How to Implement your Trajectory Data Warehouse? Design and Evaluation

Rim Moussa[1], Sandro Bimonte[2] and Robert Wrembel[3]

[1]University of Carthage, Carthage, Tunisia
[2]INRAE, Clermont-Ferrand, France
[3]Poznan University of Technology, Poznań, Poland

Abstract

The management and analysis of massive trajectory data present significant challenges in storage scalability and computational efficiency. In this paper, we introduce *TDWbench* - a benchmark designed to assess and compare two alternative trajectory data warehouse schema designs, i.e., *point-based* and *cell-based*. *TDWbench* is based on a large-scale open maritime trajectory dataset provided by the *Danish Maritime Authority* The benchmark allows assessing trade-offs between *storage efficiency*, *query performance*, and *query accuracy*. We provide experimental results that demonstrate how *TDWbench* supports the comparison of trajectory data warehouse models with realistic analytical workloads, for different spatial resolutions and scale factors.

Keywords

trajectory data warehouse, logical data warehouse schema, big data, OLAP, benchmark

1. Introduction

Trajectory Data Warehouses (TDWs) are first-class citizens in Business Intelligence systems, specifically designed for the analysis of trajectory data [1]. Several logical and physical design approaches are available for TDWs. Although this is a well-recognized need within the field, a comprehensive and unified benchmark for TDWs has not been proposed yet. Such a benchmark lies at the intersection of classical DW characteristics (i.e., scale factors and selectivity) and spatial benchmark characteristics (i.e., complex data and spatial resolutions). Moreover, due to the different existing logical models for TDWs, a TDW benchmark should: (1) address various logical models and (2) propose metrics for their evaluation and comparison. Finally, the TDW benchmark should provide a workload that effectively tests all main classes of analytical queries, specifically with aggregation, join, and selection operations.

To address these challenges, we propose *TDWbench* - a new benchmark on spatio-temporal data, designed to support popular schema designs, including the *point-based* and *cell-based* models. *TDWbench* enables a systematic investigation of the trade-offs between storage cost, query performance, and query accuracy. *TDWbench* relies on a real-world open trajectory dataset provided by the Danish Maritime Authority. The dataset includes trajectory logs ranging from 2 GB per day to 60 GB per month, for a total of more than 2 TB of historical data. This scale allows the evaluation of performance under realistic conditions. Specifically, we developed the following components: (1) two alternative logical TDW models: *point-based* and *cell-based*; (2) an analytical workload composed of 84 queries specifically designed to capture typical analytical query patterns, executed on both TDW models; (3) spatial resolution, scaling factor, and selectivity configurations; (4) a storage metric and a query accuracy metric adapted to approximate workload processing.

Furthermore, our benchmark fulfills well-known requirements of benchmarking [2], namely: relevance, repeatability, economy, fairness, and performance.

Published in the Proceedings of the Workshops of the EDBT/ICDT 2026 Joint Conference (March 24, 2026), Tampere, Finland
rim.moussa@enicar.ucar.tn (R. Moussa); sandro.bimonte@inrae.fr (S. Bimonte); robert.wrembel@put.poznan.pl (R. Wrembel)
0000-0001-6037-5718 (R. Wrembel)

2. Related Work

In this section, we provide an overview of the related work spanning two complementary areas of research: (1) TDW designs, which extend traditional data warehousing concepts to manage and analyze spatio-temporal data representing the movement of objects and (2) TDW benchmarks.

TDW designs. In [3], the authors present the design and implementation of a TDW called *T-WAREHOUSE*. *T-WAREHOUSE* integrates all the necessary components for TDW management, encompassing trajectory reconstruction, ETL processing, and visual OLAP. In [4] the conceptual schema of a segment-based TDW is proposed, where the fact table contains trajectory segments and their attributes (like the geometry of a segment route, a distance traveled, speed, and duration). In [5], the authors use Cartesian grid systems with fixed x-y dimensions to map a regular lattice onto a spherical globe. [6] focuses on and end-to-end method for building a TDW, from data pre-processing, TDW loading, to building analytical solutions.

