UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Flaris Roland Feller

MasterMobilityDB - Uma Camada de Persistência e Manipulação para Trajetórias de Múltiplos Aspectos

Florianópolis/SC/Brasil
2023

Flaris Roland Feller

MasterMobilityDB - Uma Camada de Persistência e Manipulação para Trajetórias de Múltiplos Aspectos

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do título de mestre em Ciência da Computação.
Orientador: Prof. Ronaldo Santos Mello, Dr.

Florianópolis/SC/Brasil
2023

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Feller, Flaris Roland
MASTERMobilityDB – Uma Camada de Persistência e Manipulação para Trajetórias de Múltiplos Aspectos / Flaris Roland Feller ; orientador, Ronaldo dos Santos Mello, 2023.
93 p.

Dissertação (mestrado) – Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2023.

Inclui referências.

1. Ciência da Computação. 2. Banco de Dados. 3. Mobilidade. 4. Big Data. 5. Trajetórias de Múltiplos Aspectos. I. Mello, Ronaldo dos Santos. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Flaris Roland Feller

MasterMobilityDB - Uma Camada de Persistência e Manipulação para Trajetórias de Múltiplos Aspectos

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof.(a) Carina Friedrich Dorneles, Dra.
Instituição Universidade Federal de Santa Catarina - UFSC

Prof.(a) Jônata Tyska Carvalho, Dr.
Instituição Universidade Federal de Santa Catarina - UFSC

Prof.(a) Tarlis Tortelli Portela, Dr.
Instituição Instituto Federal de Ciência Tecnologia e Educação do Paraná - IFPR

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Ciência da Computação.

ICP-Edu
Documento assinado digitalmente
Marcio Bastos Castro
Data: 12/03/2024 17:39:47-0300
CPF: ***.707.720-**
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Programa de Pós-Graduação

GOV.BR
Documento assinado digitalmente
RONALDO DOS SANTOS MELLO
Data: 12/03/2024 18:13:20-0300
CPF: ***.693.660-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Ronaldo Santos Mello, Dr.
Orientador

Florianópolis/SC/Brasil, 2023.

ICP-Edu
Documento assinado digitalmente
Flaris Roland Feller
Data: 12/03/2024 18:17:28-0300
CPF: ***.907.709-**
Verifique as assinaturas em https://v.ufsc.br

RESUMO

Os bancos de dados de objetos móveis (MODs em inglês) têm sido estudados há pelo menos 20 anos na área de banco de dados. Entretanto, a partir da evolução da captura de dados espaciais bem como da integração de dados provenientes fontes heterogêneas, gerou-se um acúmulo de volumosas coleções de dados de objetos em movimento, como automóveis, pessoas e animais. Estas coleções, conhecidas como trajetórias, coletadas de sensores e dispositivos habilitados para GPS, podem ser enriquecidas com múltiplos aspectos semânticos gerando trajetórias de múltiplos aspectos ou MATs. Tratam-se de objetos complexos, heterogêneos e de rápido crescimento. O gerenciamento destas trajetórias introduz novos desafios para MODs: a capacidade de gerenciar um grande volume de dados, o suporte a consultas espaço-temporais de baixa latência, a integração de dados complexos e heterogêneos com as MATs e a necessidade de alta taxa de transferência de inserção. De modo a superar tais desafios, este trabalho apresenta a proposta de uma camada de persistência de MATs, implementada a partir do MOD MobilityDB (ZIMANYI *et al.*, 2020), que irá suportar a manipulação de MATs. Além disso, este trabalho também trata a persistência de trajetórias representativas as quais sumarizam comportamentos comuns em MATs e das regras de dependência que representam padrões descobertos a partir da análise destas trajetórias. Por fim, são apresentados as validações dos requisitos e os testes de desempenho demonstrando a viabilidade do MASTERMobilityDB para o gerenciamento de MATs e que os objetivos propostos foram alcançados. Através de um comparativo do MasterMobilityDB com o BD estado-da-arte Secondo, tratando-se com diversas classes de consultas, é demonstrado que as consultas de trajetória são expressas no MasterMobilityDB mais naturalmente através das funcionalidades desenvolvidas do que no Secondo, e têm melhor desempenho em muitos casos típicos. Nenhuma proposta de propósito geral similar, em termos de persistência de dados para MATs, foi encontrada na literatura e na indústria.

Palavras-chave: *Moving Object Databases*. Trajetória. Múltiplos Aspectos. Espaço-temporal.

ABSTRACT

Moving Object Databases (MODs) have been studied for at least 20 years in the database field. However, with the evolution of spatial data capture as well as the integration of data from heterogeneous sources, an accumulation of voluminous collections of geographic trajectory data has been generated. These collections, known as multi-aspect trajectories, collected from sensors and GPS-enabled devices, are complex, with multiple semantic aspects, heterogeneous and rapidly growing. Managing these trajectories introduces new challenges for MODs: the ability to manage a large volume of data, support for low-latency spatiotemporal queries, integration of complex and heterogeneous data with semantic trajectories, and the need for high throughput insertion. This work presents the proposal for a multi-aspect trajectory persistence layer, implemented from MobilityDB MOD (ZIMANYI *et al.*, 2020), that supports the manipulation of multi-aspect trajectories. Furthermore, this work also deals with the persistence of representative trajectories which summarize common behaviors in multi-aspect trajectories and dependency rules that represent patterns discovered from the analysis of these trajectories. Through a comparison of MasterMobilityDB with the state-of-the-art database Secondo, dealing with several query classes, it is demonstrated that trajectory queries are expressed in MasterMobilityDB more naturally through the developed functionalities than in Secondo, and have better performance in many typical cases. No similar proposal, in terms of data persistence for multi-aspect trajectories, has been found in the literature and industry.

Keywords: Moving Object Databases. Trajectory. Multiple Aspect. Spatio-temporal.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

De acordo com (LI *et al.*, 2016), devido ao uso generalizado de dispositivos habilitados para GPS, como sistemas de navegação de veículos, *smartphones* e *wearables*, o registro de dados de posição tornou-se muito fácil e grandes quantidades desses dados são recolhidos todos os dias. Em resposta a isso, o campo de pesquisa de gerenciamento de dados de trajetórias, incluindo bancos de dados (BDs) de objetos móveis (MOD), tem estado muito ativo nos últimos 20 anos.

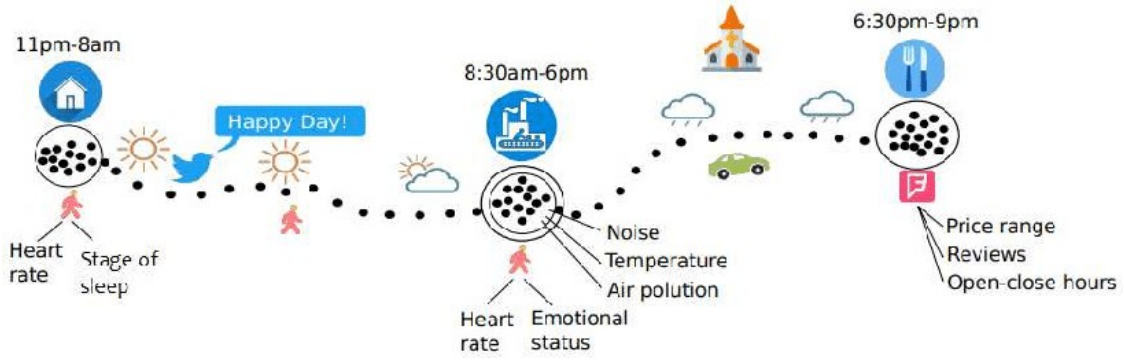
Segundo (GÜTING; SCHNEIDER, 2005), uma trajetória descreve o movimento de uma entidade, por exemplo, uma pessoa, um veículo ou um animal, ao longo do tempo. Em um nível mais baixo de abstração, é uma sequência de posições no espaço organizadas por um atributo de tempo (*timestamp*), conhecida também como trajetória bruta, correspondente à forma como os dados são gravados pelos dispositivos. Em um nível mais alto de abstração, é uma função contínua do tempo para o espaço 2D ou 3D que pode ser representada por um tipo de dados abstrato "ponto em movimento".

De acordo com (GÜTING; VALDÉS; DAMIANI, 2015), considerando que o objetivo geral do enriquecimento semântico de trajetórias é descobrir qualquer tipo de fenômeno interessante em conjuntos de dados de trajetórias, uma consideração importante é associar algum significado a uma trajetória como um todo ou partes dela. Por exemplo, para um turista, em vez de estar em alguma coordenada geográfica no Brasil por um intervalo de tempo deseja-se entender se ele está visitando um museu ou jantando em um restaurante. Para um carro, queremos estar cientes de que ele esteve em um engarrafamento durante um determinado período. Para um animal em observação gostaríamos de entender que este é um comportamento de migração, ou um pássaro voando em uma revoada. Para uma pessoa que se desloca, gostaríamos de saber se ela está andando, passeando de bicicleta ou usando um ônibus.

Pode-se observar na Figura 1 que uma trajetória pode ser um objeto complexo com várias dimensões que são contextuais ao movimento e heterogêneos na forma, que de acordo com (SANTOS MELLO *et al.*, 2019) são definidos como aspectos. Quanto mais aspectos, mais completa é a representação do movimento de um objeto, e mais informações úteis e interessantes pode-se inferir sobre ele. Esta representação de trajetória semanticamente enriquecida é denominada *trajetória de múltiplos aspectos*.

Uma trajetória de múltiplos aspectos (em inglês, MAT) não é apenas uma trajetória semântica representada como uma sequência de paradas e movimentos como descrito em (ALVARES *et al.*, 2007). Os aspectos necessitam de uma representação mais específica pois são dados de naturezas diferentes e necessitam de atributos para descrevê-los dentro do contexto da aplicação. A Figura 1 apresenta um exemplo de MAT da rotina de um cidadão, transitando da casa para o trabalho, que é enriquecida com 5 aspectos: lugares visitados, condições meteorológicas, meios de transporte,

Figura 1 – Um exemplo de trajetória multiaspecto.



Fonte: Adaptado de (SANTOS MELLO *et al.*, 2019)

postagens nas redes sociais e saúde. Cada aspecto é descrito por seus próprios atributos. Por exemplo, os lugares visitados têm uma posição espacial, uma categoria, uma classificação e um preço. A condição meteorológica tem uma posição espacial, uma descrição (por exemplo, ensolarado, nublado) e uma temperatura, e o aspecto saúde tem a frequência cardíaca. Dentre esses aspectos e seus atributos, observamos que os atributos classificação e preço estão relacionados ao ponto de interesse (POI), uma vez que se referem especificamente à categoria POI. Os atributos temperatura e descrição referem-se às condições meteorológicas e não ao POI. Da mesma forma, a frequência cardíaca do objeto está relacionada ao objeto em movimento e não ao POI nem às condições meteorológicas.

Desde a segunda metade dos anos 2000, diversos modelos de representação de dados para enriquecimento semântico de trajetórias têm sido propostos, tais como, o modelo *Stops and Moves* (SPACCAPIETRA *et al.*, 2008), o qual segmenta uma trajetória de acordo com os episódios (paradas e movimentos). Derivado deste foi desenvolvido o modelo CONSTANT, em que uma trajetória pode ser associada a um conjunto limitado de aspectos pré-definidos: as atividades desempenhadas pelo objeto, o meio de transporte, os POIs visitados, o objetivo da viagem e alguns padrões específicos de comportamento. O modelo de trajetórias simbólicas proposto por (GÜTING; VALDÉS; DAMIANI, 2015) traz a representação destes objetos como subtrajetórias anotadas com rótulos simples indexados por intervalos de tempo. Para representar, por exemplo, os POIs visitados e o meio de transporte, a mesma trajetória deve ser segmentada por POIs e por meio de transporte. Este trabalho adota o modelo MASTER (SANTOS MELLO *et al.*, 2019), pois o mesmo generaliza o estado da arte para representação de MATs. O MASTER é mais abrangente no quesito variedade dos dados uma vez que trata aspectos não apenas como rótulos semânticos simples, mas também podem ser objetos complexos e/ou informações heterogêneas intrinsecamente associadas aos rastros físicos dos objetos em movimento.

Do modelo de representação MASTER foram derivadas duas extensões, as

quais são suportadas neste trabalho: a) MAT-SG (LAGO MACHADO; MELLO; BOGORNY, 2022), para trajetórias representativas e b) MASTER-DR (SANTOS MELLO *et al.*, 2021), para regras de dependência.

A primeira, MAT-SG, apresenta um método para sumarização de MATs baseado em uma grade de células. Nesta abordagem determina pontos representativos, gerados a partir de um conjunto de trajetórias, com valores representativos para cada aspecto e cujo sequenciamento gera trajetórias representativas. Este trabalho suporta a persistência de trajetórias representativas e trajetórias não-representativas.

A segunda, MASTER-DR, trata da representação de padrões descobertos a partir da análise de MATs. Estes padrões podem estar relacionados a uma trajetória como um todo, pontos de trajetória ou objeto em movimento. Neste contexto, uma regra de dependência é um padrão que especifica dependências complexas entre valores de atributos pertencentes a uma ou mais entidades do mundo real, como detalhado na seção 2.5.

Visando implementar a persistência de trajetórias brutas e/ou semanticamente enriquecidas, a pesquisa em BDs de trajetória de objetos em movimento gerou várias implementações de MODs e algumas delas já passaram do estágio de protótipo, oferecendo versões estáveis e bastante amadurecidos como é o caso do Secondo (GÜTING; BEHR; DÜNTGEN, 2010), o Hermes (PELEKIS *et al.*, 2015) e mais recentemente o MobilityDB (ZIMÁNYI *et al.*, 2019). Este último é construído sobre o PostgreSQL e PostGIS, que são, respectivamente, um sistema de BD de código aberto amplamente utilizado e sua extensão de BD espacial. Esses dois sistemas são ativamente apoiados por uma grande comunidade de empresas e indivíduos. O MobilityDB é uma extensão do PostgreSQL escrita na linguagem C e possui uma série de funcionalidades para o tratamento de trajetórias brutas, e entende-se que ele é a melhor opção na atualidade para a implementação do MASTER. Este trabalho estende o MobilityDB, cuja justificativa pela escolha é dada na seção 4.2.

