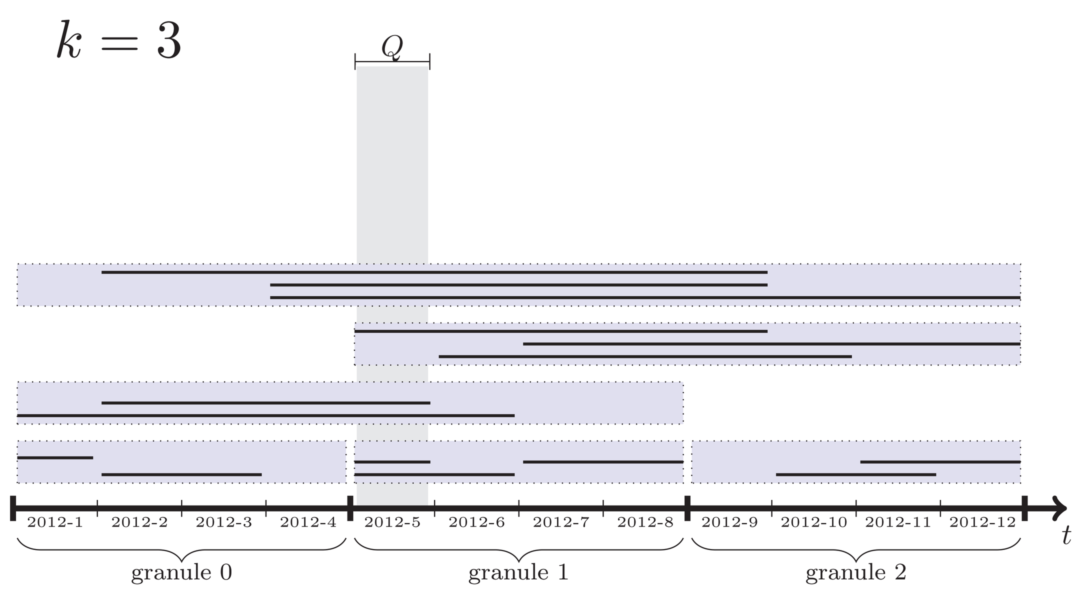
- Efficient **interval partitioning** for the overlap join in valid-time databases
  - Find all pairs of tuples with overlapping intervals
- Partition intervals according to **position** and **duration**
- Self-adjusting: automatically determine the **optimal number of partitions**

Challenges

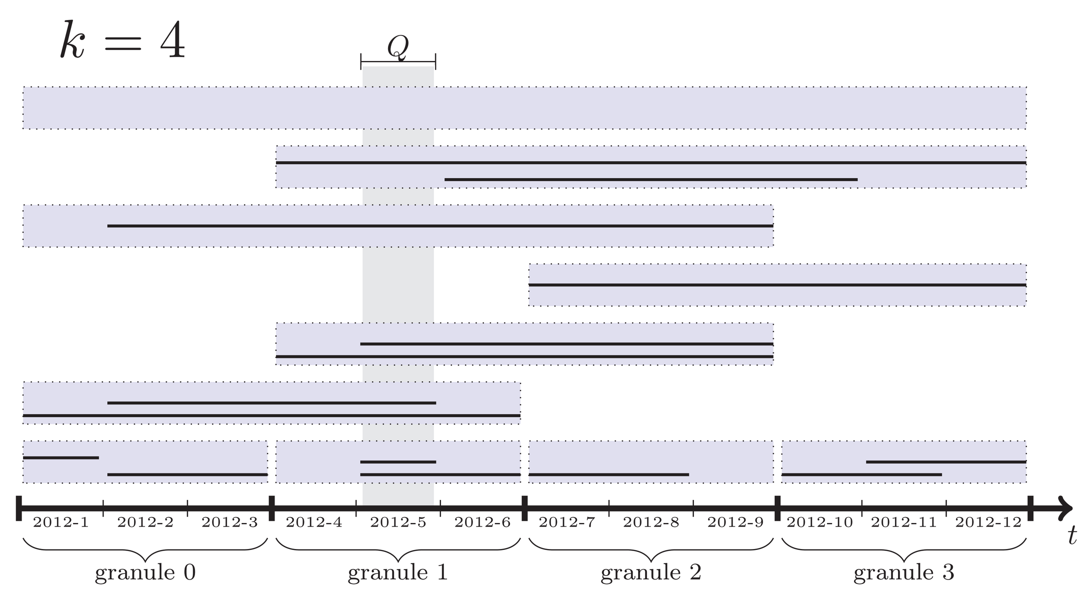
- Determine partition parameter as a trade-off between number of false hits and number of partition accesses
- Efficient access structure that allows to omit empty partitions