TDW benchmarks. *BerlinMOD* [7] is a benchmark based on human mobility data. It is parameterized by: (1) the number of people and (2) the length of observation period; both impact the scale factor. *BerlinMOD* defines 17 types of queries. [8] surveys spatial analytics systems and conducted experiments to compare five Spark-based systems on 5 different spatial queries (range, kNN, spatial joins between various geometric datatypes, distance join, and kNN join) and 4 different datatypes (points, linestrings, rectangles, and polygons). Their evaluation uses an Open Street Maps dataset of 500 million rows (56.4GB). *MobilityDB* [9] is an open-source MOD that provides abstract data types and operations for managing mobility data in PostgreSQL and PostGIS. *MobilityDB* features an extensive set of spatiotemporal operations (e.g., distance, temporal predicates, range predicates). The authors assessed the performance of *MobilityDB* using *BerlinMOD*. Distributed query capabilities in *MobilityDB* were demonstrated on a data set of 2 billion AIS ship trajectory points (500 GB), obtained from the Danish Maritime Authority. They utilized Citus—a Distributed PostgreSQL for Data-Intensive Applications [10]. The logical schema contained two tables: *Ship* and *Port*. The authors reported the performance of four analytical queries, all of which include a spatial join with specific ports and highly selective temporal filtering. In [11], the authors proposed an open-source, anonymized, metropolitan-scale dataset (*YJMob100K*) of 100,000 human mobility trajectories within 90 days, from Yahoo Japan Corporation. Location pings are mapped into 500×500 meter grid cells, and timestamps are aggregated into 30-minute intervals.

To conclude, benchmarks for spatial data storage and querying systems are essential for identifying the most appropriate logical and physical designs. However, to the best of our knowledge, no existing work proposes a fully-featured, ad hoc benchmark specifically tailored to TDWs. Such a benchmark would need to encompass multiple design schemas, varying scale factors, query workloads, and relevant performance metrics.

3. Benchmark Description

This section introduces *TDWbench*[1] designed to compare spatio-temporal data models and systems that manage trajectory data. The benchmark is constructed from a real-world dataset of maritime voyages (Section 3.1) and enables the generation of TDWs with varying data volumes, in a manner analogous to well-known TPC benchmarks. The key characteristics of *TDWbench* are discussed below.

- *Scalability*: the scale factor of the DW is represented by length of the observation period (the number of days).
- *Schema comparison*: the benchmark supports a direct comparison between *point-based* and *cell-based* schemas. For the latter, the *resolution* parameter defines cell granularity.
- *Workload*: *TDWbench* includes a representative analytical workload comprising common analytical operations such as grouping, joins, and filtering, reflecting realistic trajectory analytical queries.

[1] https://github.com/rimmmmm/TDWbench

- *Performance and accuracy reporting*: the benchmark provides a comprehensive evaluation of both performance and the quality or query results measured by accuracy. The performance metric includes query execution time, while the accuracy metric quantifies the accuracy loss induced by spatial aggregation and discretization, enabling an explicit trade-off analysis between efficiency and analytical precision.
- *Implementation*: *TDWbench* is implemented using relational DBMS PostgreSQL v.11 and its spatial extension PostGIS. Such an implementation does not compromise its genericity, as deploying the benchmark on another relational DBMS is generally a straightforward task.

3.1. Data Sources Description

We consider three data sources for building the TDW, namely: (1) *Vessels navigation AIS logs* [12]; (2) *Marine geometry data*; [13] (3) *World port index* (WPI) [14]. The AIS logs dataset does not explicitly contain trips. To obtain the navigation attributes such as departure and arrival ports, it is necessary to join the AIS data with the spatial coordinates from the WPI database.

3.2. Point-based schema

In the *point-based* schema, a trajectory T is defined as: $T = \langle p_1, p_2, \ldots, p_n \rangle$, where each position p_i is a spatial-temporal point: $p_i = \langle x_i, y_i, t_i \rangle$ such that: $x_i, y_i \in \mathbb{R}$ represent the geographical coordinates (longitude and latitude); $t_i \in \mathbb{R}$ represents the timestamp of the location ($t_1 < t_2 < \cdots < t_n$).

3.3. Cell-based schema

In the *cell-based* schema, each point p_i is mapped to a spatial cell c_i represented as a polygon. A trajectory is represented as: $T = \langle c_1, c_2, \ldots, c_m \rangle$ where $c_i = \langle poly_i, t\_entrance_i, t\_exit_i \rangle$, where $poly_i$ is a polygon describing the boundaries of the cell containing sequential positions; $t\_entrance_i$ denotes the timestamp corresponding to the vessel's entry into the cell c_i; and t_exit_i denotes the timestamp corresponding to the vessel's departure from the cell c_i.