1.1 OBJETIVOS E CONTRIBUIÇÃO

O objetivo geral desta dissertação é o projeto e o desenvolvimento de uma camada de persistência chamada *MASTERMobilityDB*, visando o armazenamento, o desempenho e a manipulação eficiente de MATs e suas extensões (MAT-SG e MASTER-DR) de acordo com os modelos lógico e conceitual definidos para o modelo MASTER (SANTOS MELLO *et al.*, 2019).

Do objetivo principal deste trabalho deriva-se os seguintes objetivos específicos:

a) Implementar uma camada de acesso sobre o MobilityDB que suporta a persistência e a manipulação de MATs.

b) Dentro desta camada é necessário desenvolver um conjunto de tipos abstra-

tos de dados, derivados dos tipos oferecidos pelo MobilityDB para permitir a representação de MATs neste banco de dados.

c) Estender ou implementar operações e funções que atuam sobre estes tipos abstratos para o armazenamento e a recuperação eficiente de MATs.

d) Prover rotinas utilitárias para apoiar os processos de extração, limpeza e carga (ETL).

e) Utilizar os recursos do MobilityDB como particionamento e índices especializados para aumentar o desempenho no armazenamento e na busca de MATs.

f) Implementar a persistência das trajetórias representativas geradas a partir do processo de sumarização de trajetórias (LAGO MACHADO; MELLO; BOGORNY, 2022), de modo a diminuir a complexidade das tarefas de data-mining e de análise utilizando o banco de dados implementado.

g) Implementar a persistência de regras de dependência (SANTOS MELLO *et al.*, 2021), resultantes de padrões descobertos via a análise das MATs. Essa é uma recente extensão do modelo MASTER.

A principal contribuição é a implementação do suporte de armazenamento e acesso ao modelo MASTER que, como foi comentado anteriormente, supera as limitações dos atuais modelos de enriquecimento semântico de trajetórias. Como ainda não existe uma solução de persistência de propósito geral que lida com MATs, este projeto pioneiro irá constituir a base para futuros trabalhos de pesquisa em BDs para estas trajetórias.

1.2 METODOLOGIA

Para um melhor tratamento dos objetivos e melhor apreciação deste trabalho é necessária a realização de uma pesquisa bibliográfica. Esta pesquisa implica que os dados e informações importantes para a realização das tarefas sejam obtidos a partir de abordagens já trabalhadas por outros pesquisadores. Sendo assim, foram compreendidos na pesquisa: livros, artigos, *surveys* e documentos eletrônicos, de modo a adquirir conhecimento sobre MODs e modelos de representação de MATs.

Esta metodologia compreende 5 etapas: (i) Um levantamento do estado da arte sobre BDs para objetos móveis (MODs) implementados em diversas arquiteturas de BDs e diferentes modelos de representação de trajetórias semânticas; (ii) A elaboração do projeto do BD; (iii) A implementação da solução de uma extensão do BD MobilityDB; (iv) Avaliação dos resultados; e (v) Documentação.

1.3 ORGANIZAÇÃO DO DOCUMENTO

Este trabalho está organizado em mais 4 capítulos. O capítulo 2 apresenta a fundamentação teórica, incluindo conceitos como tipos de trajetórias, objetos móveis e o modelo de dados MASTER. No capítulo 3 é detalhada a revisão sistemática dos trabalhos relacionados ao tema persistência de trajetórias semanticamente enriquecidas, e uma análise dos mesmos. O capítulo 4 apresenta a proposta desta dissertação e o capítulo 5 descreve as atividades futuras planejadas para essa dissertação.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos necessários para compreensão do restante do trabalho. Inicialmente aborda-se objetos móveis. Na sequência introduz-se a noção de trajetória de objetos móveis e aspectos. Em seguida é apresentado o modelo de dados MASTER para representação de trajetórias de múltiplos aspectos e o BD MobilityDB, que são os pilares deste trabalho. Por último, aborda-se a sumarização de dados de trajetórias em trajetórias representativas e as regras de dependência para MATs. Esses 2 últimos conceitos também são inclusos na estratégia de persistência proposta neste trabalho.

2.1 OBJETOS MÓVEIS

Segundo (GÜTING, 2018) o termo objeto móvel enfatiza o fato de que geometrias de representação de objetos que se movem no espaço-tempo podem mudar continuamente. Pode-se distinguir entre objetos em movimento para os quais apenas a posição dependente do tempo é de interesse e aqueles para os quais também a forma e a extensão são relevantes e podem mudar ao longo do tempo. O primeiro pode ser caracterizado como pontos móveis, o segundo como regiões móveis. Por exemplo, pontos em movimento podem representar pessoas, veículos (como carros, caminhões, navios ou aviões) ou animais. As regiões em movimento podem ser furacões, incêndios florestais, propagação de doenças epidêmicas, etc. Os dados de pontos em movimento podem ser capturados, por exemplo, por dispositivos GPS ou etiquetas RFID. Já dados de região em movimento podem resultar de sequências de processamento de imagens de satélite, por exemplo. Pontos móveis e regiões móveis podem ser disponibilizados como tipos abstratos de dados.

Güting e Schneider (GÜTING; SCHNEIDER, 2005) afirmam que enquanto mudanças discretas ocorrem em qualquer tipo de entidade espacial, mudanças contínuas parecem mais relevantes para o ponto e a região. Portanto, um ponto em movimento é a abstração básica de um objeto físico se movendo no plano ou em um espaço dimensional superior, para o qual apenas a posição, mas não a extensão, é relevante. A abstração da região móvel descreve uma entidade no plano que muda sua posição, bem como sua extensão e forma, ou seja, uma região móvel pode não apenas se mover, mas também crescer e encolher.

2.2 BIG DATA ESPACIAL

De acordo com (MIR *et al.*, 2018), o termo *Big Data* significa uma enorme quantidade de dados que são complexos de analisar. No contexto de dados espaciais temos vários exemplos de Big Data, como mapas de localização, relatórios investigati-

vos, dados coletados de milhares de sensores colocados em vários locais, mudanças climáticas e relatórios meteorológicos detalhados, e assim por diante. A enorme disseminação de dispositivos de detecção de baixo custo e a queda dramática no custo de armazenamento dos dados resultaram na extração de informações valiosas para os mais diversos fins, como planejamento urbano, segurança de tráfego, sensoriamento remoto, estudos climáticos e sistema de transporte inteligente. Boa parte destas aplicações lida com dados de trajetórias de objetos em movimento que são altamente relevantes para o big data espacial porque adicionam uma dimensão temporal à informação espacial, fornecendo entendimento sobre o comportamento dinâmico dos objetos no espaço.

O mesmo trabalho (MIR *et al.*, 2018) também afirma que o manuseio de *Big Data espacial* torna-se mais significativo e complexo se os objetos (dados geradores) forem altamente móveis e localizados espacialmente. Devido aos avanços atuais no campo da tecnologia da informação principalmente com o fornecimento de conectividade à internet quando e onde for necessário, o Big Data Espacial desponta como uma das áreas de pesquisa mais importantes e interessantes. Os termos Big Data e Big Data Espacial são fáceis de diferenciar. Por exemplo, posts e tweets no Facebook e Twitter, respectivamente, podem ser categorizados como Big Data, enquanto posts/tweets geolocalizados de diferentes partes do mundo se enquadram como Big Data Espacial. Outro exemplo de Big Data Espacial são os dados coletados de vários"nós sensores" localizados espacialmente em uma grande região geográfica. Esses dados devem ser armazenados, monitorados e processados em tempo real para lidar com situações como recuperação de desastres, possibilidade de propagação de incêndios em uma selva e controle de temperatura.

Diversas aplicações podem se beneficiar da análise do Big Data Espacial. Alguns exemplos são:

- *Mobile/Social Marketing*, onde, com o advento do GPS integrado em dispositivos móveis para permitir informações de localização para os usuários, os aplicativos de mídia social possibilitam que usuários publiquem conteúdo com informações geográficas, gerando assim uma grande quantidade de dados georreferenciados sobre os quais ferramentas mineram padrões de mobilidade individual que podem ser utilizados, por exemplo, por sistemas de recomendação, para indicar eventos culturais na rota dos usuários, oferta de restaurantes próximos baseados nos gostos dos clientes e captura ou inferência dos interesses de viagem dos usuários a partir de seu comportamento de consumo em pacotes de viagem.

- *Saúde-móvel*, por meio do uso de aplicativos instalados em smartphones e dispositivos vestíveis sensoriais como Google Glass, Apple iWatch e Samsung Gear, entre outros, que possuem funcionalidades para monitorar bati-

mentos cardíacos, pulso e frequência respiratória dos usuários. Os dados coletados de dispositivos ainda podem ser enriquecidos com dados de websites agregadores de informações de saúde como PatientsLikeMe, Health Tracking Network e Healthmap. Esta combinação de dados permite a um médico conhecer o histórico clínico completo de um paciente a partir de seu dispositivo vestível bem como avaliar se o mesmo percorreu areas infectadas por doenças. Isto pode ajudar em uma situação de emergência.

- *Desastres naturais*, onde novos fluxos de informação, como vídeos, fotografias e textos de mídia social, bem como outras fontes abertas, estão redefinindo a consciência da situação durante emergências. Mensagens de mídia social combinadas com referências geográficas contendo informações espaciais e temporais são denominadas informações geográficas voluntárias que desdobram o poder dos "cidadãos como sensores" para fornecer dados atualizados em tempo real que podem ser usados na resposta a desastres, na avaliação oportuna de danos e na realização eficiente de operações de resgate e socorro.

2.3 TRAJETÓRIAS

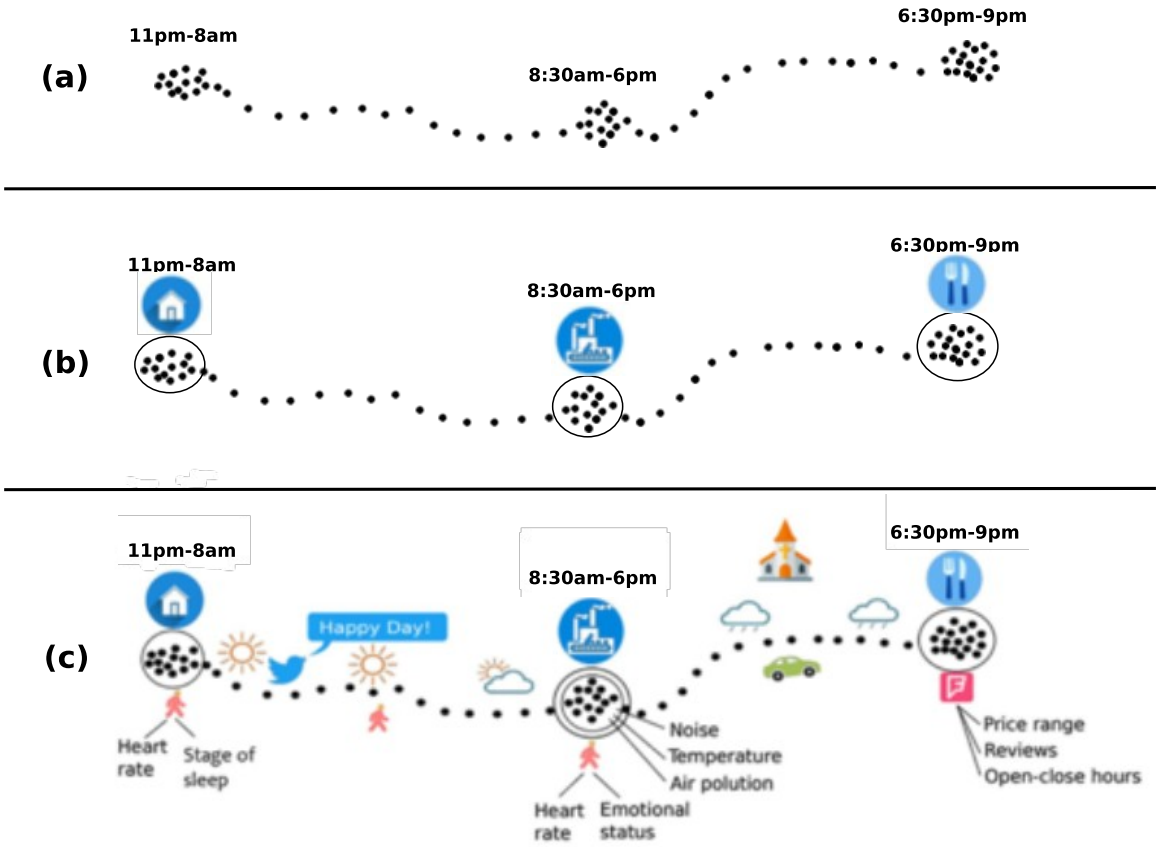
O principal tipo de dado abordado neste trabalho são as trajetórias de objetos móveis. Segundo (RENSO; SPACCAPIETRA; ZIMÁNYI, 2013), atualmente, é comum o uso de tecnologias que fornecem serviços de geolocalização, como dados gerados por rastreamento em larga escala relacionados à movimentação de um determinado objeto. Esses dados têm sido cada vez mais coletados para mineração, análise e tomada de decisão.

