Period Index: A Learned 2D Hash Index for Range and Duration Queries

Andreas Behrend\textsuperscript{1}  Anton Dignöss\textsuperscript{2}  Johann Gamper\textsuperscript{2}  Philip Schmiegelt\textsuperscript{3}  Hannes Voigt\textsuperscript{4}  Matthias Rottmann\textsuperscript{5}  Karsten Kahl\textsuperscript{5}

\textsuperscript{1}University of Bonn, Germany

\textsuperscript{2}Free University of Bozen-Bolzano, Italy

\textsuperscript{3}Fraunhofer FKIE Bonn, Germany

\textsuperscript{4}TU Dresden, Germany

\textsuperscript{5}University of Wuppertal, Germany

SSTD’ 19, Vienna, Austria
Background and Motivation

Contributions

Period Index

Experiments

Conclusion and Future Work
Temporal period data is produced in many application domains:

- Personnel data (work contract periods, project assignment periods)
- Financial data (insurance policy periods, rental contract periods)
- Medical data (hospital stay periods)

...or derived via state analysis:

- Medical data (treatment periods, high fever periods)
- Air traffic data (maneuvering periods, landing periods)
Example: Fever periods of patients

<table>
<thead>
<tr>
<th>Patient</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ann</td>
<td>[01-Aug-2019, 03-Aug-2019)</td>
</tr>
<tr>
<td>Sam</td>
<td>[02-Aug-2019, 05-Aug-2019)</td>
</tr>
</tbody>
</table>

Graphical illustration
Background and Motivation /3

- Heavy research efforts since decades
  - Temporal data storage
  - Temporal keys
  - Temporal indexing
  - Time travel queries
  - Temporal aggregation queries
  - Temporal join queries

- Some works found their way into commercial DBMSs
Temporal features in the SQL:2011 standard [7]
- Period specification, application time and system time
- Predicates like overlap, before, during
- Implementation in IBM D2, Oracle, Teradata, SQLServer [4]

Range types in PostgreSQL (2012) [9]
- Range datatype, predicates, and functions
- Advanced temporal query processing prototype [5]
An important primitive is time travel or range queries
- Find all active insurance policies *as of yesterday*
- Find all air planes maneuvering *yesterday between 08:00 and 10:00*

- An often neglected attribute is the duration of periods
  - Find all active insurance policies *longer than a year*
  - Find all air planes maneuvering for *5 to 6 hours*

Position *and* duration are key properties for querying data with intervals!
Three Types of Queries

Range Query
Find all patients that had fever last week

Duration Query
Find all patients that had fever for more than two days

Range-Duration Query
Find all patients that had fever last week for more than two days
Challenges

Range Predicate

- Corresponds to the overlap of two intervals
- Inequality among both endpoints (difficult to index and optimize)
- overlap(I, [start, end)) = Ie > start AND Is < end

Duration Predicate

- Rarely interested in a precise duration, i.e., 2h 5min 32 sec
- Needs to be expressible as a range, i.e., duration between 2h and 4h
- duration(I) > min AND duration(I) < max

We want to use both at the same time!
Background and Motivation

Contributions

Period Index

Experiments

Conclusion and Future Work
Contributions

- *Period Index*: novel index for intervals according to **duration and position** on the timeline

- Grid-based data structure with **constant time lookup** – well suited for parallelization

- Space-efficient implementation using arrays and linked lists to **avoid storage of empty cells**

- Adaptive bucket length for different **data distribution** (learn)
Related Work

Range Queries

- Interval tree [3, 6]
- Segment tree [2]

Duration Queries

- Any sorted index, most notably B-trees

Range-Duration Queries

- Multidimensional indices, most notably quad-trees, R-trees [1] and Grid file [8]
Background and Motivation

Contributions

Period Index

Experiments

Conclusion and Future Work
Intuition of the Period Index

Idea

▶ Index position of intervals using **buckets** and cells
▶ Index duration of intervals using **levels**

Observations / Assumptions

▶ Position of intervals may be arbitrarily skewed
▶ Duration of intervals follows ZIPF distribution
  (many short / few long)

Desiderata

▶ Access to **relevant** cells should be **fast**
▶ Cells should be **equally filled**
**Data Structure**

- Buckets of fixed length $l$ to index position (horizontal)
- Levels with smaller cells to index duration (vertical)

| Indexation of $|e|$ | Level 0 $|e| \in \left( \frac{l}{2}, \infty \right)$ | Level 1 $|e| \in \left( \frac{l}{4}, \frac{l}{2} \right)$ | Level 2 $|e| \in \left( \frac{l}{8}, \frac{l}{4} \right)$ |
|-----------------|-----------------|-----------------|-----------------|
| $t_i$ | $t_i + \frac{l}{8}$ | $t_i + \frac{l}{4}$ | $t_i + \frac{3l}{8}$ |
| | | | $t_i + \frac{5l}{8}$ |
| | | | $t_i + \frac{3l}{4}$ |
| | | | $t_i + \frac{7l}{8}$ |
| | | | $t_i + l$ |

- Intervals are stored in all overlapping cells on the corresponding level
- Relevant cells are calculated arithmetically
Construction / Interval Assignment

Given an interval $e = [e_s, e_e)$ with duration $|e|$.