Overlap Interval Partitioning - $\mathcal{OIP}$

- Time range is divided into k granules of equal duration
- Partitions are any sequence of contiguous granules



Low $k \Rightarrow$ fewer partition accesses (less overlapping boxes)



High $k \Rightarrow$ more precise partitions (better fitting boxes)

Constant clustering guarantee: Duration of tuple and partition differs by less than two granules.

Cost Dimensions

False hits: Overhead for tuples that are fetched for Q, but are not part of the result.

- CPU cost: identifying and discarding
- IO cost: more data is fetched

Partition accesses: Overhead for fetching and accessing partitions for Q.

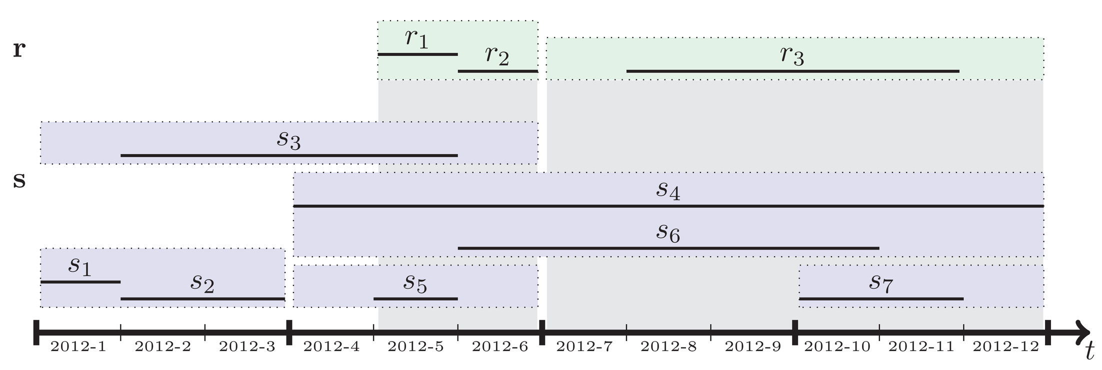
- CPU cost: search in the access structure
- IO cost: more partially filled blocks

False hits and partition accesses are **inversely** related.

OIPJoin

Algorithm

1. Determine parameter k for $\mathcal{OIP}$
2. Partition both input relations using k
3. Join tuples within overlapping partitions



Determining Parameter k for the OIPJoin

Approach: Find k by minimize the overhead cost function $cost(k)$ w.r.t. k:

partially filled blocks (1 trailing block per partition)

search in access structure (2 comparisons in access list)

$$cost(k) = |p_r| \cdot \boxed{\text{APA}} \cdot (c\_io + 2 \cdot c\_cpu) + \quad \textbf{cost for partition accesses}$$

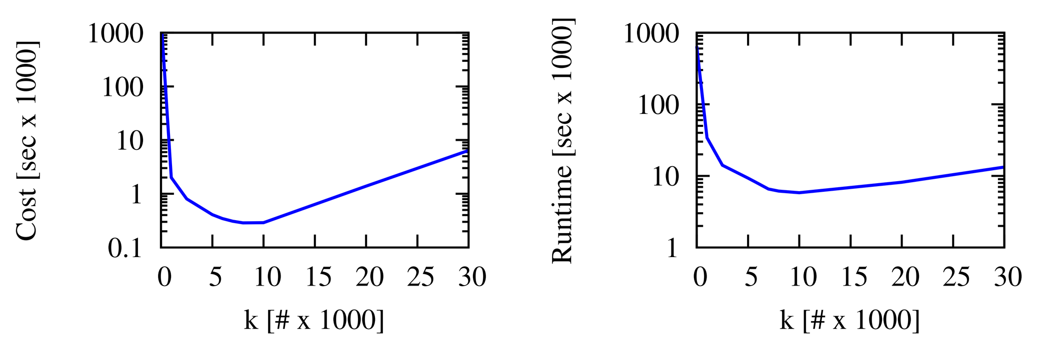
$$|p_r| \cdot n_s \cdot \boxed{\text{AFR}} \cdot \left(\frac{c\_io}{b} + 2 \cdot \frac{n_r}{|p_r|} \cdot 2 \cdot c\_cpu\right) \quad \textbf{cost for false hits}$$