De acordo com (PORTELA; CARVALHO; BOGORNY, 2022), o conceito de trajetória evoluiu ao longo do tempo. A *trajetória bruta* (GÜTING; SCHNEIDER, 2005) de um objeto em movimento é uma sequência de pontos no espaço geográfico ao longo do tempo, consistindo, assim, de duas dimensões: espacial e temporal. Por volta de 2007 (ALVARES *et al.*, 2007), surgiu o conceito de *trajetória semântica*, em que uma terceira dimensão é agregada em trajetórias de dados, ou seja, uma trajetória espaço-temporal bruta *(x, y, t)* é enriquecida com informações semânticas em princípio paradas e movimentos. Com a presença onipresente dos dispositivos móveis, a popularização das redes sociais e a utilização de redes de sensores e dispositivos IoT, grandes volumes de dados de mobilidade estão a ser criados e recolhidos diariamente. As trajetórias semânticas, também chamadas de Trajetórias de Múltiplos Aspectos, envolvem o enriquecimento de trajetórias brutas com dimensões heterogêneas adicionais como condição climática, ponto de interesse (POI), preço, humor do usuário, dia da semana e assim por diante.

) du-
oral
mân-
ho

a

trajetória de múltiplos aspectos (c).



Fonte: Adaptado de (SANTOS MELLO *et al.*, 2019).

Com o uso comum da IoT e das mídias sociais, foi possível enriquecer as trajetórias com um número ilimitado de informações semânticas. Quando a trajetória inteira, ou alguns de seus pontos, está associada a muitos contextos semânticos, ou aspectos, essas trajetórias são conhecidas como *trajetórias de múltiplos aspectos (MAT)* (SANTOS MELLO *et al.*, 2019). Assim sendo, uma MAT é composta por três dimensões (espacial, temporal e semântica), mas essa terceira dimensão pode representar aspectos múltiplos e heterogêneos. Um aspecto pode estar relacionado a um objeto móvel, a trajetória inteira, ou qualquer ponto da mesma e pode conter qualquer tipo de dados, desde anotações simples a objetos complexos.

A Figura 2(c) mostra a trajetória bruta enriquecida com informações como o meio de transporte utilizado pelo indivíduo, postagens em redes sociais, condições

meteorológicas, informações de saúde e assim por diante. Este exemplo destaca que uma trajetória de múltiplos aspectos é um objeto complexo cujos atributos podem ter conteúdos simples (dados alfanuméricos) ou complexos (geometria, imagem, som, etc), que exigem um tratamento de persistência diferenciado, de acordo com os contextos descritos.

2.4 MODELO DE DADOS MASTER PARA TRAJETÓRIAS MULTI-ASPECTO

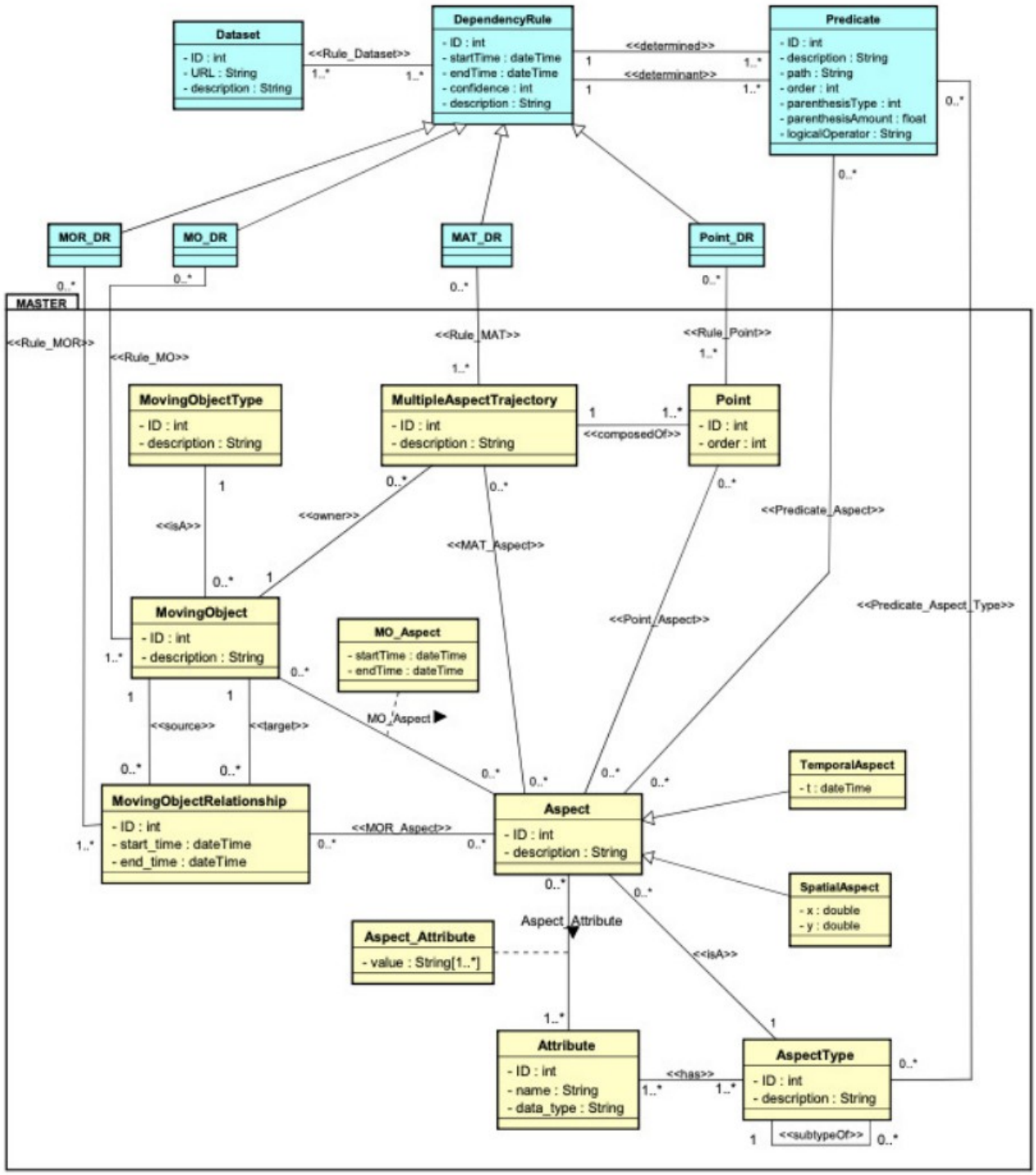
Conforme mencionado no capítulo ao longo dos últimos anos alguns modelos de trajetórias semânticas vêm sendo propostos. Eles foram concebidos como abordagens gerais que podem ser aplicadas a qualquer domínio de aplicação envolvendo objetos em movimento. Entretanto, vários conceitos das aplicações da Big Data espacial não podem ser modelados de forma simples e direta como padrões de paradas e movimentações, como por exemplo, táxis entrando ou saindo de uma parada para realizar viagens ou a associação de variáveis ambientais com um determinado padrão. O modelo CONSTANT é limitado a um subconjunto de aspectos relacionados a subtrajetórias ou a toda a trajetória (por exemplo, atividade realizada pelo objeto, meio de transporte e objetivo da viagem) e o modelo de trajetórias simbólicas é limitado a rótulos simples para um único aspecto.

Para sanar as limitações desses modelos anteriores, recentemente foi proposto o modelo MASTER (SANTOS MELLO *et al.*, 2019), um modelo mais flexível e expressivo que permite a representação de múltiplas dimensões semânticas. Em particular, ele introduz o conceito de *aspecto*, que consiste em um fato do mundo real que é relevante para a análise de dados da trajetória. Diferentes tipos de aspectos são modelados: *(i) aspectos voláteis*, geralmente associados aos pontos de trajetória, uma vez que eles variam durante o movimento do objeto; *(ii) aspectos de longo prazo*, que não mudam durante toda a trajetória e, portanto, estão associados a toda a trajetória; *(iii) aspectos permanentes*, que permanecem durante toda a vida de um objeto, portanto são conectados ao objeto em movimento e não à trajetória. Com base nesta noção, uma MAT é definida como uma sequência de pontos espaço-temporais de um objeto em movimento com um possível conjunto de aspectos associados a ela.

O modelo conceitual de dados do MASTER é mostrado na Figura 3. Conforme comentado anteriormente, um *aspecto* é um fato relevante para uma análise de dados das trajetórias e é caracterizado por um *tipo de aspecto*, que define os metadados e atributos do aspecto. Um tipo de aspecto também pode derivar outros tipos de aspectos. Um aspecto suporta números, faixas de valores, texto, geometrias ou objetos complexos. Por exemplo, um tipo de aspecto *hotel* pode ter os seguintes atributos: *coordenadas geográficas, endereço, estrelas, tipos de quartos* e *instalações*. Já um aspecto pertencente a este tipo pode ser: *"Il Campanario Resort"*, com os seguintes pares atributo-valor: *coordenadas geográficas, –27,439771, –48,500802; endereço,*

"Avenida Búzios, Florianópolis"; estrelas, 5; tipos de quartos, {"suíte", "suíte júnior"}; instalações, {"academia", "piscina", "restaurante", "bar", "serviço de praia"}.

Figura 3 – O modelo conceitual para trajetórias de múltiplos aspectos.



Fonte: Adaptado de (SANTOS MELLO *et al.*, 2021)

Uma *MAT*, por sua vez, é uma sequência P de pontos de um objeto móvel, um conjunto de aspectos de longo prazo e uma descrição. Cada ponto $p_i \in P$ é uma

tupla contendo as coordenadas x e y, a dimensão tempo e um conjunto de aspectos relacionados ao ponto. Ainda, um *objeto móvel* (MO) é uma entidade que pode se mover no espaço e no tempo e possui uma descrição, um conjunto de aspectos e um tipo que o categoriza. O modelo permite também a definição de um ou mais relacionamentos entre dois objetos móveis que podem conter aspectos, como por exemplo o tipo do relacionamento (amizade, familiar, etc)

As entidades do MASTER são melhor detalhadas no Capítulo 4 onde é apresentada a estratégia de implementação.

2.5 REGRAS DE DEPENDÊNCIA DE MAT

Uma extensão ainda mais recente do MASTER introduziu a modelagem das regras de dependência como padrões descobertos sobre dados presentes em *datasets* de MATs (entidades destacadas em azul) e que envolvem aspectos (SANTOS MELLO *et al.*, 2021). Uma regra de dependência (em inglês DR), é similar a uma regra de associação composta por uma parte determinante e uma parte determinada, sendo cada uma dessas partes composta por um conjunto de predicados. Por exemplo, suponhamos que um algoritmo de mineração de dados de trajetórias semânticas, ao analisar um dataset de pessoas se locomovendo por automóvel encontrou a seguinte regra de dependência: "velocidade média > 80 AND (dia da semana = ’Sábado’ OR dia da semana = ’Domingo’) AND (período = ’0-6’ OR período = ’18-24’) => objetivo = ’Lazer’, com percentual de confiança de 72%". Neste caso os predicados seriam as expressões lógicas conectadas pelos operadores AND ou OR e o tipo do predicado seria determinante se estiver a esquerda do sinal "=>"e determinado se estiver a direita.

Segundo (SANTOS MELLO *et al.*, 2021), como vários padrões podem ser encontrados nos MATs, eles podem ser valiosos dependendo de sua precisão e vida útil temporal. Assim, associamos uma confiança e um tempo de validade a cada DR. Na Figura 3 a modelagem conceitual das regras de dependência pode ser visualizada nas entidades em azul as quais se relacionam com o restante do modelo MASTER. A seguir são descritas as principais entidades do MASTER-DR.

a) DependencyRule: é um padrão de dados descoberto em um conjunto de conjuntos de dados de MATs, com uma descrição *description*, uma confiança *confidence* e um tempo de validade (*startTime* e *endTime*). Associam-se à entidade DependencyRule conjuntos de predicados PRE e PÓS que especifica, respectivamente, suas partes determinante e determinada e suas especializações: MOR_DR (relacionamento entre objetos), MO_DR (objetos móveis), MAT_DR (MATs) e Point_DR (Pontos), que guardam relacionamentos do tipo muitos-para-muitos com as entidades do MASTER. As especializações da entidade DR não são exclusivas, pois um mesmo DR pode servir,

por exemplo, como um padrão para um MAT inteiro em um contexto e um padrão para alguns pontos do MAT em outro contexto. Um mesmo padrão relacionado ao clima, por exemplo, pode ocorrer durante toda a trajetória ou apenas em alguns de seus pontos.

b) Dataset: é uma fonte de dados de MATs com uma descrição description, uma URL com sua localização e um conjunto de DRs descoberto sobre ela.

c) Predicate: faz parte de uma condição determinante ou determinada da DR. Possui uma descrição *description*, a DR dona do predicado, o tipo de condição do DR onde está inserido, (determinante - 0; determinado - 1) (*condition_type*), a expressão do caminho (SANTOS MELLO *et al.*, 2021) que define o predicado (*path*), sua ordem dentro da condição (*order*), os conjuntos não obrigatórios de ocorrências de aspectos e tipos de aspecto nos quais o predicado é válido e atributos opcionais que denotam se o predicado é precedido ou não por um parêntese aberto (0) ou sucedido por um parêntese fechado (1) (*parenthesisType*), a quantidade desse tipo de parêntese (*parenthesisAmount*) e se o predicado é precedido por um operador lógico: OR (0) ou AND (1) (*logicalOperator*).