1. Find corresponding level:

$$x = \min \left( \max \left( \left\lfloor \log_2 \left( \frac{l}{|e|} \right) \right\rfloor, 0 \right), n_l \right)$$

2. Place $e$ into cells $c_i$ s.t.:

$$\left\lfloor \frac{e_s - o}{l} \cdot 2^x \right\rfloor \leq i \leq \left\lceil \frac{e_e - o}{l} \cdot 2^x \right\rceil$$

Relevant cells for interval $e$ are calculated **arithmetically**

$f(e, d) \rightarrow \{c_i, c_j, \ldots\}$
Example / Interval Assignment

- Buckets length $l = 24h$ and number of levels $n_l = 3$
- $e_5 = [02:45, 08:30)$ with $|e_5| = 5h\ 45m$

$e_5 \rightarrow$ level 2, cells $\{c_1$ and $c_2\}$
Query Evaluation

Given a Range-Duration query $Q$ with range $e = [e_s, e_e)$ and duration restriction $d = [d_{min}, d_{max})$.

1. For each level $x$ s.t.:

$$\min\left(\max\left(\lfloor \log_2\left(\frac{l}{d_{min}}\right)\rfloor, 0), n_l\right) \leq x \leq \min\left(\max\left(\lfloor \log_2\left(\frac{l}{d_{max}}\right)\rfloor, 0), n_l\right)$$

2. Scan all cells $c_i$ s.t.:

$$\left\lfloor \frac{e_s - o}{l} \cdot 2^x \right\rfloor \leq i \leq \left\lfloor \frac{e_e - o}{l} \cdot 2^x \right\rfloor$$

Relevant cells for query $Q$ are calculated \texttt{arithmetically}

$f(e, d) \rightarrow \{\ldots, c_i, \ldots\}$
Example / Query Evaluation

\( Q(e, d): e = [08:00, 10:00) \) and \( d = [5h, 10h) \)

\[
\begin{align*}
&Q 
\rightarrow \{(\text{level 1, cells } c_1), (\text{level 2, cells } c_2)\}
\end{align*}
\]
Analysis

- The number of cells proportionally determines the number of collisions in the index; Large dataset $\rightarrow$ many cells

- The maximum duration of intervals limits the maximum bucket length $l$

- The minimum duration of intervals limits the maximum number of levels $n_l$

- Given $l$ and the number of levels $n_l$ we can control the total number of cells.

  $\#\text{cells} = (2 \cdot 2^{n_l} - 1) \cdot \frac{|D|}{l}$ adjustable in the range of 1 to $2 \cdot |D|$

\[|D| \text{ is the domain size}\]
Adaptive Bucket Length

- What if start times are not uniformly distributed?
- Use Histogram of starting points to model the distribution
- Replace regular time division with weight from cumulative histogram

- Relevant Cells are calculated \textit{arithmetically}
  \[ f(e, d) + H \rightarrow \{c_i, c_j, \ldots\} \]
Background and Motivation

Contributions

Period Index

Experiments

Conclusion and Future Work
Experiments - Setup

- Adaptive bucket lengths

- Competitors
  - Grid file
  - Interval tree (Range Queries only)
  - B+-tree (Duration Queries only)

- Runtime in Query/sec
  - Range Queries only
  - Duration Queries only
  - Range-Duration Queries
  - Different duration distributions
Experiments - Adaptive bucket lengths

- Impact of start time point distribution on cell fill factor

- Adaptive bucket length is able to counteract (learn) data skew and distribute load among cells
Experiments - Range Query

Period index applies efficient calculation of relevant cells (compared to trees)

Period index has smaller cells for small intervals (compared to Grid)
Experiments - Duration Query

Period index applies efficient calculation of relevant cells (compared to trees)

Regular grid may in some cases have less relevant cell accesses
Experiments - Range-Duration Query

- Interval Tree
- B+-tree
- Grid
- Period Index

- Period index applies both range and duration restriction (compared to trees)
- Period index has smaller cells for small intervals (compared to Grid)


Background and Motivation

Contributions

Period Index

Experiments

Conclusion and Future Work
Conclusion and Future Work

- Period index: efficient access method for period-duration queries
- Constant time access to relevant cells
- Adaptive to different start time distributions
- Well suited for parallelization

Future Work

- Extend to more fine grained duration distribution among cells
- Extensive experiments on parallelization and NUMA
- Adaptability to dynamic scenarios similar to extendible hashing

Thank you for your attention!