3.4. Point-based vs. cell-based: resolution

On the one hand, points representing a trajectory result in the most accurate trajectory, whereas cells offer an approximate trajectory. On the other hand *point-based* storage occupies a larger disk space as compared to *cell-based*, and as a consequence queries on a *point-based* DW schema are more expensive. Moreover, the size of a cell plays a crucial role in the trade-off between accuracy of a query result, data storage, and query execution time. Figures 1 and 2 illustrate a trip trajectory represented as a sequence of points (red curve) alongside its corresponding spatial cell polygons (blue cells). RES2 resolution illustrates a finer spatial granularity compared to RES1 resolution.

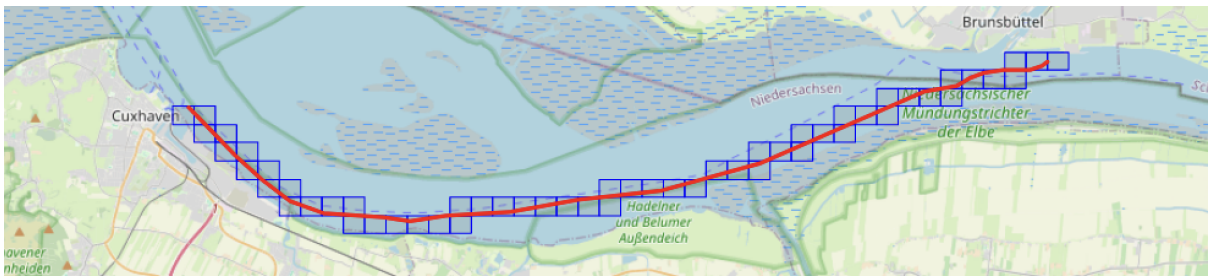

Figure 1: An example *point-based* trajectory (red curve) and *cell-based* trajectory (blue cells) from port *Cuxhaven* to *Brunsbuttel*; resolution (RES2-cell dimension: height ≈ 0.61 km and width ≈ 0.72 km).

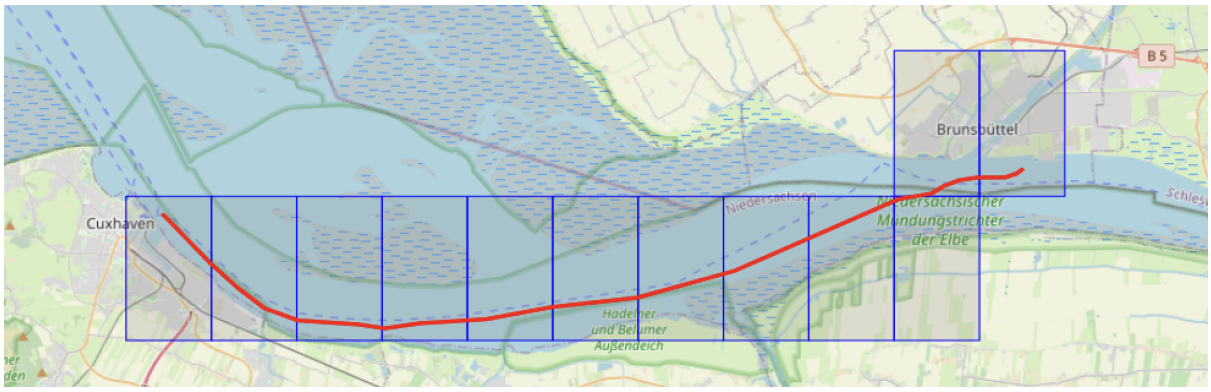

Figure 2: An example *point-based* trajectory (red curve) and *cell-based* trajectory (blue cells) from port *Cuxhaven* to *Brunsbuttel*; resolution (RES1-cell dimension: height ≈ 4.89 km and width ≈ 2.88 km).