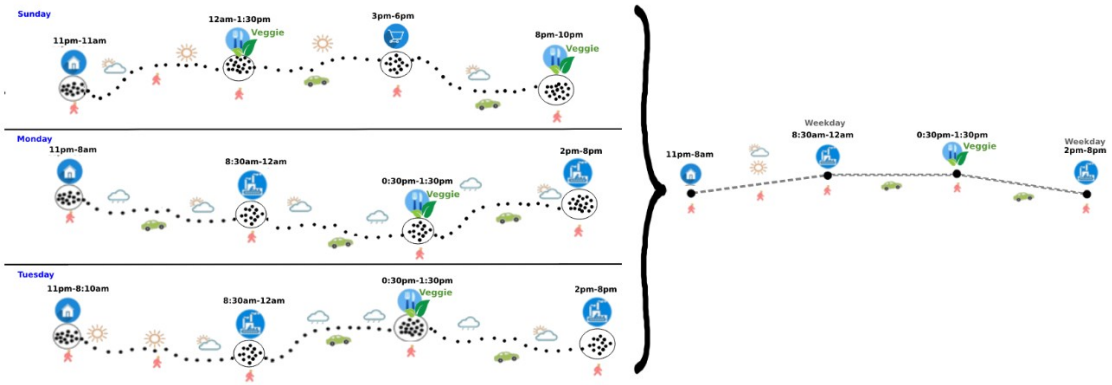
2.6 SUMARIZAÇÃO DE DADOS DE TRAJETÓRIAS

Sumarização de dados é um processo que gera informações representativas de um conjunto de dados. De acordo com (LEE; HAN; WHANG, 2007), uma trajetória representativa é uma trajetória imaginária que denota o comportamento principal de um conjunto de trajetórias. Já para (PANAGIOTAKIS *et al.*, 2012), uma trajetória representativa pode ser definida de diferentes maneiras, de acordo com o foco considerado, como interesse, densidade, frequência e distância entre pares.

De acordo com (LAGO MACHADO; MELLO; BOGORNY, 2022), a mineração e análise de dados de MATs são complexas e não triviais devido ao volume de dados e à heterogeneidade. Uma solução para esses problemas é a sumarização de dados para gerar dados representativos, considerando as dimensões do movimento: espaço, tempo e semântica. Esta dissertação de mestrado dá apoio a este trabalho em nível de doutorado, também em desenvolvimento, no sentido de persistir trajetórias multiaspecto (MATs) sumarizadas. Neste trabalho os autores propõem um método baseado em grade para sumarização de dados de trajetória de múltiplos aspectos denominado MAT-SG. Ele traz diversas contribuições: (i) segmentação da trajetória em uma grade espacial de acordo com a dispersão dos pontos de dados; (ii) expressa um conjunto de dados de trajetória por uma sequência de pontos representativos com valores representativos para cada dimensão, considerando suas particularidades de tipo de dados.

Na Figura 4 é apresentado um exemplo de trajetória representativa de um indivíduo (LAGO MACHADO; MELLO; BOGORNY, 2022). No lado direito, seus MATs estão resumidos em um MAT representativo que apresenta as ações realizadas com frequência, como trabalhar durante a semana e almoçar em um restaurante vegetariano entre 13h30 e 13h30. Trata-se de um padrão de comportamento que pode ser útil para diversas tomadas de decisão, como por exemplo, para um sistema de recomendação encontrar outros POIs para essa pessoa (outros restaurantes vegetarianos, por exemplo) ou sugerir ações visando a sua saúde ou satisfação.

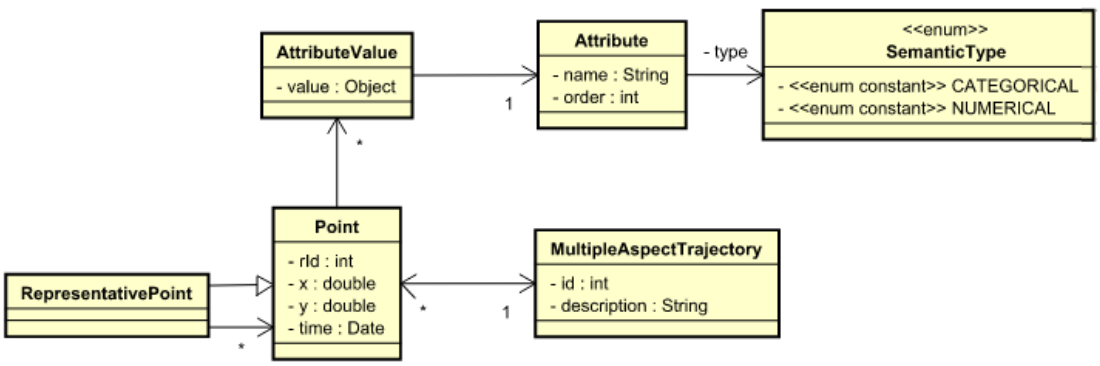
Figura 4 – Exemplos de MATs (à esquerda) com a representativa (à direita)



Fonte: Adaptado de (LAGO MACHADO; MELLO; BOGORNY, 2022)

O modelo conceitual do MAT-SG, mostrado na Figura 5, é uma extensão do modelo MASTER, para tratamento das trajetórias representativas. Além dos conceitos de trajetórias e objetos em movimento já tratados no MASTER, podem ser identificados os pontos representativos na entidade *RepresentativePoint*. Esta entidade denota o ponto representativo e a população de pontos que ele representa. A sequência de pontos constitui uma trajetória representativa.

Figura 5 – Modelo conceitual do MAT-SG



Fonte: Adaptado de (LAGO MACHADO; MELLO; BOGORNY, 2022)

2.7 O BANCO DE DADOS MOBILITYDB

Segundo (ZIMANYI *et al.*, 2020) o MobilityDB é um MOD que se baseia em PostgreSQL e PostGIS. Ele estende o sistema de tipos do PostgreSQL/PostGIS com tipos de dados abstratos (ADTs) para representar dados de objetos em movimento. Ele define, por exemplo, o tipo temporal *tgeompoint* para representar objetos de pontos geométricos em movimento.

Os tipos do MobilityDB são totalmente integrados à plataforma para se beneficiar dos recursos de gerenciamento de dados existentes e das melhorias futuras. Por exemplo, o tipo *tgeompoint* se baseia no tipo *geometry* do PostGIS, que armazena geometrias 2D e 3D, e aproveita seus recursos de transformação do sistema de coordenadas. A implementação atual inclui 6 tipos temporais: ponto de geometria temporal (*tgeompoint*), ponto de geografia temporal (*tgeogpoint*), *tint, tfloat, tbool* e *ttext*. Os tipos temporais armazenam os valores dos tipos base em um conjunto de instantes no tempo ou de intervalos como mostrado nos exemplos abaixo:

a) tfloat ’17@2018-01-01 08:00:00, 17.5@2018-01-01 08:05:00, 18@2018-01-01 08:10:00’, guarda os valores de temperatura a cada instante, onde os valores entre estes instantes são desconhecidos.

b) tint ’[40@2018-01-01 08:00:00, 70@2018-01-01 08:05:00, 55@2018-01-01 08:10:00]’, guarda os valores de velocidade em km/h onde os valores entre estes instantes são interpolados.

c) tgeompoint ’[Point(1 1)@2001-01-01 08:00:00, Point(2 2)@2001-01-01 08:05:00, Point(3 3)@2001-01-01 08:10:00]’, guarda a posição do objeto, onde os valores das coordenadas entre estes instantes são interpolados.

O sistema de tipos é extensível para que mais tipos sejam implementados no futuro - conforme será feito com o MASTER. Esses tipos vêm com um conjunto de funções que podem ser usadas em consultas SQL. A implementação dessas funções usa a estrutura de operações, indexação e otimização existentes. O resultado direto dessa abordagem é que o MobilityDB se beneficia das melhorias feitas no PostgreSQL e PostGIS. Por exemplo, versões recentes do PostgreSQL vêm com novos recursos que permitem o processamento paralelo de consultas, que o MobilityDB incorpora de forma transparente. A arquitetura do MobilityDB pode ser vista na Figura 6.

De acordo com (ZIMÁNYI *et al.*, 2019), o MobilityDB implementa uma série de operações em tipos temporais. Essas funções e operadores são polimórficos, ou seja, seus argumentos podem ser de vários tipos, e o tipo de resultado pode depender dos tipos dos argumentos. Quando mais de um argumento temporal é aceito por uma operação, o resultado é definido apenas na interseção de seus intervalos de tempo. Se os intervalos de tempo forem disjuntos, o resultado será nulo. Há também operações para realizar comparações e aritmética sobre tipos temporais, para entrada

e saída de valores temporais, para calcular a distância e para acessar as propriedades dos tipos temporais. Finalmente, o MobilityDB define agregações temporais como *tmin(stream(tfloat))*, que retorna um *tfloat* que representa a cada instante o valor mínimo de toda a entrada. Da mesma forma, existem operações agregadas para máximo temporal, soma e média para inteiros temporais e pontos flutuantes, bem como centroide temporal para pontos de geometria temporal. Alguns exemplos de funções podem ser vistos na Tabela 1.

Ainda de acordo com (ZIMÁNYI *et al.*, 2019), o MobilityDB pode ser integrado a diversos ambientes, componentes e ferramentas que podem ser usados no gerenciamento e na visualização de dados tais como: a) QGIS e Grafana para análise gráfica dos dados de trajetórias de objetos móveis, b) linguagens de programação como: C, Java e Python, c) Citusdata [1], uma ferramenta que trata o particionamento horizontal dos dados, d) Docker e Kubernetes para o gerenciamento de contêiners e e) o mesmo já foi validado em ambientes de nuvem como AWS e Azure.

Tabela 1 – Exemplos de operações do MobilityDB

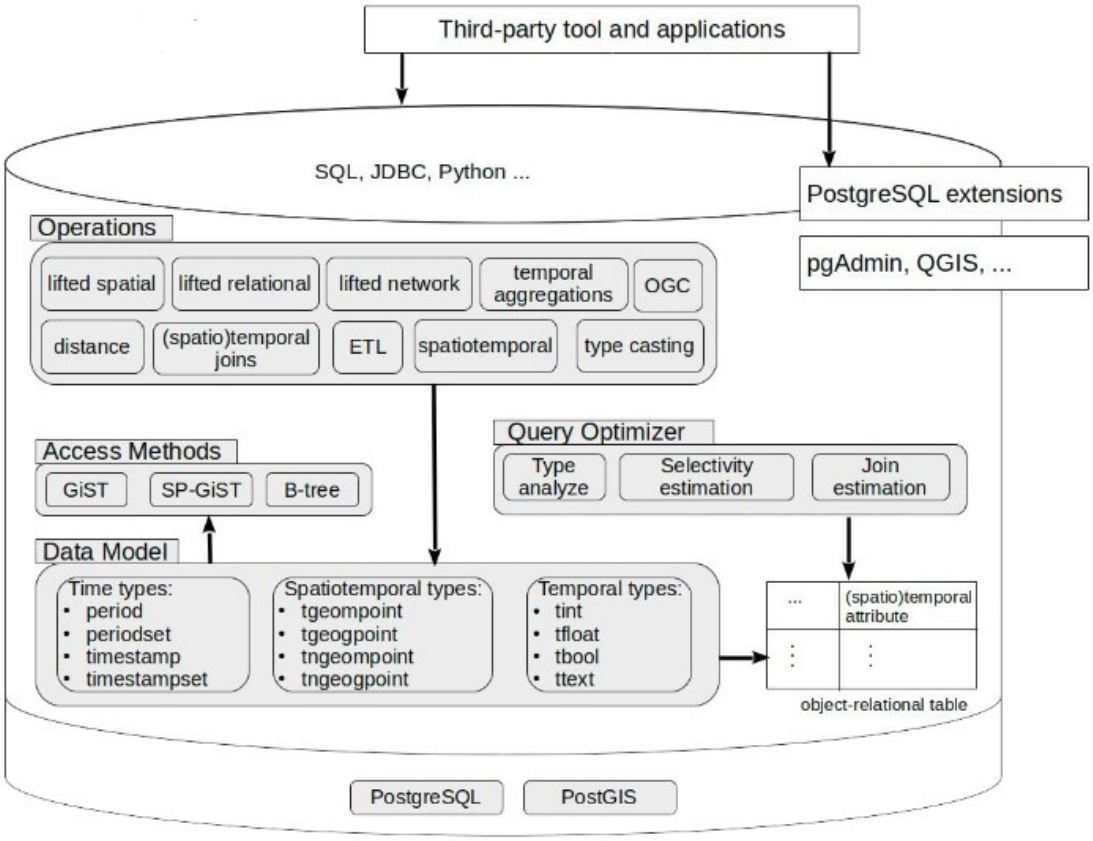
trajectory(tgeompoint) -> geometry(line)	min/max(tfloat) -> float
speed(tgeompoint) -> tfloat	atmin/atmax(tfloat) -> tfloat
cumulativeLength(tgeompoint) -> tfloat	length(tgeompoint) -> float

O MobilityDB também estende os índices GiST e SP-GiST fornecidos pelo PostgreSQL, para que possam funcionar sobre os tipos temporais. O índice GiST implementa uma árvore *R-tree* para tipos alfanuméricos temporais e para tipos de pontos temporais. O índice SP-GiST, diferente do anterior, é baseado em um princípio de particionamento de espaço que divide o espaço de dados em regiões não sobrepostas; cada nó da árvore corresponde a uma partição do espaço, o que pode levar a um desempenho de consulta mais eficiente quando as geometrias envolvidas na consulta não se sobrepõem. O tipo de armazenamento é sempre um *bounding box* onde a dimensionalidade depende do tipo: box 1D (ou seja, um período) para *tbool* e *ttext*, box 2D (valor e tempo) para *tint* e *tfloat*, box 3D ou 4D para *tgeogpoint* e *tgeompoint* dependendo se os pontos subjacentes são pontos 2D ou 3D.

Cabe ainda salientar que o MobilityDB se baseia nas estratégias de execução de consultas do PostgreSQL. O otimizador do PostgreSQL estima a seletividade de predicados para escolher os planos de execução mais eficientes para consultas, ou seja, otimização baseada em custo. Ele decide, por exemplo, se deve realizar varreduras de índice e em quais tabelas, se deve paralelizar a execução da consulta e a ordem de cálculo dos predicados. Para isso, ele coleta estatísticas sobre o conteúdo das tabelas usando o comando ANALYZE e as armazena na tabela de catálogo pg_statistic. Essas estatísticas incluem *slots* para valores predefinidos, como a porcentagem de nulos e o

[1] https://www.citusdata.com/

Figura 6 – Arquitetura do MobilityDB.



