MASTERMobilityDB: A Persistence and Manipulation Layer for Trajectories of Multiple Aspects

Flaris Roland Feller[1], **Ronaldo dos Santos Mello**[2]

[1]PPGCC – INE – Universidade Federal de Santa Catarina (UFSC) – Caixa Postal 476 – 88.040-370 – Campus Universitário Trindade – Florianópolis, SC – Brasil

flaris.feller@gmail.com, r.mello@ufsc.br

Abstract. *Spatiotemporal data about the movement of real-world entities collected from sensors and GPS devices, as well as their integration with other contextual georeferenced data, has produced voluminous and complex collections of trajectory data. These collections, known as multiple aspect trajectories (MATs), raise new challenges for moving object databases. This work introduces MASTERMobilityDB, a MAT persistence layer based on the MASTER representation model for MATs, built on top of MobilityDB database through the development of an API. A comparison with the state-of-the-art SecondoDB demonstrates that queries over MATs are expressed in MASTERMobilityDB more naturally and perform better. No similar proposal was found in the literature and industry.*

1. Introduction

Due to the widespread use of GPS-enabled devices, recording position data has become very easy, and large amounts of these data are collected daily. In response, trajectory data management and moving object databases (MODs) has been an active research topic.

According to [Güting et al. 2015], a trajectory describes the movement of an entity, for example, a person, a vehicle, or an animal. At a lower level of abstraction, it is a sequence of positions in space ordered in time, also known as a raw trajectory. When a raw trajectory, in turn, is enriched by several semantic dimensions that are contextual to the movement and heterogeneous in shape, we have the so-called *multiple aspect trajectory*, or simply MAT [Mello et al. 2019]. Figure 1 shows an example of MAT describing the daily movement of an individual starting at home, passing by his/her office and finishing at a restaurant. This movement is enriched by several semantic aspects, like weather condition, means of transportation and *POI (Point-Of-Interest)* attributes.

Since middle of the 2000s, several data models for semantic trajectories have been proposed, such as the *stops and moves* [Spaccapietra et al. 2008], CONSTANT [Bogorny et al. 2014] and the symbolic trajectories [Güting et al. 2015]. This work adopts the *MASTER* data model [Mello et al. 2019, Mello et al. 2021]. MASTER is more comprehensive regarding data variety since it treats aspects not only as simple semantic labels as in symbolic trajectories, or limited by predefined aspects as in CONSTANT. It can also represent complex objects and heterogeneous information associated with a trajectory.

For the persistence of trajectory data, the Database (DB) research community has generated several implementations of MODs such as Secondo [Güting et al. 2015], Her-

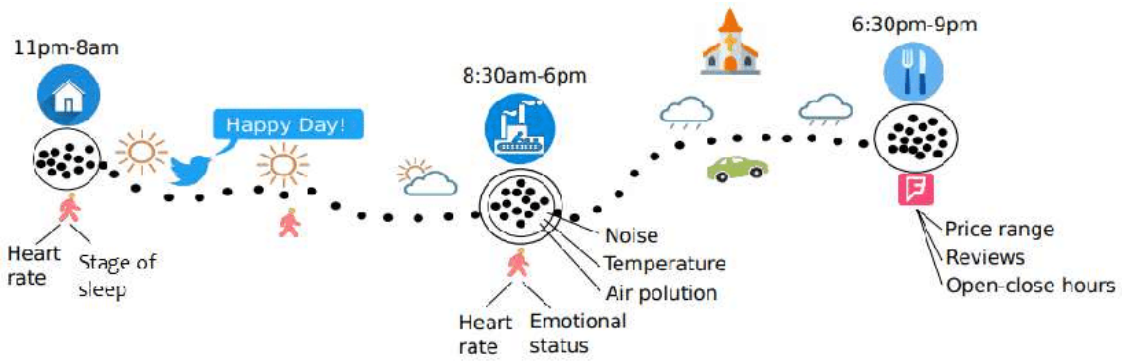

Figure 1. A MAT example.

mes [Pelekis et al. 2015] and MobilityDB [Zimanyi et al. 2020]. The latter is built on top of PostgreSQL[1] and PostGIS[2] extension, which are actively supported by a large community of companies and individuals. MobilityDB has many features for processing high volumes of trajectories, such as abstract data types, specialized indexes, a rich set of operators and functions, and support for several environments, tools and programming languages. Due to it, MobilityDB is a very good option for storing and manipulating trajectory data, however, it focuses on raw trajectories, while Secondo and Hermes are limited to a single or few fixed semantic properties associated to a trajectory, being not able to represent a MAT.

Thus, in order to deal with this gap, this paper presents *MASTERMobilityDB*, a persistence layer built on top of MobilityDB MOD to manipulate MATs based on the MASTER model. We designed and implemented new abstract data types for the MASTER model entities, as well as efficient methods for manipulating them, all of them extending the existing MobilityDB data model. This layer is materialized through an API written in PL-SQL.

The rest of this paper is organized as follows. Section 2 presents basic concepts regarding MAT and the MASTER Data Model. Section 3 exposes a state-of-the-art review of proposals for semantic trajectories persistence. Section 4 details the MASTERMobilityDB. Section 5 discusses experiments over two known semantic trajectory datasets: *Foursquare* and *BerlinMOD*. Section 6 concludes the paper.

2. Multiple Aspect Trajectory and the MASTER Data Model

A MAT [Mello et al. 2019] is a sequence of points $(p_1, p_2, ..., p_n)$ collected for a moving object, with $p_i = (x, y, t, A)$ being the i-th point of the trajectory generated in the location *(x,y)* at timestamp *t* and described by the set $A = a_1 : v_1, a_2 : v_2, ..., a_r : v_r$ of *r* aspect-value pairs that characterize various aspects of the trajectory. In short, aspects represent relevant real-world facts such as social media posts, climate, or transportation modes. Each aspect has attributes that provide detailed information about it, as shown in Figure 1 for an individual's MAT.