3.5. Workload

The analytical workload consists of 12 batches (see Table 1) of 7 queries each. Queries either include filtering on the fact tables or incorporate temporal or spatial selectivity predicates. Three *spatial selectivity values* (6%, 18%, 36%) and three *temporal selectivity values* (8%, 16%, 33%) are considered, resulting in a total of 84 queries. The proposed workload is designed to: (1) highlight the differences in query processing performance across various DW schemas, (2) incorporate complex query patterns, and (3) address analytical queries that are relevant to maritime transportation. A query can be composed of the following operations: join, grouping, aggregation, and filtering.

Table 1
Description of the workload batches (*Bn* - batch number, *Patterns* - query pattern, *SOG* - speed over ground

Bn	Patterns	Example of query
B1	Spatial Join, Grouping, Numerical Agg.	Compute AVG SOG per marine area
B2	Spatial Join, Grouping, Spatial Agg.	Compute the Area covered by vessels trajectories within each marine area
B3	Spatial Join, No Grouping, No Aggregation	Determine the First point/First cell for each marine area that vessel visits
B4	Spatial Join, No Grouping, Numerical Agg.	Compute the AVG SOG by all trip points/cells within marine areas
B5	Spatial Join, No Grouping, Spatial Agg.	Compute the total area covered by all trip points/cells within marine areas
B6	Equi-Join, Grouping, Numerical Agg.	Compute the AVG SOG per vessel type
B7	Equi-Join, Grouping, Spatial Agg.	Compute the area covered by trips between each pair of ports
B8	Equi-Join, No Grouping, No Aggregation	Determine the first point / first cell for each trip
B9	Equi-Join, No Grouping, Numerical Agg.	Calculate the max. elapsed time within a segment of a trip
B10	Equi-Join, No Grouping, Spatial Agg.	Calculate the max. spatial distance traveled within a short segment of a trip
B11	No Join, No Grouping, Numerical Agg.	Compute the AVG SOG
B12	No Join, No Grouping, Spatial Agg.	Compute the total area covered by all trip points/cells

3.6. Metrics

The main metrics we use in *TDWbench* are: *storage size*, *time performance*, and *query accuracy*, for a given cell resolution. The *time performance* is measured as the total runtime for a given TDW volume. The *query accuracy* is a new metric that quantifies the deviation of approximate query results on *cell-based* schemata from the exact results obtained on the *point-based* schema.

To evaluate the *query accuracy* we use the Root Mean Squared Error (RMSE). For a given measure *m*, RMSE is computed using the value of *m* for the cell-based model $V(m_{cell})$ and the value of *m* for the point-based model $V(m_{point})$, for all the values of *m* output by a query. The formula is the following:

$$\text{RMSE}_m = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(V(m_{cell})_i - V(m_{point})_i\right)^2}$$

4. TDWbench deployment

In this section, we report some experiments to show how *TDWbench* supports comparing TDW implementations according to different parameters: the type of a model, the scaling factor, the resolution, and queries.

The trajectory data warehouse was implemented in PostgreSQL v11. The TDW stored AIS logs from Jan 2023, in two cell resolutions: (1) *RES1-Cell* with approximate dimensions of ~ 4.9 km $\times 4.9$ km (at the equator) and (2) *RES2-Cell* with approximate dimensions of ~ 1.2 km $\times 0.6$ km (at the equator). The TDW was run on a node on the French national grid computing infrastructure - *GRID5000*[2]. The node was equipped with Intel Xeon E5-2620 v4 (2.1 GHz), 16 cores (2×8), and 64 GB RAM node.

Storage: As expected, dimension (e.g., vessels) sizes increase with the scaling factor, but they do not change according to the schema used, since they do not contain trajectory data. The schemas affect the storage of fact tables: *TripPoint*, *RES1 Cell polygon*, and *RES2 Cell polygon*.

The *point-based* schema (*TripPoint*) needs the highest storage volume since it stores individual vessel positions with the highest spatial and temporal resolution (i.e., 35 GB for SF=30). The *cell-based* schema reduces the storage size according to the resolution used. *RES2* has a higher resolution than *RES1*, therefore the fact table for *RES1* occupies less space than that of *RES2* (i.e., 3 GB vs 5.5 GB for SF=30).

Runtime: Figure 3 presents the aggregated runtime results per schema for the two evaluated scale factors. The *cell-based* schema provides substantial performance benefits. Figure 4 presents the runtime distribution across all batches. Batches that include a spatial join consistently show higher runtime.