Fonte: Adaptado de (ZIMANYI *et al.*, 2020)

tamanho médio. Para tabelas grandes, é obtida uma amostra aleatória do conteúdo da tabela, em vez de examinar todas as linhas. Isso permite que grandes tabelas sejam analisadas em um curto período de tempo.

3 TRABALHOS RELACIONADOS

3.1 REVISÃO SISTEMÁTICA

O primeiro esforço para atender os objetivos deste trabalho foi a realização de uma pesquisa bibliográfica de modo a adquirir conhecimento sobre persistência de trajetórias para objetos móveis em BDs e implementação de modelos de representação de trajetórias semânticas. Para isto foi realizada uma revisão sistemática do estado da arte sobre tais temas. Buscas por trabalhos referentes ao tema persistência de trajetórias semânticas nos principais repositórios digitais de estudos da área da Ciência da Computação foram realizadas.

De acordo com (KITCHENHAM, 2004), o processo da revisão sistemática define um protocolo para pesquisas que inclui: (i) as questões de pesquisa, (ii) as *strings* de busca utilizadas nas principais bases de dados científicas e (iii) os critérios de inclusão e exclusão de trabalhos.

Com relação às questões de pesquisa, definiu-se uma questão principal e algumas questões secundárias, conforme segue:

Questão principal:

- RQ1 - Quais são as propostas para persistência de trajetórias semânticas?

Questões secundárias:

- RQ2 - Quais soluções de gerência de dados são utilizadas como base para a persistência de trajetórias?
- RQ3 - Extensões a essas soluções são desenvolvidas ou utilizadas?
- RQ4 - Quais modelos de dados são considerados para representar trajetórias semânticas?

As *strings* de busca aplicadas aos trabalhos científicos foram as seguintes:

Texto:
(“moving object“ AND database AND spatiotemporal) OR
(trajectory AND database AND spatiotemporal) OR
(“moving object“ AND graph) OR (multiple AND aspect AND trajectoy)

Titulo:
“moving object“ OR trajectory

Abstract:
database

Strings de busca

Tabela 2 – Critérios de Inclusão e exclusão

Inclusão	**Exclusão**
Artigos escritos na língua inglesa	Trabalhos que não apresentam uma proposta de implementação
Proposta de modelagem inclui a implementação em um BD	Trabalhos sem avaliação de resultados
Trabalhos escritos a partir de 2010	Trabalhos que tratem apenas de trajetórias brutas
Persistência de trajetórias semanticamente enriquecidas	

Tabela 3 – Relação dos repositórios de trabalhos científicos pesquisados

Repositório	**Resultados**	**Selecionados**
IEEE	169	1
Wiley	55	2
ACM	144	5
Scopus	284	4
Springer	183	3
DBLP	293	1
Total	**1128**	**16**

A Tabela 2 define os critérios de inclusão e de exclusão e a Tabela 3 apresenta as bases de dados pesquisadas e os resultados obtidos.

Inicialmente foram selecionados 1128 trabalhos. Após a exclusão dos trabalhos duplicados com o auxílio da ferramenta *Zotero* [1], este número diminuiu para 391. A seguir foram aplicados os critérios de inclusão e exclusão no título e no *abstract*, reduzindo o número para 146. Por fim, aplicando os mesmos critérios no texto completo, o número final de trabalhos selecionados ficou em 16. Esses trabalhos são descritos a seguir.

3.2 ESTRATÉGIAS DE PERSISTÊNCIA DE TRAJETÓRIAS SEMÂNTICAS

Esta seção visita diversos MODs, iniciando pela tecnologia relacional onde são apresentados os MODs mais populares: (*Hermes* e *Secondo*), seguido por outras propostas baseadas em diferentes tecnologias de BD como , grafo, triplestore e multimodelo. O MobilityDB também é um MOD relacional, para persistência de trajetórias brutas, tendo sido explicado na seção 2.7. Na tabela 4 é apresentado um resumo dos MODs estudados. Por fim é realizada a comparação dos trabalhos analisados com o MASTERMobilityDB.

[1] *www.zotero.org*

3.2.1 Bancos de Dados Objeto-Relacionais

Hermes (PELEKIS; THEODORIDIS, 2014) é uma iniciativa pioneira da comunidade científica para promover o armazenamento eficiente de trajetórias brutas e semânticas em BDs. Da sua especificação foram derivadas 3 implementações:

(a) Hermes@Oracle: implementação realizada através de um data cartridge do BD Oracle utilizando as primitivas espaciais disponíveis do Oracle Spatial;

(b) Hermes@Postgres: biblioteca escrita na linguagem C da qual são importadas as funcionalidades como uma extensão do BD PostgreSQL juntamente com o PostGIS;

(c) Hermes@Neo4j: plugin do BD NoSQL Neo4j Spatial.

Na proposta de armazenamento do Hermes, 2 repositórios distintos são previstos: um BD de trajetórias brutas e um BD de trajetórias semânticas. Também é definido um conjunto de tipos de dados baseado em um modelo conceitual que representa trajetórias brutas e semânticas. Adotando a definição de um tipo de dados de trajetória bruta T de um objeto em movimento como uma tripla *(o-id, traj-id, geom)*, onde o-id é o identificador de um objeto em movimento, traj-id é a trajetória deste objeto e geom é uma polilinha tridimensional (x, y, t) que representa seu movimento. Por sua vez, o conceito de subtrajetória bruta T' de T pode ser representada pelo mesmo tipo de dado e é definida de forma semelhante como uma tripla *(o-id, traj-id, subtraj-id, geom)*, onde subtraj-id é o identificador da subtrajetória específica do objeto em movimento, e geom é a porção de T entre dois carimbos de data/hora, ti e tj, $ti \leq tj$. Da mesma forma, o suporte a trajetórias semânticas pode ser alcançado através de 2 novos tipos de dados:

(a) *episode* de uma trajetória semântica corresponde a T', definido como uma tupla *(defineTag, MBB, episodeTag, activityTag, T-link)*, onde defineTag é um flag que define *episode* como Parada ou Movimento; MBB é uma tupla (MBR, t_{inicio}, t_{fim}) correspondendo à aproximação tridimensional de T', onde MBR é a geometria envolvente da projeção espacial de T', e t_{inicio}, t_{fim} a projeção temporal de T'. EpisodeTag e activityTag contêm informações semânticas textuais sobre paradas ou movimentos, e T-link é um link para a representação de T;

(b) *semantic trajectory* T_{sem} de um objeto em movimento corresponde a uma T, definido como uma tripla *(o-id, semtraj-id, T_{sem})*, onde o-id é o identificador de um objeto em movimento; semtraj-id é a trajetória semântica de um objeto em movimento; e T_{sem} é uma sequência de episódios, $\{e_1, ..., e_n\}$ pertencente a T e ordenado no tempo, ou seja, $e_i[t_{fim}] \leq e_{i+1}[t_{inicio}]$, $1 \leq i < n$.

De modo a oferecer suporte a essas funcionalidades por meio de uma linguagem de consulta, é definido um conjunto de métodos e de operadores para os tipos de

dados do Hermes, que são aplicados nos atributos das tabelas contendo as trajetórias.

Sobre indexação, o Hermes adapta uma estrutura para dados de mobilidade, o índice *TB-tree (Trajectory Bundle Tree)*, usado em trajetórias brutas, acompanhado de um índice textual, como uma lista invertida. Este método de acesso híbrido para trajetórias semânticas é denominado *Semantic Trajectory Bundle Tree (STB-Tree)*.

Outro esforço pioneiro na implementação da persistência de trajetórias de objetos móveis é o BD Secondo (GÜTING; BEHR; DÜNTGEN, 2010). Ele se propõe a ser uma infraestrutura que pode ser estendida para fins de pesquisa em MODs utilizando outros parâmetros e *datasets*. A infraesturutura consiste de: *(i)* um BD de objetos móveis baseado na biblioteca BerkeleyDB para BDs embutidos que trata a persistência e as transações no BD; *(ii)* um módulo gerenciador; e *(iii)* várias extensões do Secondo, chamadas *módulos de álgebra*, que torna a arquitetura aberta a novas implementações.

O módulo gerenciador do Secondo possui 3 componentes principais: *kernel*, *otimizador* e *interface com o usuário*. O primeiro trata da execução dos comandos, acesso as álgebras e também do armazenamento dos dados. O otimizador utiliza o modelo objeto-relacional e suporta um dialeto SQL, mapeando os comandos para o pseudo-código nativo do Secondo. A interface com o usuário é um *frontend* Java utilizada para o envio de comandos e visualização espacial do resultado das consultas. Está disponível também a ferramenta *BerlinMOD* para geração de *datasets*.

Outro trabalho dos mesmos autores apresenta uma extensão, ou álgebra, do BD Secondo, para tratamento de trajetórias simbólicas (GÜTING; VALDÉS; DAMIANI, 2015). Uma trajetória simbólica é uma sequência de pares (t, l), onde t é um intervalo de tempo e l é um rótulo descrevendo certos aspectos de uma trajetória. A informação simbólica é computada a partir do próprio movimento ou obtida do ambiente geográfico. Exemplos incluem nomes de estradas e meios de transporte. O objetivo é fornecer um modelo simples para qualquer tipo de informação semântica sem definir as localizações geométricas. Se considerarmos meios de transporte, uma trajetória simbólica para um indivíduo O2 é denotada por:

Symbolic(O2) = < ([*t* 1, *t* 2], *A_pe*), ([*t* 2, *t* 3], *Onibus*), ([*t* 3, *t* 4], *Metro*), ([*t* 4, *t* 5], *A_pe*), ([*t* 5, *t* 6], *Em_casa*) >

Weka-STPM (Semantic Trajectory Preprocessing Module) (BOGORNY *et al.*, 2011) é o primeiro sistema de enriquecimento semântico e de mineração de dados de trajetórias implementado na ferramenta *Weka* (FRANK *et al.*, 2010). Ele abrange as etapas de pré-processamento de dados, mineração de dados e visualização de dados e padrões. Dois métodos são disponibilizados para enriquecer trajetórias: o *IB-SMoT*, que considera a presença ou a ausência do objeto em movimento em locais relevantes, e o *CB-SMoT* para enriquecer semanticamente trajetórias usando velocidade média máxima e limite de velocidade para identificar regiões de interesse de baixa veloci-

dade. Com o Weka-STPM o usuário é capaz de adicionar informações semânticas às trajetórias, em uma etapa de pré-processamento, e então explorar diversos algoritmos disponíveis no Weka para mineração de dados. Como os dados de trajetória semântica são armazenados em um BD, isso permite que o usuário explore os dados com consultas espaciais e não espaciais. O Weka-STPM é testado em um esquema objeto-relacional (OR) no BD PostGIS.

Já o trabalho de Brandoli implementa uma aplicação para dados espaço-temporais que descrevem as atividades de pesca no Mar Adriático ao longo de 4 anos (BRANDOLI *et al.*, 2022). Um BD é construído baseado na fusão de 2 fontes de dados complementares: trajetórias de embarcações de pesca (obtidas de um *Sistema de Identificação Automática terrestre - AIS*) e relatórios de captura de peixes (a quantidade e o tipo de peixe capturado). São apresentadas as fases da criação do BD, começando pelos dados brutos e passando pela exploração dos dados, limpeza dos dados, reconstrução da trajetória e enriquecimento semântico através do uso do modelo MASTER (SANTOS MELLO *et al.*, 2019). A implementação do BD utiliza o *MobilityDB*.

Posteriormente, são realizadas várias análises com o objetivo de mapear as atividades pesqueiras de algumas espécies-chave e inferir novos conhecimentos úteis para o manejo pesqueiro. Além disso, é investigado o uso de métodos de aprendizado de máquina para prever a *captura por unidade de esforço (CPUE)*, um indicador da exploração de recursos pesqueiros para orientar o desenho de políticas específicas. Uma variedade de métodos de previsão, tomando como entrada os dados do BD e fatores ambientais, como temperatura do mar, altura das ondas e *Clorophilla*, são utilizados para avaliar sua capacidade de previsão.

No trabalho desenvolvido por (XU; LU, H.; BAO, 2023) os autores propõem a persistência de trajetórias multi-atributo a qual consiste em uma trajetória espaço-temporal e um conjunto de atributos descritivos. Tais atributos enriquecem a representação das trajetórias espaço-temporais tradicionais para se ter um conhecimento abrangente dos objetos em movimento. Em um MOD, grande parte das consultas contém dois predicados, espaço-temporal e atributo, e retorna os objetos cujas localizações estão dentro de um limite de distância para a trajetória da consulta e os atributos contêm valores esperados. Existem diferentes planos de execução para responder à consulta. Com base neste cenário o trabalho foca na melhoria da capacidade de processamento de um MOD onde os autores propõem um otimizador que é essencialmente necessário para: (i) estimar com precisão o custo de estratégias de consulta alternativas em termos de acessos ao disco, (ii) construir um módulo de tomada de decisão que classifique automaticamente os dados de maneira apropriada e seleciona o plano de consulta ideal, e (iii) atualizar os modelos analíticos quando novas trajetórias são alcançadas. Tal otimizador baseado em custo suporta distribuição de dados espaço-temporais uniformes e não uniformes e incorpora distribuição de atributos. A persistência das trajetórias multi-

atributo e o otimizador são totalmente desenvolvidos dentro do BD Secondo e avaliado de forma abrangente em termos de precisão e eficácia usando grandes conjuntos de dados reais e sintéticos.