Based on the definition of a MAT, the MASTER data model [Mello et al. 2019, Mello et al. 2021] was proposed to remedy the limitations of previous representation

[1] *www.postgresql.org*
[2] *www.postgis.net*

models. MASTER's relational logical schema is presented in Figure 2[3]. In particular, it introduces the concept of *aspect*, which is a real-world fact relevant to the trajectory data analysis.

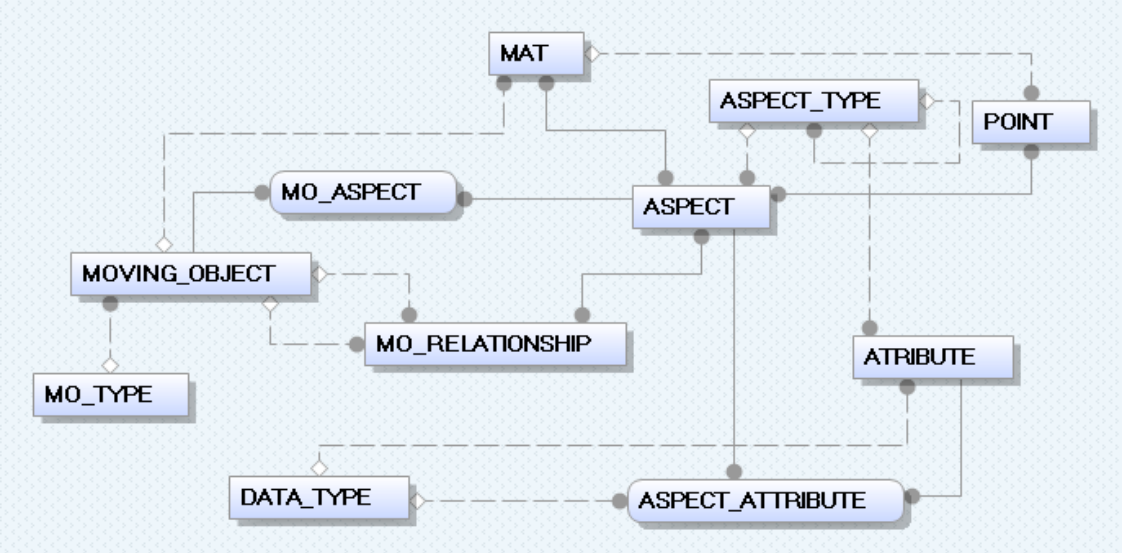

Figure 2. The MASTER data model

An aspect is characterized by a *aspect type*. It defines its metadata and attributes. An aspect supports numbers, ranges of values, text, geometries, or complex objects. For example, an aspect type *hotel* may have the following attributes: *geographical coordinates, address, stars, room types* and *facilities*, and an aspect belonging to this type can be: *"Il Mare Resort"*, with the following attribute-value pairs: *geographical coordinates, −27,439771, −48,500802; address, "Main Ave., 45, Atlantis"; stars, 5; room types, {"suite", "junior suite"}; facilities, {"gym", "swimming pool", "restaurant", "bar", "beach service"}*. An aspect can also be associated to a MAT, a point, a MO, or a relationship between MOs. A MAT belongs to a MO of a type that can be a person, a drone, an animal, a car, or even a natural phenomenon, such as an hurricane.

3. Related Work

A first effort to meet the objectives of this work was to carry out a systematic review of state-of-the-art on the persistence of semantic trajectories in DBs and the implementation of representation models [Feller and Mello 2022]. Seventeen works were selected (see Table 1), organized in the following categories of leading DB technologies: *object-relational (OR), graph, triplestore*, and *multimodel*, including the MASTERMobilityDB.

On implementing a MOD based on an OR DB, a trajectory is represented as an abstract data type (ADT), encapsulating spatiotemporal and semantic features and operations on its properties. Persistence is achieved through relations with attributes inherited from ADTs and handled through SQL. Triplestore-based MOD implementations present ontologically-based approaches for modeling semantic trajectories considering three main ontologies: domain, time, and spatial, which contextualize movement, space,

[3]Table attributes are ommited for sake of understanding of the proposed model.

Table 1. Comparison of solutions for persistence of semantic trajectories

MOD DB	Base DB	Extension	Technology
Hermes@Oracle	Oracle	Spatial	OR
Hermes@Postgres	PostgreSQL	PostGIS	OR
Secondo	BerkeleyDB	Algebra	OR
Weka-STPM	PostgreSQL	PostGIS	OR
[Brandoli et al. 2022]	MobilityDB	PostGIS	OR
[Xu et al. 2023]	Secondo	Symbolic	OR
DeepVQL	PostgreSQL	PostGIS	OR
MASTERMobilityDB	**MobilityDB**	**PostGIS**	**OR**
[Gómez et al. 2019]	Neo4j	Neo4j Spatial/APOC	Graph
[Noureddine et al. 2021]	Neo4j	Neo4j Spatial/APOC	Graph
GSM	Neo4j	Neo4j Spatial	Graph
Hermes@Neo4j	Neo4j	Neo4j Spatial	Graph
FrameSTEP	Virtuoso	-	Triplestore
STriDE	Stardog	-	Triplestore
[Torres et al. 2020]	Apache Jena	-	Triplestore
[Tamilmani and Stefanakis 2019]	PgSQL/Neo4j	PostGIS	Multimodel
[Wannous et al. 2013]	Oracle	Spatial/RDF	Multimodel

and time. Along with the ontologies, rules are defined. The proposals emphasize the use of SPARQL and GeoSPARQL. Property graph-based solutions model semantic trajectories with vertices (points or episodes) and edges that connect them. They can be annotated with semantic or spatiotemporal properties as attributes. At last, the multimodel proposal encapsulates the geometry and semantics of mobility data in different data models. For example, the relational model maintains the raw GPS trajectory, and the graph model associates several semantic properties.