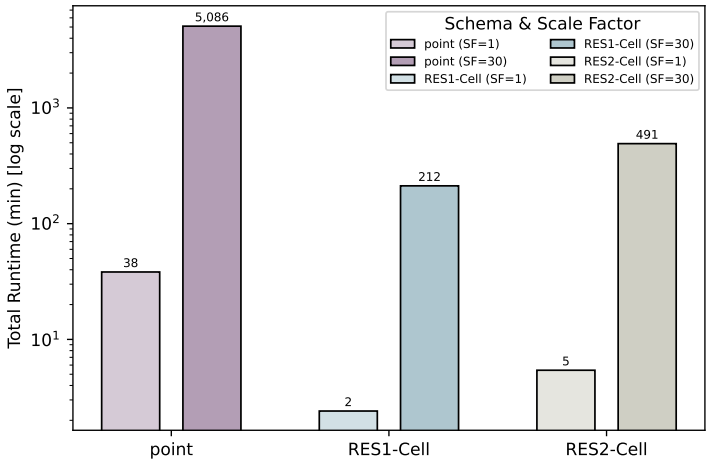

Figure 3: Total runtime per a schema (*point-based* and *cell-based*) and resolution (*RES1* and *RES2*), for two scale factors SF=1 and SF=30

Accuracy: *TDWbench* allows also to compare the accuracy of query results, represented by the RMSE. An example is shown in Figure 5 that details the accuracy for each query of Batch 7 and the two resolutions. As expected, the more fine-grained cell size, the lower RMSE is. The differences in the RMSE values between *RES1* and *RES2* for the same query can be substantial, see *SpatialFilter* of 6% and 18%.

5. Conclusion and Future Work

In this paper we presented *TDWbench* - a benchmark for evaluating TDWs. Built upon over a 2TB trajectory dataset from the Danish Maritime Authority, *TDWbench* implements a scaled dataset and a representative analytics workload to facilitate a detailed investigation of the trade-offs between: (1) storage size (not reported here due to space limit), (2) query performance, and (3) query accuracy. Its experimental results validate the benchmark utility. We are currently extending this research through

[2]https://www.grid5000.fr/

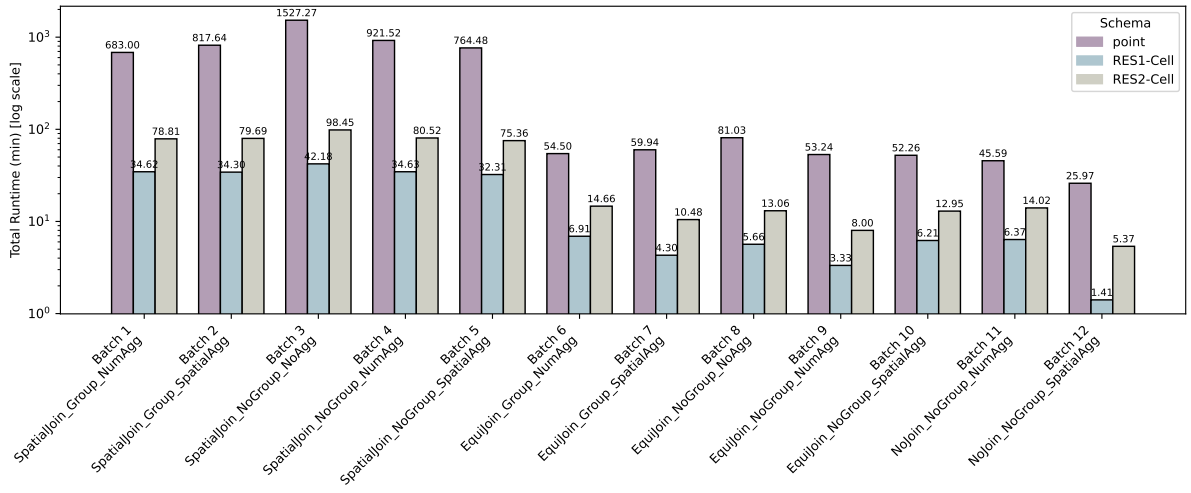

Figure 4: Total runtime per a schema (*point-based* and *cell-based*) and resolution (*RES1* and *RES2*) and batch with SF=30