DeepVQL (LEW; YOO; NAM, 2023) é um trabalho que aborda a extração de trajetórias com base em vídeos extraídos de câmeras, com o uso de técnicas de aprendizagem profunda. Recentemente, surgiram novos sistemas que suportam linguagens de consulta declarativas semelhantes a SQL, com foco no desenvolvimento de seus próprios sistemas para suportar novas consultas combinadas com aprendizagem profunda que não são suportadas nativamente. O sistema DeepVQL proposto foi desenvolvido estendendo-se o BD PostgreSQL, onde são propostas várias novas funções definidas pelo usuário (UDFs) para oferecer suporte à funcionalidade de BD de vídeo no PostgreSQL, ao mesmo tempo que oferece suporte à detecção de objetos, rastreamento de objetos e consultas de análise de vídeo. A vantagem deste sistema é a sua capacidade de utilizar consultas com regiões espaciais específicas ou durações temporais como condições para analisar objetos em movimento. Uma das aplicações mais úteis de linguagens de consulta declarativas, como DeepVQL, é um sistema de monitoramento de vídeo de tráfego. Para as validações os autores publicaram uma interface[2] para o sistema DeepVQL onde se pode consultar às trajetórias e vídeos e acessar os experimentos realizados.

3.2.2 Bancos de Dados NoSQL de Grafo

Um trabalho nesta categoria propõe um modelo de dados baseado em um grafo de propriedades para representar trajetórias semânticas (GÓMEZ *et al.*, 2019). O modelo é baseado em técnicas de modelagem de grafos OLAP para favorecer a agregação de dados de trajetórias representados como grafos. O modelo é aplicado ao conjunto de dados *Foursquare New York* enriquecido com informações contextuais. Para fins de persistência, considera-se o BD Neo4j Spatial, bem como a biblioteca APOC para manipulação da dimensão temporal. Para efeito de comparação, o mesmo dataset foi carregado em uma tabela no BD relacional PostgresSQL. Uma coleção de 12 consultas analíticas foi definida e submetida a ambos os BDs. Considerando o tempo de execução das consultas à tabela no BD PostgreSQL versus os grafos no BD Neo4j os resultados demonstraram uma melhor viabilidade do Neo4j.

Um outro trabalho modela espaços internos e externos para apoiar uma representação homogênea e contínua da mobilidade humana nestes espaços (NOUREDDINE *et al.*, 2021). Ele apresenta uma esquema espacial hierárquico que engloba espaços internos e externos através da captura de sensores (RFID ou GPS, respectivamente), um modelo de trajetória semântica flexível que integra várias fontes de dados em espaços internos e externos onde as trajetórias são geradas. A proposta é aplicada

[2] http://deepvql.urbanai.net/

a dados reais do projeto *Polluscope*[3], onde trajetórias humanas com múltiplos dados contextuais são coletadas em um contexto de detecção de multidões. Tais trajetórias são caracterizadas por dados externos e geo-localizados usando *OpenStreetMap* [4]. As trajetórias em espaços internos são geradas aleatoriamente através do simulador *VITA* (LI *et al.*, 2016). Ambas as trajetórias são armazenadas no BD Neo4j Spatial utilizando a bibilioteca APOC. Por fim, vários exemplos de processamento de dados envolvendo consultas como: vizinhos mais próximos, baseadas em local, entre outros, são avaliados e discutidos demonstrando a viabilidade da solução.

O BD *GSM (Geo-Social-Moving)*, cuja implementação estende o BD orientado a grafos Neo4j para permitir a gestão unificada de trajetórias, é uma outra proposta relacionada (ZHANG; LU, F.; XU, 2016). O foco do trabalho são relações sociais para objetos em movimento. Uma grande quantidade de tipos de dados definidos pelo usuário e operadores correspondentes são propostos para facilitar consultas geo-sociais em objetos em movimento. O GSM utiliza 3 grafos: *(i)* o *grafo geográfico*, que modela o espaço geográfico para consultas utilizando a subdivisão do espaço em regiões de Voronoi (WIKIPÉDIA, 2022) construídas através do agrupamento de pontos das trajetórias que se encontram mais próximas de um POI do que de qualquer outro, gerando uma coleção de nós e arcos onde os nós representam as regiões de Voronoi e os arcos representam as adjacências; *(ii)* o *grafo social*, que modela as relações entre os objetos móveis sendo que os nós do grafo representam os objetos e possuem atributos como nome ou tipo, e os arcos correspondem às relações sociais; e *(iii)* o *grafo de movimento*, que modela as trajetórias dos objetos móveis, onde cada trajetória é dividida em segmentos de trajetória usando um diagrama de Voronoi. Cada segmento é representado por um único nó do grafo, as arestas associam esses nós sequencialmente e a trajetória completa é modelada como um conjunto de nós. Comparado com soluções baseadas em sistemas tradicionais de gerenciamento de BD relacional estendido, caracterizados por operações de junção de tabelas demoradas, o modelo GSM proposto caracterizado pela travessia de grafos é considerado mais poderoso na representação de objetos em movimento em massa com relacionamentos sociais, e mais eficiente e estável para consultas de relações sociais geo-localizadas.

3.2.3 Bancos de Dados Triplestore

Uma ontologia denominada STEP (Semantic Trajectories Episodes) é proposta para enriquecimento semântico de trajetórias tratadas como uma série de episódios (NOGUEIRA, T. P.; MARTIN, 2015). Nesta ontologia há elementos para a anotação semântica da trajetória e dos episódios incluindo atributos quantitativos, qualitativos, espaciais e temporais. Entre as vantagens de escolher uma abordagem de Web Se-

[3] *http://polluscope.uvsq.fr*
[4] *https://openstreetmap.org*

mântica com ontologias para modelagem e implementação da solução, os autores destacam a possibilidade de integrar diferentes fontes de dados através de consultas federadas, suporte à raciocínio de máquina por inferência e a integração de dados e metadados. *FrameSTEP* (NOGUEIRA, T. *et al.*, 2018) implementa a ontologia STEP para a persistência de trajetórias utilizando o *BD Virtuoso*[5], um triplestore que aceita consultas escritas na linguagem SPARQL, e para consultas espaciais, podem ser utilizadas as funções internas do Virtuoso ou GeoSPARQL da OGC. A expressividade do STEP é demonstrada na comparação com os modelos de enriquecimento semântico *Baquara2* (FILETO *et al.*, 2015) e *CONSTAnT* (BOGORNY *et al.*, 2014).

O framework *STriDE* (KHALID; CRUZ; GINHAC, 2018) considera trajetórias semanticamente enriquecidas para tratar segurança na construção civil. A construção de trajetórias semânticas é base para reconhecer movimentos inseguros dos trabalhadores que podem levar a acidentes. As trajetórias capturadas através de sensores como câmeras inteligentes, wearables e GPSs são anotadas semanticamente. Define-se uma hierarquia de conceitos descritos como triplas RDF. O STriDE, que é baseado em ontologias, tem a capacidade de armazenar dados de objetos dinâmicos para rastrear mudanças na forma, tamanho e atributos destas entidades (p.ex., um usuário, uma trajetória e uma sala), cada qual com sua fatia de tempo e conceitos. O STriDE usa o conceito de fatias de tempo para acompanhar a evolução das entidades, persistindo os objetos processados no triplestore *Stardog*[6], acessível via SPARQL e GeoSPARQL. Com a aplicação do modelo *HMM* (LIU *et al.*, 2015) sobre as trajetórias semanticamente enriquecidas, os movimentos categorizados do usuário são visualizados usando a abordagem *Building Information Modeling (BIM)* (BULBUL; TAYLOR; OLGUN, 2014). BIM é um modelo interativo de construção inteligente que contém a geometria da construção e informações em tempo real relacionadas aos locais da construção mais suscetíveis a movimentos inseguros dos usuários. As visualizações em BIM podem ser usadas pelos níveis de gerência na elaboração de estratégias de segurança aprimoradas.

O trabalho de Torres apresenta um modelo de *stops-and-moves* (ALVARES *et al.*, 2007) anotados semanticamente com dados adicionais (TORRES *et al.*, 2020). Uma linguagem de consulta para trajetórias que inclui seletores para paradas ou movimentos com base em suas anotações semânticas, e expressões de sequência que definem como combinar os resultados dos seletores com a sequência de episódios que a trajetória semântica define, é proposta. Uma ontologia é proposta para enriquecer semanticamente trajetórias segmentadas e apresenta expressões sequenciais de stops-and-moves. Em seguida, descreve um modelo concreto de trajetória semântica em RDF, e a estratégia de persistência utilizando o triplestore *Apache Jena*. O trabalho

[5] *www.virtuoso.com*
[6] *www.stardog.com*

também define expressões de busca ao longo de trajetórias semânticas com base no uso de palavras-chave para especificar consultas stop-and-move e na adoção de termos com semântica predefinida para compor expressões de sequência. Em seguida, mostra como compilar essas expressões em consultas SPARQL. Finalmente, ele apresenta um experimento de prova de conceito sobre um conjunto de dados de trajetórias semânticas construído com conteúdo gerado por usuários do *Flickr* combinado com dados da *Wikipedia*.

3.2.4 Propostas Multimodelo

Um trabalho propõe um modelo de dados de grafos que enriquece a geometria simplificada de trajetórias com a semântica perdida no processo de simplificação, como velocidade, rumo, tempo, distância percorrida latitude e longitude (TAMILMANI *et al.*, 2019). Seguindo uma abordagem multimodelo, trajetórias brutas, inicialmente modeladas e armazenadas em um BD PostGIS, são simplificadas de acordo com suas características espaciais e temporais usando *Distância Euclidiana Sincronizada (SED)* (MERATNIA; BY, 2004), enquanto a estrutura de dados *Semanically Enriched Line simpliFication (SELF)* (TAMILMANI; STEFANAKIS, 2017) é adotada para preservar a semântica de pontos eliminados no processo de simplificação.

Na sequência, trajetórias simplificadas enriquecidas são modeladas no BD Neo4j Spatial em termos de nós e arestas. A simplificação baseada em SED foi realizada em um conjunto de dados de trajetórias de navios em agosto de 2013 no Mar Egeu. Diversas classes de consultas são executadas provando que, com o enriquecimento semântico do SELF para realizar análises de trajetórias, não é necessário lidar com todos os seus pontos.

Um estudo de caso de mecanismo de inferência em trajetórias semânticas é apresentado no trabalho de Wannous (WANNOUS *et al.*, 2013). Ele propõe uma solução baseada em ontologia para modelagem de trajetórias semânticas integrando dados e regras de tempo para trajetórias de focas e suas atividades como se alimentar, viajar e descansar, com o objetivo de identificar suas regiões principais de alimentação. Os dados de entrada são fornecidos pelo laboratório *LIENSS*[7], onde são capturadas as trajetórias destes animais. As regras ontológicas que mapeiam este cenário são codificadas em RDF e as trajetórias semanticamente enriquecidas são persistidas em um BD Oracle 11g que oferece suporte para o armazenamento de triplas RDF e suporta consultas híbridas em SQL e SPARQL. Entende-se que se trata de uma abordagem multi-modelo pois integra duas estratégias de persistência: triplas RDF embutidas em tabelas objeto-relacionais. O trabalho apresenta experimentos evidenciando o desempenho e a precisão do modelo de persistência na identificação das regiões de interesse com base nas trajetórias.

[7] *http://lienss.univ-larocheller.fr*

Tabela 4 – Comparação de soluções para persistência de trajetórias semânticas

BD MOD	BD Base	Extensão	Tecnologia
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; LU, H.; BAO, 2023)	Secondo	Symbolic	OR
DeepVQL	PostgreSQL	PostGIS	OR
MasterMobilityDB	MobilityDB	-	OR
(GÓMEZ *et al.*, 2019)	Neo4j	Neo4j Spatial/APOC	Grafo
(NOUREDDINE *et al.*, 2021)	Neo4j	Neo4j Spatial/APOC	Grafo
GSM	Neo4j	Neo4j Spatial	Grafo
Hermes@Neo4j	Neo4j	Neo4j Spatial	Grafo
FrameSTEP	Virtuoso	-	Triplestore
STriDE	Stardog	-	Triplestore
(TORRES *et al.*, 2020)	Apache Jena	-	Triplestore
(TAMILMANI *et al.*, 2019)	PostgreSQL Neo4j	PostGIS	Multimodelo
(WANNOUS *et al.*, 2013)	Oracle	Spatial	Multimodelo

3.3 COMPARATIVO

A Tabela 4 apresenta um comparativo das propostas de persistência de trajetórias semânticas encontradas pelo processo de revisão sistemática. Esta tabela é complementada pela figura 7 onde percebe-se uma tendência de uso dos modelos de dados OR e grafo, provavelmente pela flexibilidade que eles oferecem para conectar os diversos pontos que definem uma trajetória e suas propriedades espaço-temporais e semânticas. No caso do modelo OR, trajetórias semânticas são persistidas em relações onde os pontos da trajetória são armazenados em colunas de geometria e as anotações semânticas em colunas alfanuméricas. Soluções triplestore também se destacam pela habilidade de representar dados pontuais conectados através de triplas RDF que respeitam uma ontologia com propriedades espaço-temporais ou semânticas. Por último, tem-se a proposta multimodelo que trabalha com versões detalhadas e simplificadas de trajetórias nos modelos de dados relacional e de grafo, respectivamente.

Na figura 8 tem-se a evolução dos trabalhos desde a última década, com uma média de 1,5 trabalhos por ano, sendo que em 2023 foram publicados 3 novos trabalhos. Lembrando que foram contabilizados somente trabalhos que tratam com trajetórias semânticas.