By generalizing the aspect concept, *MATs* represent a new view of trajectories and a new paradigm for mobility data. The MASTERMobilityDB differs from the state-of-the-art by being the first proposal for persistence and manipulation of MATs. We adopt a MOD based on the OR technology as the basis for MASTERMobilityDB (*i.e.*, MobilityDB) to take the advantage of the reusability principle for extending the MobilityDB data model as well as its data access methods.

4. MASTERMobilityDB

MASTERMobilityDB is a MAT data management layer on top of MobilityDB DB Management System (DBMS). MobilityDB, in turn, was built over PostGIS DBMS, which is an extension of the PostgreSQL relational DB for managing georeferenced data. Each component on this stack of DB technologies processes a part of a MAT according to the data types it supports, and communicates with the others through available APIs. PostgreSQL handles alphanumeric data, relationships, referential integrity, and transactions. PostGIS treats geometries and spatial operations. MobilityDB handles raw trajectories and, finally, MASTERMobilityDB is based on the MASTER data model.

4.1. Design issues

MASTERMobilityDB implements several issues to efficiently support the persistence and manipulation of MATs, such as ADTs, tables and methods. Such objects are organized into logical groups using *DB schemes*, providing a way to separate different parts of the

development according to their purpose. The schemes are as follows: *(i) MASTER*: the objects that support the MASTER model and extensions; *(ii) partitions*: physical segmentation of high-volume tables; *(iii) util*: methods used in ETL procedures; and *(iv) staging*: user-developed ETL procedures. The source code for MASTERMobilityDB, as well as its API, are available at *https://github.com/ffeller/MasterMobilityDB*.

In order to design MASTERMobilityDB, we consider spatial data types offered by PostGIS and the *TGeomPoint* type introduced in MobilityDB for the persistence of raw trajectories. Based on these spatiotemporal data types, MASTERMobilityDB defines a series of ADTs, *i.e.*, composite data types that mirror the MASTER model. An example of an ADT is *aspect*, which holds six attributes: *(i) aspect_id*, an integer that identifies the aspect; *(ii)* a variable character (*description*); *(iii)* two float values (x and y representing spatial coordinates); *(iv)* a *timestamp (t)*; *(v)* the *space_time flag* (integer), which indicates the aspect's nature (*spatial*, *temporal*, *spatiotemporal*) or *semantic*, and; *(vi) aspect_type_id*, an integer value that is a reference to a key in the corresponding *aspect_type* relation.

A key-design issue we consider in the definition of the MASTERMobilityDB logical data model is the double representation of trajectory points, as shown in Figure 3. For the MAT representation, we consider the *TGeomPoint* type, as stated before (the *RAW_TRAJECTORY* attribute in the MAT table). This specific MobilityDB data type is efficient for filtering and querying conventional raw trajectories. Nevertheless, we also define the *POINT* table in order to allow the association of aspects that are specific to a trajectory point, like a point that holds a POI. This is in consonance with the MASTER model, giving flexibility to bind aspects to parts of a trajectory besides the whole trajectory.

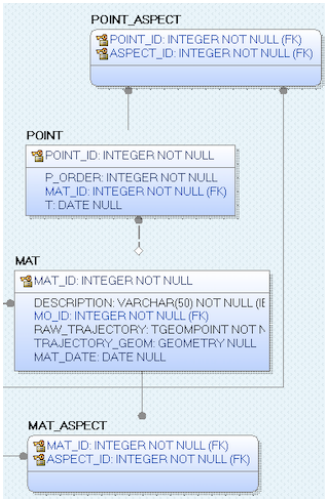

Figure 3. Part of the MASTERMobilityDB logical data model.

In order to simplify the manipulation of MATs, we also design several methods for MASTERMobilityDB. These methods were implemented within the *master* and *util* schemes as procedures and functions in MobilityDB's PL/SQL language. Since the volume of data tends to increase, something considered in the layer design is the manipulation of several tuples in one operation, like *create_many()*, which inserts several tuples in one method call. As an example, Table 2 shows the main methods for the *aspect* ADT.

Table 2. Examples of methods for the *aspect* ADT

Method	Return	Parameters	Description
aspect_count()	integer		counts tuples
aspect_create()		INOUT aspect_typ	creates a new tuple
aspect_create_many()		INOUT aspect_typ[]	creates one or more tuples
aspect_delete_many()		IN aspect_typ[]	deletes one or more tuples
aspect_delete_all()			delete all tuples
aspect_delete_by_id()		IN integer	deletes one tuple by PK id
aspect_delete_by_name()		IN varchar	deletes one tuple by name
aspect_find_all()	SETOF aspect_typ		finds all tuples
aspect_find_by_id()	SETOF aspect_typ	IN integer	finds one tuple by PK id
aspect_find_by_name()	SETOF aspect_typ	IN varchar	finds one tuple by name
aspect_update()		INOUT aspect_typ[]	updates a tuple

4.2. Alternatives for improving data access performance

The integration of motion data and alphanumeric data, as well as the transformation of these large volume of data into useful information is usually complex and expensive. In this sense, MASTERMobilityDB adopts useful features inherited from MobilityDB for loading, querying and building MATs: *(i) object partitioning*; *(ii) temporary tables*; *(iii) external tables*, and *(iv) specialized indexes*. They are detailed as follows.