more data is fetched (1 false hit within a block)

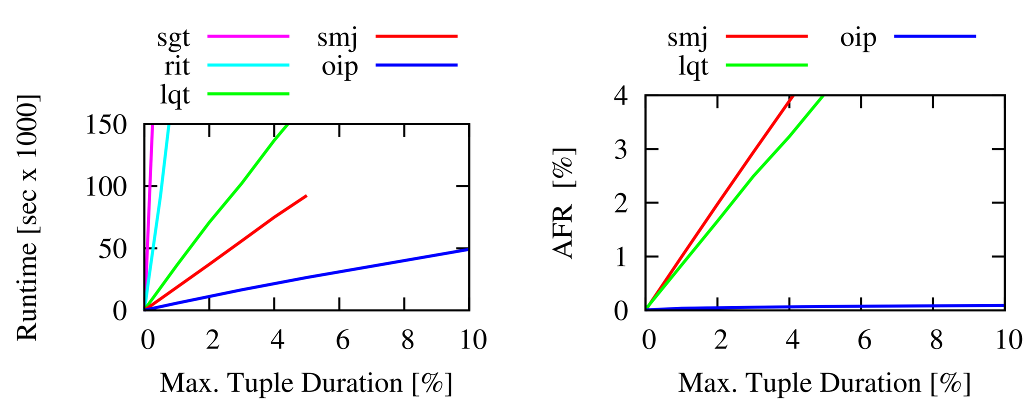
identifying and discarding (2 comparisons per false hit)