Na implementação de um MOD baseado em um BD OR, uma trajetória é representada como um tipo de dados abstrato (ADT), encapsulando características espaço-

Figura 7 – Quantidade de trabalhos por tecnologia

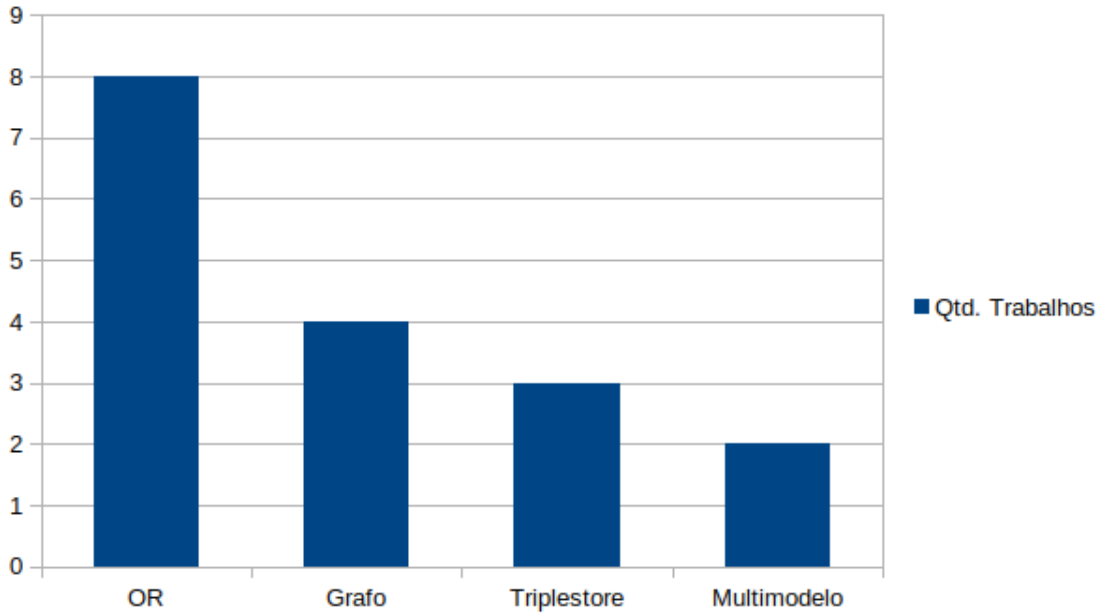
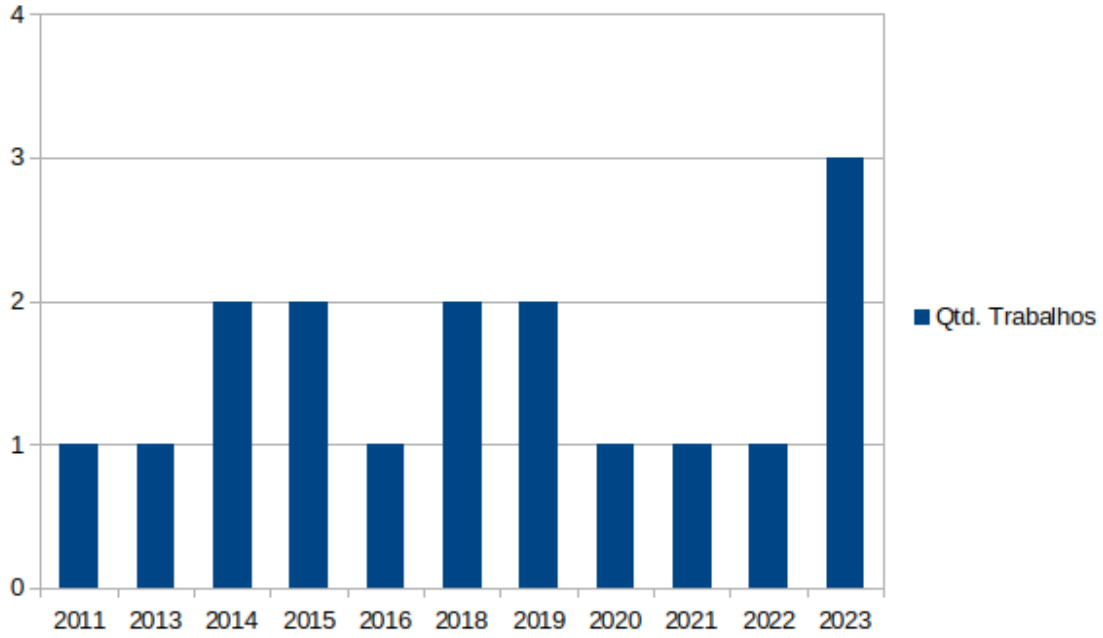


Figura 8 – Quantidade de trabalhos por ano



temporais e semânticas, bem como operações sobre suas propriedades. Junto com a estrutura de dados proposta, também deve haver uma linguagem de manipulação composta por operações e funções baseadas em conjuntos em ADT, como *Union* e *Intersect*. Além disso, são definidos índices para os ADTs para aumentar o desempenho das consultas. A persistência é alcançada por meio de relacionamentos com atributos que herdam de ADTs e são tratados por meio de SQL. Entre os MODs analisados, Hermes@Oracle, Hermes@Postgres, Secondo e o trabalho de (BRANDOLI *et al.*, 2022)

seguem esta estratégia, enquanto Weka-STPM a implementa em parte porque utiliza a ferramenta de mineração de dados Weka em vez de definir ADTs. PostGIS DB é adotado pela maioria das soluções.

As implementações de MODs baseados em triplestore apresentam abordagens de base ontológica para modelagem de trajetórias semânticas considerando 3 componentes principais: *ontologia de domínio, ontologia de tempo* e *ontologia espacial*, que representam conceitos básicos de domínio contextualizando movimento, espaço e tempo , bem como as relações entre eles. Junto com as ontologias, regras são definidas. Alguns deles são declarativos (por exemplo, viajar é uma atividade) e outros são calculados a partir de fórmulas nos dados de entrada. Em seguida, é feita uma integração semântica entre as ontologias por meio de consultas para entender as relações temporais e espaciais, como, por exemplo, uma atividade ocorrida em uma determinada área e durante um intervalo de tempo específico. As propostas enfatizam o uso de SPARQL e GeoSPARQL, porém são bastante heterogêneas em termos de BD base para persistência de dados.

A proposta multimodelo encapsula a geometria e a semântica dos dados de mobilidade em diferentes modelos de dados. O modelo relacional mantém os dados brutos das trajetórias GPS, que são previamente limpos para remover dados imprecisos e *outliers*, e posteriormente persistidos como um fluxo de tuplas espaço-temporais. O modelo grafo, por outro lado, associa diversas propriedades semânticas, como velocidade, aceleração, ângulo de movimento, densidade e intervalo de tempo, a partes de uma trajetória bruta.

Por último, soluções baseadas em grafos de propriedades modelam vértices e arestas que podem ser anotadas com propriedades. Eles são usados para representar trajetórias brutas e semânticas. Neste modelo, por exemplo, as coordenadas de pontos ou episódios podem ser representadas como vértices, e há uma aresta de um ponto ao próximo na sequência de pontos da trajetória. Além disso, podem ser incluídas coordenadas espaço-temporais como propriedades, bem como outras características dos locais visitados. Também podem ser definidos dados contextuais hierárquicos, o que permite que o gráfico de trajetória seja representado em diferentes granularidades, sendo muito útil em análises estatísticas. A representação de trajetórias por meio de grafos permite que elas sejam armazenadas em sua forma nativa em um BD como o Neo4j, que é um produto maduro com diversas bibliotecas de funções, como Neo4j Spatial e APOC, que é adotado por todas as soluções.

Trajetórias de múltiplos aspectos representam uma nova visão de trajetórias e um novo paradigma para dados de mobilidade. Esses aspectos não são apenas simples rótulos semânticos, mas também podem ser objetos complexos e/ou informações heterogêneas intrinsecamente associadas aos traços físicos dos objetos em movimento. A proposta do *MASTERMobilityDB* se diferencia do estado da arte apre-

sentado neste estudo por ser a primeira proposta para persistência e manipulação e acesso a trajetórias de múltiplos aspectos em um MOD de propósito geral.

4 PROPOSTA

Conforme mencionado nos capítulos anteriores, os MODs fornecem conceitos em seu modelo, nas suas estruturas de dados e na sua implementação para representar objetos móveis, ou seja, geometrias em constante mudança de tempo e local, bem como histórias completas de representações de movimentos. O objetivo no projeto de funções e operações de consulta para objetos móveis é poder fazer perguntas sobre tais movimentos, realizar análises e derivar informações, de uma maneira mais simples e elegante possível. O sistema subjacente deve suportar a execução eficiente de tais análises.

A Figura 9 apresenta uma visão geral da solução proposta, onde podem ser vistos os componentes de gerenciamento de dados da mesma. Cada componente da solução trata uma parte dos dados de trajetórias conforme os tipos de dados que ele oferece e se comunica com os demais através de APIs disponíveis para diversas linguagens de programação como C e PL/SQL. Detalhando-se a arquitetura, inicialmente tem-se o BD PostgreSQL[1] que trata os dados alfanuméricos, as relações, a integridade referencial e as transações. O próximo componente é o PostGIS[2] um sistema bastante conhecido para tratamento de geometrias e operações espaciais. Na sequência tem-se o MobilityDB que trata as trajetórias brutas e as suas operações. No último nível, tem-se o *MASTERMobilityDB* (camada de gestão de MATs proposta neste trabalho) cujos tipos abstratos de dados baseados no modelo físico do MASTER representam as MATs.

A solução MASTERMobilityDB contempla a persistência das trajetórias de múltiplos aspectos que pode ser separada em dois componentes:

a) Espaço-temporal, implementada pelo MobilityDB, contendo as geometrias das trajetórias e dos objetos móveis.

b) Alfanumérico, que modela de maneira flexível os múltiplos aspectos heterogêneos com base no modelo físico do MASTER.

Seguem-se a este núcleo, a persistência para as duas extensões já mencionadas:

a) MAT-SG, para a persistência das trajetórias representativas.

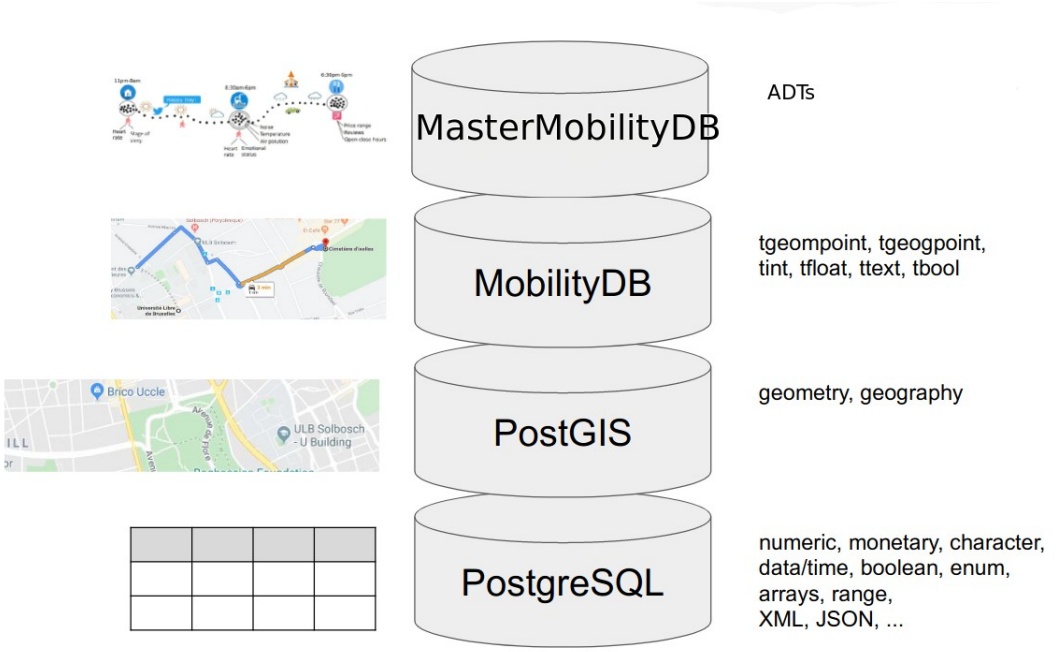
b) MASTER-DR, para persistência das regras de dependência.

Está fora do escopo da proposta o cálculo das trajetórias representativas bem como das regras de dependência, ficando estas implementações para trabalhos futuros.

Algo que foi considerado ao longo da implementação foi o desempenho frente ao grande volume de dados que tende a aumentar mais e mais ao longo do tempo, já que

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

Figura 9 – Visão geral da solução proposta



Fonte: Adaptado de (ZIMANYI *et al.*, 2020)

as coletas são feitas de dispositivos integrados com GPS, geram uma grande massa de dados. Neste caso foram utilizadas diversas alternativas para diminuir o tempo de carga das MATs e de execução das consultas como o processamento em lote dos dados de entrada, a utilização de índices espaço-temporais especializados, divisão de consultas complexas por meio de tabelas temporárias, a utilização de tabelas externas e o particionamento dos objetos do BD.

Sendo assim, considerando as MATs, o MASTERMobilityDB deve estar apto a persistir o objeto móvel, suas trajetórias e os dados complexos referentes a seus aspectos, regras de dependência e trajetórias representativas, fornecendo um ambiente adequado para manipular e visualizar as MATs dos objetos em movimento.

O código fonte e as instruções para instalação do MASTERMobilityDB estão disponíveis no GITHUB através do link *https://github.com/ffeller/MasterMobilityDB*.

4.1 REQUISITOS FUNCIONAIS DA CAMADA PROPOSTA

Os requisitos funcionais definem as características e capacidades que a camada de persistência proposta deve cumprir para processar, gerenciar e analisar dados de trajetórias semanticamente enriquecidas de acordo com o modelo de representação MASTER. Estes requisitos são baseados nos princípios gerais de um MOD (AGGARWAL, 2013) como: (i) gerenciamento de dados espaço-temporais, (ii) modelagem de dados, (iii) indexação e consulta, (iv) busca e análise, (v) integração com outros sistemas, (vi) desempenho, (vii) interface com o usuário e (viii) interoperabi-

lidade Uma das atividades mais importantes deste projeto é a definição da plataforma de BD a ser estendida para o armazenamento das MATs de acordo com o modelo de dados MASTER. Na revisão bibliográfica apresentada no Capítulo 3 foram mostrados vários sistemas de BDs de diversas tecnologias que serviram como base para persistência de trajetórias. Deve-se levar em consideração a maior complexidade do MASTER na escolha do BD alvo, por isso são apresentados a seguir os requisitos funcionais da camada proposta:

a) R1: deve permitir o armazenamento da geometria da trajetória.
b) R2: deve possuir primitivas espaciais que permitam a manipulação da geometria da trajetória.
c) R3: deve prover índices espaço-temporais para a recuperação das trajetórias.
d) R4: deve suportar o desenvolvimento de tipos abstratos de dados.
e) R5: deve permitir o armazenamento e a recuperação dos dados alfanuméricos das trajetórias e dos objetos em movimento
f) R6: deve suportar o armazenamento de trajetórias espaço-temporais brutas.
g) R7: deve estar disponível em uma versão estável, o que exclui os protótipos.
h) R8: deve ser de código aberto e não licenciado.
i) R9: deve ser flexível para suportar uma quantidade variável de atributos de diferentes tipos de dados.