For datasets that generate large volumes of MATs, tables such as *MAT* or *Point* from the MASTER model can be divided into smaller objects called partitions. It results in better performance when accessing these tables and its indexes, as it is propagated to the partitions using parallelism. For this purpose, there are the *create_partitions_by_date()* and *drop_partitions_by_date()* methods in the *util* schema as described in the Table 3. With this, partitions are created in the *partitions* scheme according to the period and interval (day, week, month, etc.). Table 3 also shows other methods of the *util* schema.

Table 3. Utility methods for ETL procedures

Method	Parameters	Description
create_partitions_by_date	p_schema text, p_table text, p_start date, p_end date, p_column text, p_interval text, p_partschema text	Creates partitions for relation p_schema.p_table for the period.
drop_partitions_by_date	p_schema, p_table, p_start, p_end, p_interval, p_partschema	Drops partitions of relation p_schema.p_table for the period.
disable_fks	p_schema text, p_table text	Disables FKs for the relation.
enable_fks	p_schema text, p_table text	Enables FKs for the relation.
disable_indexes	p_schema text, p_table text	Disables indexes for the relation.
enable_indexes	p_schema text, p_table text	Enables indexes for the relation.
reset_sequence	p_schema text, p_table text	Resets sequence's value.
reset_sequences	p_schema text	Resets all sequences's values.

A temporary table in MobilityDB is a table that exists for a session in the DB and is automatically deleted when the session is closed. They help to store intermediate results within a specific context. In MASTERMobilityDB, temporary tables are widely used in complex queries for manipulating a large volume of tuples. The strategy here is to divide a

complex query into more straightforward queries that have their results stored in indexed temporary tables. It provides superior performance and lower resource consumption.

External tables (or *Foreign Data Wrappers (FDW)*) is a practical feature of MobilityDB that allows working with remote data or data in other formats, such as CSV files, spreadsheets, and JSON documents. FDWs enable the definition of external tables that access these remote data sources and integrate them into a single DB. For MASTERMobilityDB, external tables are essential when mapping data from CSV files. When implementing ETL processes, the following settings must be configured: *(i)* the directory of input files; *(ii)* a user mapping and permissions; and *(iii)* an external table for each file. This strategy makes easier to work with input data without managing it in regular tables, and can be combined with temporary tables.

Finally, *GiST (Generalized Search Tree)* and *SP-GiST (Space-Partitioned Generalized Search Tree)* offer specialized trajectory indexing capabilities. The query optimizer can automatically choose GiST or SP-GiST indexes without programming intervention. In MASTERMobilityDB, MAT queries can be resolved efficiently using alphanumeric indexes for aspects and attributes as a primary filter, and GiST or SP-GiST indexes are considered to accelerate operations and searches over spatiotemporal data.

4.3. Examples of queries

MASTERMobilityDB extends query capabilities of MobilityDB by allowing queries involving the semantic dimension besides queries over raw trajectories, which are limited to spatiotemporal dimensions. Some examples of queries are presented in Figure 4. They show the usage of new useful MASTERMobilityDB methods that allow filters over semantic aspects and their relationships with the main spatiotemporal MASTER data model entities (moving objects, MATs and points). A complete list of MASTERMobilityDB query operations is available at [Feller 2023].

Query 1 selects MATs whose sequence of points contains the *'Green City Hotel'* and *'Viktoriapark'* POIs, which are aspects, together with the sequence of visited locations, the distance traveled being returned by the MobilityDB *LENGTH()* method and the trajectory geometry by the MobilityDB *TRAJECTORY()* method. The use of semantic annotations for POIs and the usage of the MASTERMobilityDB *POINT_FIND_BY_ATTRIBUTE()* method allows a more efficient implementation of the query because it avoids POI and trajectory geometry comparison.

Query 2 presents a *nearest-neighbor* query where both the reference and the candidate objects are moving objects that have passed by *region 74* (an aspect), which can be accessed by the the MASTERMobilityDB *POINT_FIND_BY_ASPECT()* method. The query starts by computing the nearest-neighbors in a MobilityDB temporary table *TRIPDISTANCES*. Then, the main query verifies for each pair of trips *T1* and *T2* where both belong to *TRIPDISTANCES*.

Finally, in Query 3, trajectories of *passenger vehicles* (a moving object aspect) are returned with the aid of the MASTERMobilityDB *MOVING_OBJECT_FIND_BY_ATTRIBUTE()* method. Their related trips are filtered using the *EXTRACT()* method, which returns the day of the week, and the MobilityDB *DURATION(), SPEED(), LENGTH(),* and *TRAJECTORY()* methods, which return the trajectory duration, speed, distance, and geometry, respectively.