Result: $k = f(\text{CPU cost, IO cost, relation sizes})$

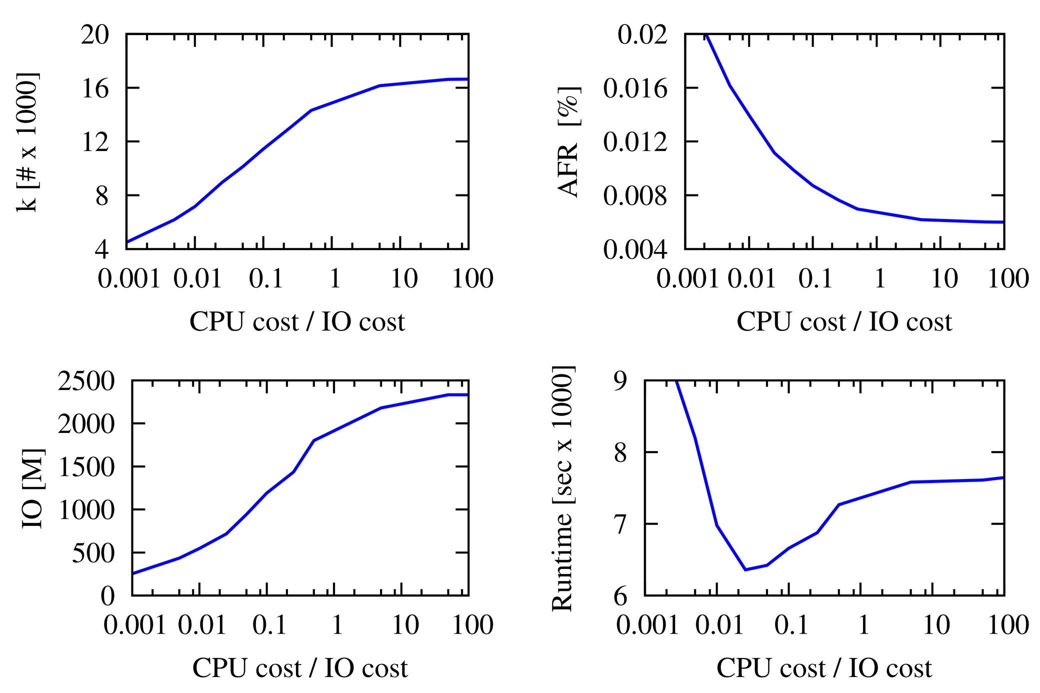
Empirical Experiments



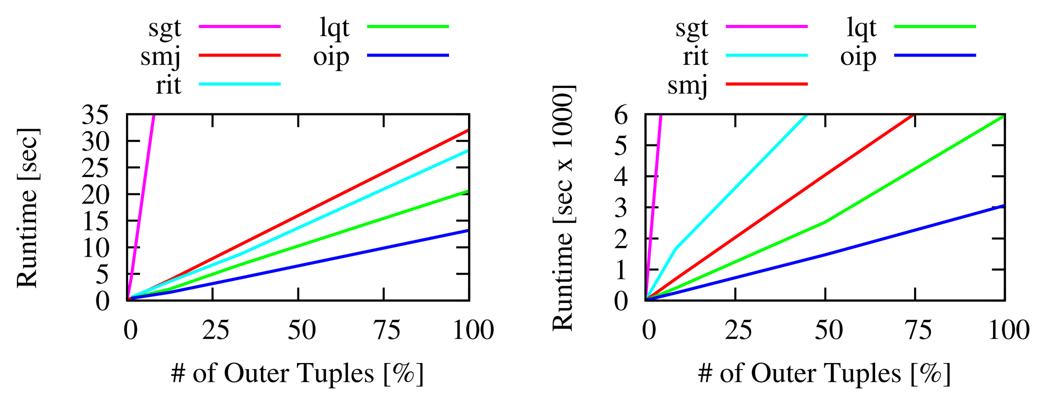
Cost function compared to Runtime



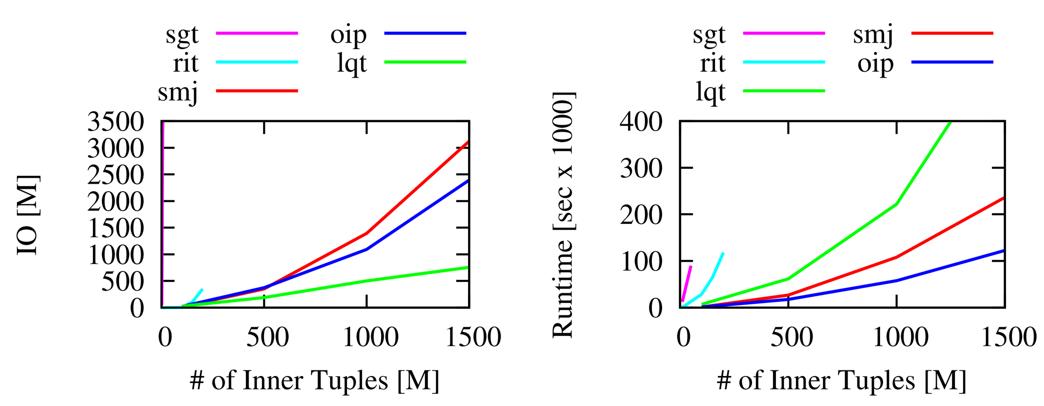
Impact of long tuples



k adapting to CPU and IO cost



Personnel and file change data



Disk Resident Data

Summary

Summary

- $\mathcal{OIP}$ partitions intervals according to position and duration.
- Long tuples in the datasets do not deteriorate the performance of $\mathcal{OIP}$.
- OIPJoin is self-adjusting: k is determined by minimizing the total cost for false hits and partition accesses.
- OIPJoin is robust for long tuples.

Future Work

- Advanced statistics to calculate the number of empty partitions and the reduced average number of partition accesses APA, for instance using histograms.
- Study the maintenance of $\mathcal{OIP}$.
- Refinement of cost function for different buffer replacement strategies.

Published in: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, Snowbird, Utah, USA, June 2014.