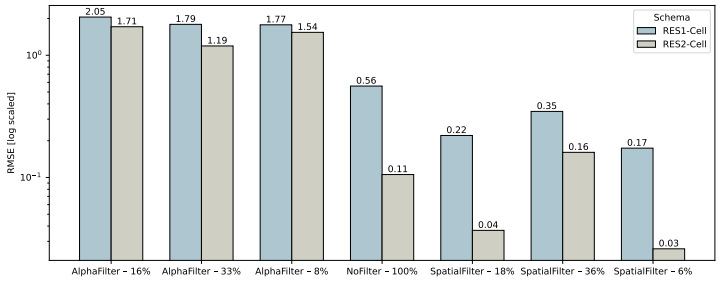

Figure 5: Accuracy per resolution for each query of Batch 7

experimental work on alternative DW schemas for storing spatial data and on in-depth analysis of their impact on query accuracy. Further works with focus on evaluating all 84 queries of *TDWbench*, and evaluating alternative storage systems, like MobilityDB.

Acknowledgement

The work of S. Bimonte is supported by the CHIST-ERA grant ANR-24-CHR4-0004-0 'GIS4IoRT'. The work of R. Wrembel is supported from the National Science Centre (NCN), Poland, grant no. 2024/06/Y/ST6/00136, originally funded from the EU project *Chist-Era call 2023*.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] A. A. Vaisman, E. Zimányi, Trajectory data warehouses, in: C. Renso, S. Spaccapietra, E. Zimányi (Eds.), Mobility Data: Modeling, Management, and Understanding, Cambridge University Press, 2013, pp. 62–82.

[2] J. von Kistowski, J. A. Arnold, K. Huppler, K.-D. Lange, J. L. Henning, P. Cao, How to build a benchmark, in: ACM/SPEC Int. Conf. on Performance Engineering, 2015.

[3] L. Leonardi, G. Marketos, E. Frentzos, N. Giatrakos, S. Orlando, N. Pelekis, A. Raffaetà, A. Roncato, C. Silvestri, Y. Theodoridis, T-warehouse: Visual OLAP analysis on trajectory data, in: Int. Conf. on Data Engineering (ICDE), IEEE, 2010.
[4] C. Renso, S. Spaccapietra, E. Zimányi, Mobility Data: Modeling, Management, and Understanding, Cambridge University Press, 2013.
[5] L. Wu, Y. Xu, Q. Wang, F. Wang, Z. Xu, Mapping global shipping density from ais data, Journal of Navigation 70 (2017).
[6] G. Marketos, E. Frentzos, I. Ntoutsi, N. Pelekis, A. Raffaetà, Y. Theodoridis, Building real-world trajectory warehouses, in: ACM Int. Workshop on Data Engineering for Wireless and Mobile Access (Mobide), 2008, pp. 8–15.
[7] C. Düntgen, T. Behr, R. H. Güting, Berlinmod: a benchmark for moving object databases, VLDB Journal 18 (2009).
[8] V. Pandey, A. Kipf, T. Neumann, A. Kemper, How good are modern spatial analytics systems?, VLDB Endowment 11 (2018).
[9] M. S. Bakli, M. A. Sakr, E. Zimányi, Distributed mobility data management in MobilityDB, in: Int. Conf. on Mobile Data Management, IEEE, 2020.
[10] U. Cubukcu, O. Erdogan, S. Pathak, S. Sannakkayala, M. Slot, Citus: Distributed postgresql for data-intensive applications, in: Int. Conf. on Management of Data (SIGMOD), 2021.
[11] T. Yabe, K. Tsubouchi, T. Shimizu, Y. Sekimoto, K. Sezaki, E. Moro, A. Pentland, Yjmob100k: City-scale and longitudinal dataset of anonymized human mobility trajectories (version 2), 2023.
[12] Danish Maritime Authority makes historical AIS data available, 2018. URL: https://safety4sea.com/danish-maritime-authority-publishes-historical-ais-data/, accessed Jan, 2026.
[13] Natural earth, 2025. URL: https://www.naturalearthdata.com/downloads/, accessed Jan, 2026.
[14] National Geospatial Intelligence Agency. World Port Index, 2019. URL: https://msi.nga.mil/Publications/WPI, accessed Jan, 2026.