Query 1

```
WITH POI_PASSED  (
  SELECT P.MAT_ID
    , ARRAY_AGG(ASP.DESCRIPTION ORDER BY P.T) POIS
    , ST_UNION(ARRAY_AGG(ST_TRANSFORM(
      ST_SETSRID(ST_MAKEPOINT(ASP.X, ASP.Y), 4326), 5676)
        ORDER BY P.T)) COORD
  FROM POINT_FIND_BY_ATTRIBUTE('POI', NULL, NULL) P
    CROSS JOIN LATERAL UNNEST(P.ASPECT_A)  ASP
  GROUP BY P.MAT_ID
) SELECT P.MAT_ID, LENGTH(M.RAW_TRAJECTORY), P.POIS
  , TRAJECTORY(M.RAW_TRAJECTORY), P.COORD
FROM POI_PASSED P
  INNER JOIN MAT M USING(MAT_ID)
WHERE P.POIS @>
  CAST(ARRAY['Green City Hotel',
    'Viktoriapark'] AS VARCHAR(50)[])
  AND MAT_DATE BETWEEN '2007-06-11 00:00:00'
    AND '2007-06-11 23:59:59';
```

Query 2

```
WITH MAT1 AS (
  SELECT T1.MO_ID, T1.MAT_ID, T1.RAW_TRAJECTORY
  FROM MAT T1
    JOIN POINT_FIND_BY_ASPECT('region', '74') P USING(MAT_ID)
), TRIPDISTANCES AS (
  SELECT T1.MO_ID AS MO_ID1, T1.MAT_ID AS MAT_ID1,
    T3.MO_ID AS MO_ID2, T3.MAT_ID AS MAT_ID2, T3.DISTANCE
  FROM MAT1 T1 CROSS JOIN LATERAL (
    SELECT T2.MO_ID, T2.MAT_ID,
      MINVALUE(T1.RAW_TRAJECTORY <-> T2.RAW_TRAJECTORY)
        AS DISTANCE
    FROM MAT1 T2 WHERE T1.MO_ID < T2.MO_ID AND
      PERIOD(T1.RAW_TRAJECTORY) && PERIOD(T2.RAW_TRAJECTORY)
    ORDER BY DISTANCE LIMIT 3 ) AS T3 )
SELECT T1.MO_ID, T1.MAT_ID, T2.MO_ID, T2.MAT_ID, TD.DISTANCE
FROM MAT T1 JOIN MAT T2 ON T1.MO_ID < T2.MO_ID
  JOIN TRIPDISTANCES TD ON T1.MO_ID = TD.MO_ID1
    AND T1.MAT_ID = TD.MAT_ID1 AND
  T2.MO_ID = TD.MO_ID2 AND T2.MAT_ID = TD.MAT_ID2
ORDER BY T1.MO_ID, T1.MAT_ID, T2.MO_ID, T2.MAT_ID;
```

Query 3

```
SELECT MAT_ID, DURATION(M.RAW_TRAJECTORY) DURATION
  , LENGTH(M.RAW_TRAJECTORY)/1000 DISTANCE
  , TWAVG(SPEED(M.RAW_TRAJECTORY))*3.6 SPEED
  , TRAJECTORY(M.RAW_TRAJECTORY) TRAJECTORY
FROM MOVING_OBJECT_FIND_BY_ATTRIBUTE(
    'Vehicle','type','passenger') MO
  INNER JOIN MASTER.MAT_FIND_ALL() M USING(MO_ID)
WHERE EXTRACT(DOW FROM M.MAT_DATE) = 1 --Monday
  AND DURATION(M.RAW_TRAJECTORY)
    BETWEEN INTERVAL '15 minutes'
      AND INTERVAL '30 minutes'
  AND LENGTH(M.RAW_TRAJECTORY)/1000 < 20.0;
```

Figure 4. Examples of queries executed by MASTERMobilityDB.

5. Experimental Evaluation

We provide an experimental evaluation of MASTERMobilityDB by comparing the time performance of data operations against the symbolic trajectories model offered by Secondo DB, the state-of-the-art in terms of semantic trajectory DBMS. Two well-known trajectory datasets were considered: *(i) BerlinMOD* [Güting et al. 2015], which contains simulated data from vehicles circulating in the city of Berlin during 28 days; and *(ii) Foursquare* [Ding et al. 2015]: real data of check-ins in the cities of New York and Tokyo, observed during 10 months. Computing times were achieved on an Intel Core I7 2.8 GHz computer with 16 GB RAM and 512 GB HD, running Ubuntu 22.04. Measurements are carried out by varying the considered percentage of the input data sample: *10%, 40%, 70%*, and *100%*. Next sections detail the design and the results of the experiments.

5.1. BerlinMOD experiment

The BerlinMOD dataset is composed by CSV files described in Table 4. It has been imported to Secondo DB and MASTERMobilityDB through specific ETL processes, generating 292940 daily trajectories.

For Secondo DB, each trip t_i in the *trips* file was converted to a tuple in a relational table, also called *Trips*, with an attribute RT that holds the geometry of the raw trajectory that correspond to the trip. RT value was generated by the grouping of t_i spatiotemporal column values in the *trips* file. In turn, *POIs* and *regions* visited by t_i are the semantic aspects in BerlinMOD. They were represented in Secondo DB as specific semantic complex attributes sem_atts in the *Trips* table. The sem_atts domains hold the geometry of t_i subtrajectories whose points are annotated with POIs' labels or regions' labels, respectively. These annotated t_i trajectories were calculated by the intersection of t_i raw

trajectory geometry with the POI or region geometries. Finally, other non georeferenced data, like *licences* and time *periods*, were stored into regular relational tables.

Table 4. BerlinMOD dataset files

File name	Fields	Description	MMDB	Secondo
instants.csv	InstantId, Instant	Timestamps used for queries	Temp table	Table
licences.csv	LicenceId, Licence, VehId	Vehicle licences used for queries	Temp table	Table
periods.csv	PeriodId, Begin, End	Periods used for the queries	Temp table	Table
points.csv	PointId, X, Y, Name	POIs visited	Aspect	Sem. Att.
regions.csv	RegionId, PointNo, X, Y, Yend, Name	Polygons for regions passed	Aspect	Sem. Att.
trips.csv	TripId, VehId, X, Y, Instant	Vehicles movements	MATs	Table
vehicles.csv	VehId, Licence, Type, Model	Vehicle descriptions	Moving Objs.	Table

For MASTERMobilityDB, *regions* and *POIs* were persisted at the *Aspect* table, and their columns' information at *Attribute* and *Aspect_Attribute* tables. The relationship between POIs and regions with the raw trajectories was calculated in the same way as for Secondo DB. On doing this, it was possible to persist data at the *Point_Aspect* table. The *MAT* table holds the raw trajectories, including the trajectory geometry in the *raw_trajectory* attribute, and the trajectory points were stored in the *Point* table. Vehicles data were stored in the *Moving_Object* table, and other non-georeferenced data have been imported to temporary tables.

The execution mean times on this experiment were determined by running 17 queries, three times each, as well as the load time, as described in Table 5. For sake of paper space, we do not show the implementation of each query in MASTERMobilityDB and Secondo DB. The queries comprise combinations of alphanumeric, temporal, spatial, and spatiotemporal predicates, operations, and aggregations over the data, and the specified query filters allows to test a wide range of index structures, access methods, and spatiotemporal operators.

Table 5. Queries for BerlinMOD dataset

Qry	Description
1	What are the vehicle models with license plates that are on Licenses?
2	How many vehicles are there that are passenger cars?
3	Where were the vehicles with plates on Licenses at each moment of Instants?
4	What license plate numbers belong to the vehicles that passed through the places of interest?
5	What is the minimum distance between places where ten vehicles, drawn from Licenses, were and another ten?
6	Which pairs of "truck" signs are 10m or less apart?
7	What are the license plate numbers of the "passenger" cars that reached the POIs before all the "passenger" cars?
8	What are the total distances traveled by vehicles with License plates during Periods periods?
9	What is the greatest distance a vehicle traveled during each period?
10	When and where did vehicles with License plates meet other vehicles ($distance < 3m$)?
11	Which vehicles passed through a POI point in one of the Instants moments?
12	Which vehicles met at a point in the POIs in an instant of Instants?
13	Which vehicles circulated within one of the regions during the Periods periods?
14	Which vehicles traveled within one of the regions in one of the Instants?
15	Which vehicles passed through a POI during a period of Periods?
16	List the pairs of license plates for vehicles, both in Licenses, where the corresponding vehicles are present in a region during a period of Periods, but are not present at that time.
17	Which POIs were visited by a maximum number of different vehicles?
18	How long does building MATs from the BerlinMOD dataset take?

Table 6 presents the query execution times for MASTERMobilityDB (MM), Secondo DB (Sec), and the spent time difference (Dif). The 10% sample has shown close values, but for higher samples, the difference was more significant in favor of MASTERMobilityDB. It can be explained by the way Secondo DB persists semantic data in terms of labels, dealing with one semantic aspect at a time. So, all semantic complex attributes that hold these labels must be searched for queries that involve multiple aspects to check

if the aspects occur together. On the other hand, MASTERMobilityDB models the semantic dimensions as aspects and attributes, which permit access to any information for the moving objects, as well as their MATs and points. Instead, load time (Query 18) was slightly lower for Secondo DB due to the few number of tables to insert data.

Table 6. Query execution times over MATs for BerlinMOD dataset

Qry	10%			40%			70%			100%		
#	MM	Sec	Dif	MM	Sec	Dif	MM	Sec	Dif	MM	Sec	Dif
1	0,04	0,05	-0,01	0,04	0,04	-0,01	0,04	0,02	0,01	0,04	0,03	0,01
2	0,02	0,05	-0,03	0,02	0,04	-0,02	0,02	0,02	-0,01	0,02	0,02	0,00
3	0,09	0,08	0,01	0,18	0,11	0,07	0,28	0,09	0,19	1,37	0,05	1,32
4	0,19	3,95	-3,76	0,35	19,73	-19,37	0,51	20,10	-19,59	0,62	16,24	-15,62
5	0,15	3,79	-3,65	0,29	19,11	-18,82	0,45	15,63	-15,18	0,85	20,70	-19,85
6	0,43	33,01	-32,58	3,40	588,02	-584,61	7,11	1.253,52	-1.246,40	14,96	2.549,96	-2.535,00
7	0,81	1,55	-0,74	1,80	8,45	-6,65	2,67	8,28	-5,61	3,93	17,30	-13,36
8	0,04	0,10	-0,06	0,07	0,22	-0,15	0,11	0,23	-0,12	0,15	0,32	-0,17
9	5,29	163,06	-157,77	23,14	738,96	-715,82	38,80	1.131,39	-1.092,59	65,43	1.692,15	-1.626,72
10	0,08	16,97	-16,89	0,16	221,67	-221,51	0,24	701,91	-701,67	0,40	1.502,05	-1.501,66
11	0,31	0,07	0,24	0,11	0,07	0,04	0,17	0,11	0,07	0,18	0,16	0,02
12	0,03	0,09	-0,06	0,03	0,08	-0,05	0,03	0,10	-0,07	0,03	0,12	-0,09
13	0,10	1,63	-1,52	0,38	4,98	-4,60	0,39	9,60	-9,21	0,52	12,84	-12,33
14	0,72	0,20	0,52	0,76	0,44	0,33	0,81	0,87	-0,06	0,93	1,42	-0,49
15	0,16	0,16	0,00	0,19	0,26	-0,08	0,21	0,44	-0,23	0,29	0,62	-0,33
16	0,22	6,94	-6,72	0,30	16,42	-16,12	0,30	28,67	-28,37	0,46	38,68	-38,22
17	0,27	1,57	-1,30	1,24	6,84	-5,60	1,24	13,97	-12,73	1,34	18,51	-17,18
18	419,66	372,18	47,47	1.704,04	1.328,30	375,73	3.619,04	2.728,42	890,62	4.848,66	3.339,07	1509,59

5.2. Foursquare experiment

This dataset includes long-term check-in data in New York and Tokyo collected from Foursquare from 12 April 2012 to 16 February 2013. It contains files in CSV format with 8 columns, as shown in Table 7. The dataset had been imported to Secondo DB and MASTERMobilityDB, generating 50603 trajectories calculated weekly from real user check-ins.

Table 7. Data from the Foursquare dataset

#	Description	MMDB	Secondo
1	User ID (anonymized)	Moving Object	Attribute
2	Venue ID (Foursquare)	Aspect	Sem. Att.
3	Venue category ID (Foursquare)	Attribute	Sem. Att.
4	Venue category name (Fousquare)	Attribute	Sem. Att.
5	Latitude	MAT/Point	RT Att.
6	Longitude	MAT/Point	RT Att.
7	Timezone offset in minutes from UTC	MAT/Point	RT Att.
8	UTC time	MAT/Point	RT Att.

For the Secondo DB, a *CheckinSymTraj* table was created to hold user trajectories. It contains an *UserId* that identifies the moving object, and a *RT* attribute to hold the weekly raw trajectory built by the aggregation of *Latitude, Longitude, UTC time*, and *Timezone* offset. Besides *RT*, some *sem_atts* attributes were defined, in the same way of BerlinMOD dataset, to label some *Venue* semantic aspects, as stated in Table 7.

For MASTERMobilityDB, a *Venue* was mapped to an aspect with a *Venue Category ID*, and *Venue Category Name* added to the *Attribute* table. In turn, MATs were created in a same way of Secondo DB, i.e., generated by the aggregation of the same CSV colunms by *User ID* and *week*. Additionally, the intersection of the venues' geometries with MAT and points generated the relationships from the venue aspect with *MAT* and *Points* table, as in BerlinMOD experiment design.

For this experiment, we got the execution mean times for 16 queries, three times each, as well as the load time, as shown in Table 8. Table 9, in turn, shows the query execution times for MASTERMobilityDB, Secondo, as well as the difference. We can notice

that the execution time difference is considerably lower than the BerlimMOD experiment. This behavior can be explained by the lower density of points, as this dataset deals with check-ins that are not so frequent. This scenario lends to simpler trajectory geometries which make the values closer but the difference for all percentages remains in favor of MASTERMobilityDB because it models semantic trajectories in a more efficient way if compared to Secondo DB.

Table 8. Queries for Foursquare dataset

Qry	Description
1	How many trajectories exist for each user with IDs in Users?
2	Where have the users present in Users been at each instant from Instants?
3	Which user IDs belong to Users who passed the POIs from Venues?
4	What is the minimum distance between places from Tokyo where ten users, drawn from Users and another ten?
5	What are the pairs of user IDs from New York within 10m or less of each other?
6	What are the IDs of the users who reached the POIs before all users during the entire observation period?
7	What do users travel the total distances with User IDs from Users during each interval from Periods?
8	What does a user travel the greatest distance during each interval from Periods?
9	When and where did users from Tokyo with User IDs meet other users (distance ¡ 3m), and what are these last IDs?
10	Which users passed by a POI point at New York in one of the Instants?
11	Which users found themselves at a point in the POIs at Tokyo in an instant of Instants?
12	Which users passed by one of the Regions from New York during each interval from Periods?
13	Which users passed by one of the Regions at Tokyo in an instant of Instants?
14	Which users passed a point from Points at Tokyo during a period from Periods?
15	List the pairs of users that were both located within a region from Regions at New York during a period from Periods.
16	List the first time at which a user visited a point at Tokyo in Points?
17	How long does building MATs from the Foursquare dataset take?

Table 9. Query execution times over MATs for Foursquare dataset

Qry	10%			40%			70%			100%		
#	MM	Sec	Dif	MM	Sec	Dif	MM	Sec	Dif	MM	Sec	Dif
1	0.01	0.03	-0.02	0.01	0.04	-0.02	0.01	0.04	-0.02	0.02	0.05	-0.03
2	0.01	0.04	-0.03	0.02	0.06	-0.03	0.02	0.06	-0.03	0.04	0.08	-0.04
3	0.04	3.44	-3.40	0.03	15.68	-15.65	0.03	15.68	-15.65	0.04	47.05	-47.01
4	0.03	0.08	-0.05	0.03	0.13	-0.11	0.03	0.13	-0.11	0.04	0.19	-0.15
5	0.03	0.07	-0.04	0.03	0.10	-0.07	0.03	0.10	-0.07	0.04	0.25	-0.21
6	0.08	4.41	-4.33	0.12	25.26	-25.14	0.12	25.26	-25.14	0.14	83.42	-83.28
7	0.01	11.28	-11.27	0.02	16.28	-16.26	0.02	16.28	-16.26	0.04	17.45	-17.42
8	0.05	0.12	-0.08	0.14	0.41	-0.27	0.14	0.41	-0.27	0.36	1.16	-0.79
9	0.02	0.25	-0.24	0.02	3.44	-3.42	0.02	3.44	-3.42	0.04	19.67	-19.62
10	0.07	0.24	-0.16	0.14	1.19	-1.05	0.14	1.19	-1.05	0.25	4.83	-4.59
11	1.56	0.62	0.95	1.27	3.97	-2.70	1.27	3.97	-2.70	1.79	21.16	-19.37
12	0.64	0.18	0.46	2.16	0.72	1.44	2.16	0.72	1.44	5.46	2.74	2.72
13	1.60	0.50	1.10	1.64	1.89	-0.24	1.64	1.89	-0.24	1.67	5.82	-4.15
14	1.41	0.59	0.83	1.43	2.05	-0.62	1.43	2.05	-0.62	13.47	6.49	6.98
15	0.63	15.31	-14.68	4.67	60.40	-55.73	4.67	60.40	-55.73	22.66	154.18	-131.53
16	0.10	0.19	-0.08	0.27	0.66	-0.39	0.27	0.66	-0.39	0.86	1.67	-0.82
17	5.96	1.46	4.49	25.82	6.95	18.86	45.68	12.44	33.23	66.20	18.30	47.90

6. Conclusion

Mobility data management and analysis have emerged in the last decade as a very active research domain addressing diverse issues such as clustering, integration, mining, indexing, persistence, and privacy. While previous research focused on processing raw trajectories collected from sensors and GPS devices, for example, recent research focuses on methods for enriching a trajectory with more contextualized and application-oriented information. Adding semantics to movement data brings enormous potential for the analysis of mobility-related phenomena.

Within this context, this work addresses an important issue that has not tackled by the literature: the persistence of MATs based on the pioneer data model in the literature called MASTER. The purpose here is to store and manipulate unlimited semantic data about mobility, where the representation and query of MATs are considered from a DB perspective. We materialize our solution, called *MASTERMobilityDB* as an extension of

a comprehensive raw trajectory DBMS called MobilityDB. New ADTs and an API dedicated to the representation and manipulation of MATs, respectively, were developed. An experimental evaluation comparing MASTERMobilityDB against the state-of-the-art in terms of DBMS for management of semantic trajectory data demonstrates the feasibility and superior time performance for querying MAT data.

We consider MASTERMobilityDB as a basis for several future projects related to MAT data management. One ongoing work in our DB research group is the support for MAT data integrity constraints, *i.e.*, to allow the definition and guarantee of some classes of integrity constraints over MAT data without introducing a big overhead in data management. Another study in progress is the support for MAT data analytics through the modeling of specific machine learning methods that could be applied over MAT entities, as well as the persistence of analysis' results.

References

Bogorny, V., Renso, C., Aquino, A., Siqueira, F., and Alvares, L. (2014). CONSTAnT - A Conceptual Data Model for Semantic Trajectories of MOs. *Trans. GIS*, 18(1):66–88.

Brandoli, B. et al. (2022). From Multiple Aspect Trajectories to Predictive Analysis: A Case Study on Fishing Vessels in the Northern Adriatic Sea. *GeoInformatica*, 26:551–579.

Ding, Z., Yang, B., Güting, R., and Li, Y. (2015). Network-Matched Trajectory-Based MOD: Models and Applications. *IEEE Trans. Intell. Transp. Syst.*, 16(4):1918–1928.

Feller, F. (2023). MasterMobilityDB - Uma Camada de Persistência e Manipulação para Trajetórias de Múltiplos Aspectos. Master's thesis, Universidade Federal de Santa Catarina.

Feller, F. and Mello, R. (2022). A Survey on Persistence Strategies for Semantically Enriched Trajectories. In *GEOINFO 2022*, pages 50–62. MCTIC/INPE.

Güting, R., Valdés, F., and Damiani, M. (2015). Symbolic Trajectories. *ACM Trans. Spatial Algorithms Syst.*, 1(2):7:1–7:51.

Gómez, L. et al. (2019). Analytical Queries on Semantic Trajectories using Graph Databases. *Transactions in GIS*, 23:1078–1101.

Mello, R., Bogorny, V., Alvares, L., Santana, L., Ferrero, C., Frozza, A., Schreiner, G., and Renso, C. (2019). MASTER: A Multiple Aspect View on Trajectories. *Transactions in GIS*, 23(4):805–822.

Mello, R., Schreiner, G. A., Alchini, C. A., dos Santos, G. G., Bogorny, V., and Renso, C. (2021). Dependency Rule Modeling for Multiple Aspects Trajectories. In *40th International Conference on Conceptual Modeling, ER*, volume 13011 of *Lecture Notes in Computer Science*, pages 123–132. Springer.

Noureddine, H. et al. (2021). A Hierarchical Indoor and Outdoor Model for Semantic Trajectories. *Transactions in GIS*, 26:214–235.

Pelekis, N., Frentzos, E., Giatrakos, N., and Theodoridis, Y. (2015). HERMES: A Trajectory DB Engine for Mobility-centric Applications. *Int. J. Knowl. Based Organ*, 5:19–41.

Spaccapietra, S., Parent, C., Damiani, M. L., de Macêdo, J. A. F., Porto, F., and Vangenot, C. (2008). A Conceptual View on Trajectories. *Data Knowl. Eng.*, 65(1):126–146.

Tamilmani, R. and Stefanakis, E. (2019). Modelling and Analysis of Semantically Enriched Simplified Trajectories Using Graph Databases. *Advances in GIScience of the ICA*, 1:1–8.

Torres, Y. et al. (2020). Stop-and-Move Sequence Expressions over Semantic Trajectories. *International Journal of Geographical Information Science*, 35:1–26.

Wannous, R., Malki, J., Bouju, A., and Vincent, C. (2013). *Time Integration in Semantic Trajectories Using an Ontological Modelling Approach*, volume 185, pages 187–198. Springer Berlin Heidelberg.

Xu, J., Lu, H., and Bao, Z. (2023). A Query Optimizer for Range Queries over Multi-Attribute Trajectories. *ACM Trans. Intell. Syst. Technol.*, 14(1).

Zimanyi, E. et al. (2020). MobilityDB: A Mobility Database Based on PostgreSQL and PostGIS. *ACM Transactions on Database Systems*, 45:1–42.