4.2 JUSTIFICATIVA PARA A ESCOLHA DO SISTEMA DE BANCO DE DADOS BASE

O modelo de dados MASTER é o primeiro modelo de dados na literatura voltado à representação de trajetórias enriquecidas com ilimitados aspectos semânticos. Assim sendo, a ideia inicial para a camada de persistência proposta é a extensão de um sistema de BD de trajetórias existente.

Conforme explicado no capítulo 3, nos últimos anos a comunidade científica vem investindo em esforços para a persistência de dados de trajetórias de onde surgiram diversas propostas de BDs de trajetórias. Com base na revisão sistemática realizada foram definidos 3 candidatos: Hermes, Secondo e MobilityDB. Na Tabela 5 pode ser visualizado como cada MOD suporta os requisitos funcionais que foram definidos para o MASTERMobilityDB.

Estes 3 sistemas são boas alternativas, mas o MobilityDB se mostrou o mais adequado para a implementação do MASTER dado que possui uma implementação bastante madura e que pode ser integrado com outras ferramentas e ambientes conforme mencionado no capítulo 2. O Hermes é um projeto que não sofre atualização desde 2017 e o seu website encontra-se indisponível. Por fim, o Secondo é um produto

Tabela 5 – Características dos trabalhos relacionados

BD MOD	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**
Hermes@Oracle	X	X	X	X	X	X	X	-	X
Hermes@Postgres	X	X	X	X	X	X	X	X	X
Secondo	X	X	X	X	X	X	X	X	-
Weka-STPM	X	X	-	-	X	-	X	-	
(BRANDOLI *et al.*, 2022)	X	X	X	X	X	X	-	X	-
(XU; LU, H.; BAO, 2023)	X	X	-	X	X	X	-	X	X
DeepVQL	X	X	-	X	X	X	-	X	-
MasterMobilityDB	X	X	X	X	X	X	X	X	X
(GÓMEZ *et al.*, 2019)	X	X	-	X	X	X	-	X	X
(NOUREDDINE *et al.*, 2021)	X	X	-	X	X	X	-	X	X
GSM	X	X	-	X	X	X	-	X	-
Hermes@Neo4j	X	X	X	X	X	X	X	X	X
FrameSTEP	X	X	-	X	X	X	-	-	X
STriDE	X	X	-	X	X	X	-	-	-
(TORRES *et al.*, 2020)	X	X	-	X	X	X	-	X	X
(TAMILMANI *et al.*, 2019)	X	X	-	X	X	X	-	X	X
(WANNOUS *et al.*, 2013)	X	X	-	X	X	X	-	X	-

de código aberto mas de difícil integração com outras ferramentas e o mesmo carece de interfaces de acesso como ODBC, OLEDB e JDBC. O conceito de trajetórias simbólicas no modelo de dados do Secondo é limitado a um atributo e de um aspecto, basicamente uma etiqueta ou *label*, o que é insuficiente para suportar o MASTER.

4.3 MODELO LÓGICO DO MASTERMOBILITYDB

O modelo lógico do *MASTERMobilityDB* tem o objetivo de representar logicamente as trajetórias e as suas relações com os objetos móveis, os aspectos e seus atributos, abstraindo as especificidades do BD MobilityDB. Tal modelo deve considerar que o BD proposto seja flexível o suficiente para suportar qualquer tipo de aplicação que precise acessar e gerenciar MATs. Desta forma, o modelo deve ser aberto para que a configuração dos aspectos possa variar de uma aplicação para a outra e também dentro da mesma aplicação, uma vez que as configurações de aspectos podem variar de acordo como o contexto do movimento. Por exemplo, para um individuo relaxando em casa, podem ser anotados os aspectos: frequência cardíaca e estágio do sono; trabalhando no escritório: pressão sanguínea e estado emocional. Para um indivíduo jantando em um restaurante, pode-se considerar a satisfação com o atendimento, e assim por diante. O mesmo processo pode ser realizado para trajetórias, pontos da trajetórias, relacionamentos entre objetos móveis e regras de dependência, conforme descrito na Seção 2.4.

Considerando este cenário, optou-se por modelar logicamente o MASTER utilizando a abordagem relacional, uma vez que o MobilityDB deriva deste modelo de BD e que o MASTER é um modelo de representação de trajetórias flexível, ou seja, para acrescentar um novo aspecto não é necessário alterar a estrutura das relações definidas no modelo de dados e da mesma forma não é necessário informar valores nulos para os aspectos e atributos, definidos de acordo com a aplicação, quando estes não são pertinentes a uma trajetória.

Aplicando-se regras de transformação sobre o modelo conceitual do MASTER - escrito na forma de diagrama de classes da UML, obteve-se o modelo lógico relacional mostrado na Figura 10, extraído de (SANTOS MELLO *et al.*, 2021). Nele estão presentes as entidades do núcleo do MASTER e também as extensões MAT-SG e MASTER-DR.

4.4 MODELO FÍSICO DO MASTERMOBILITYDB

O modelo físico do MASTERMobilityDB foi desenvolvido pela transformação do modelo lógico em um esquema de BD objeto-relacional chamado *MASTER* onde o BD subjacente é o MobilityDB. Tal esquema é criado no momento da instalação da extensão MASTERMobilityDB compreendendo o núcleo do MASTER e as extensões MAT-SG e MASTER-DR.

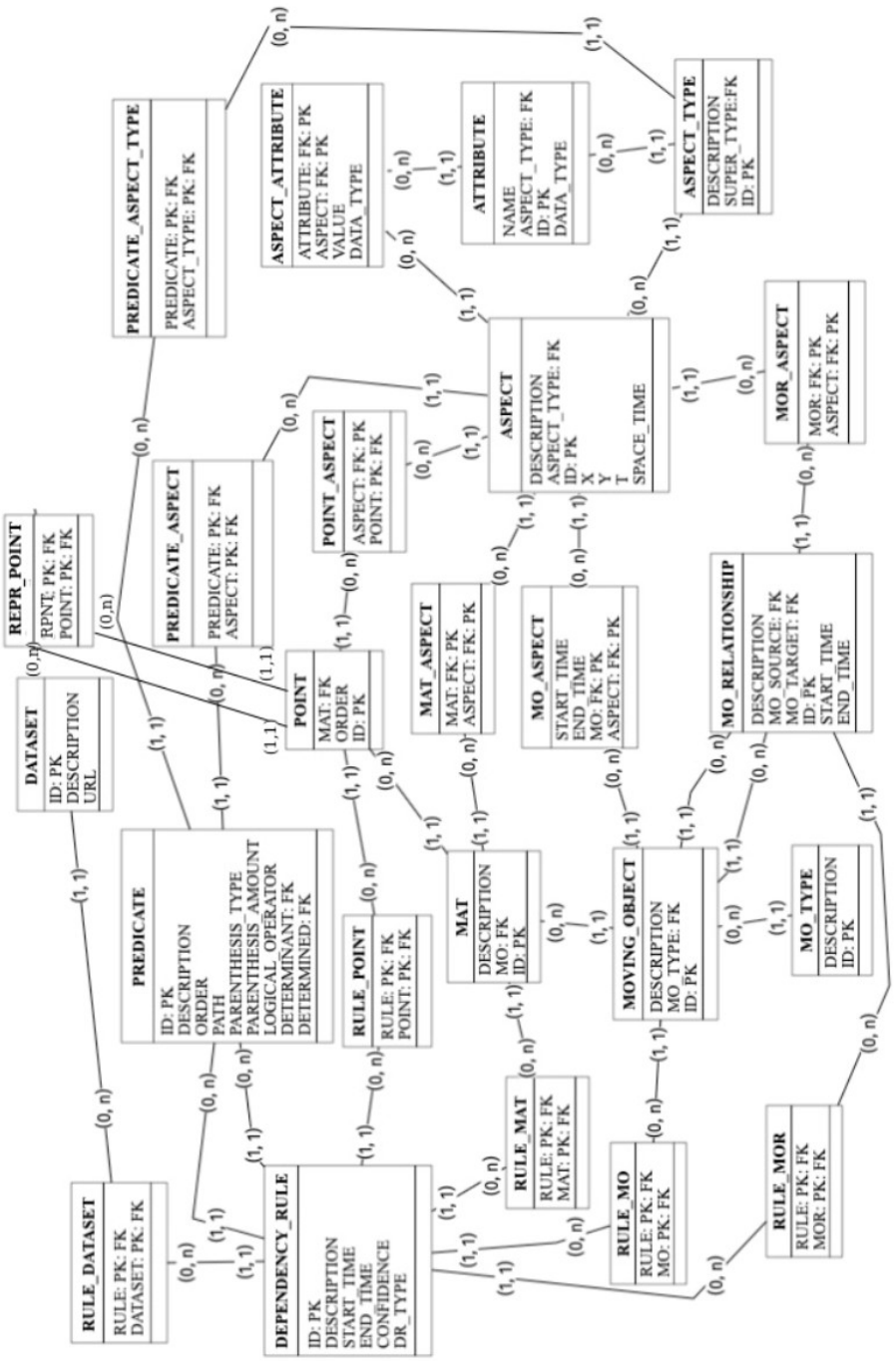
No modelo físico mostrado na Figura 11 tem-se inicialmente a relação ASPECT, que representa um fato ou objeto do mundo real relevante para uma ou mais trajetórias representadas na relação MAT cuja relação NxM é representada pela tabela MAT_ASPECT. O aspecto é caracterizado por um tipo de aspecto (ASPECT_TYPE) que também pode ser um subtipo de um tipo de aspecto mais geral, permitindo uma classificação de subtipo de tipo de aspecto, exemplo: POI - alojamento - hotel. O aspecto também apresenta os atributos X, Y e T opcionais, utilizados por objetos ou fatos que possuem uma referência espaço temporal.

Aspectos possuem atributos (ATTRIBUTE) os quais modelam as características do mesmo e possuem valores distintos para cada aspecto (ASPECT_ATTRIBUTE). Um exemplo de aspecto pode ser o Hotel Faial em Florianópolis, do tipo Hotel, localizado nas coordenadas (-27.59383, 48.55551) e o mesmo possui os atributos: estrelas(4), quantidade de quartos(115), estacionamento('S'), Bar('S') e Restaurante('S').

A relação MAT, que referencia um objeto em movimento (MOVING_OBJECT), guarda no seu atributo RAW_TRAJECTORY a trajetória bruta associada à trajetória de múltiplos aspectos, e na relação POINT, somente os pontos da trajetória que possuem anotações (aspectos associados), já que todos pontos coletados para a MAT, semanticamente anotados ou não, estão salvos no atributo RAW_TRAJECTORY.

Pontos possuem aspectos (POINT_ASPECT), por exemplo, os locais de interesse na cidade aonde um turista esteve durante um dia de uma viagem. Por fim,

Figura 10 – Modelo lógico do MASTERMobilityDB



temos a relação MOVING_OBJECT que representa o objeto em movimento - MO, e também pode ter aspectos (MO_ASPECT), e pode se relacionar com outros objetos em movimento (MO_RELATIONSHIP) de alguma forma.

Na relação MAT tem-se o atributo MAT_DATE, ou data de início da trajetória assim como na relação POINT tem-se o atributo T. Tais atributos podem ser usados para se fazer o particionamento destas duas relações, subdividindo-as em objetos menores conforme é explicado na seção 4.9.1.

O atributo VALUE na relação ASPECT_ATTRIBUTE, do tipo text, guarda o valor do atributo referente ao aspecto ASPECT e no atributo DATA_TYPE_ID é informado o tipo de dados. Como os valores dos atributos podem assumir diversas representações como texto, numérico, geometria (WKT), imagem (base64), etc, optou-se por utilizar o tipo de dados text visto que este permite o armazenamento de valores de até 1GB em formatos variados. Estes dois atributos cumprem um papel importante na indexação dos valores dos atributos visto que podem ser usados na criação de índices parciais, conforme é descrito na seção 4.9.4.

Na Figura 12, tem-se o modelo físico da extensão MAT-SG que modela as trajetórias representativas, as quais possuem um comportamento similar a uma MAT, porém os seus pontos representativos definidos em REPR_POINT são calculados com base na dispersão de pontos (POINT_REPR_POINT) das MATs que a mesma representa.

Por último, tem se o modelo físico da extensão MASTER-DR representado na Figura 13 o qual apresenta inicialmente as regras de dependência na relação DEPENDENCY_RULE, juntamente com os seus predicados (PREDICATE) e o DATASET associado. Além disso a mesma possui especializações para regras de dependência de: *(i)* trajetória (MAT_DR); *(ii)* relacionamentos entre objetos em movimento (MOR_DR); *(iii)* pontos (POINT_DR) e; *(iv)* objetos em movimento (MO_DR). As especializações por sua vez, associam zero ou mais entidades do núcleo do MASTER como trajetórias (MAT_MAT_DR) e pontos (POINT_POINT_DR). Por exemplo, para um dataset *Diaries*, SE velocidade média > 80 AND (dia da semana = ’Sábado’ OR dia da semana = ’Domingo’) AND (período = ’0-6’ OR período = ’18-24’) => objetivo = ’Lazer’, com grau de confiança de 72%. Nesse caso os predicados determinantes encontram-se à esquerda do operador "=>"e o predicado determinado à direita.

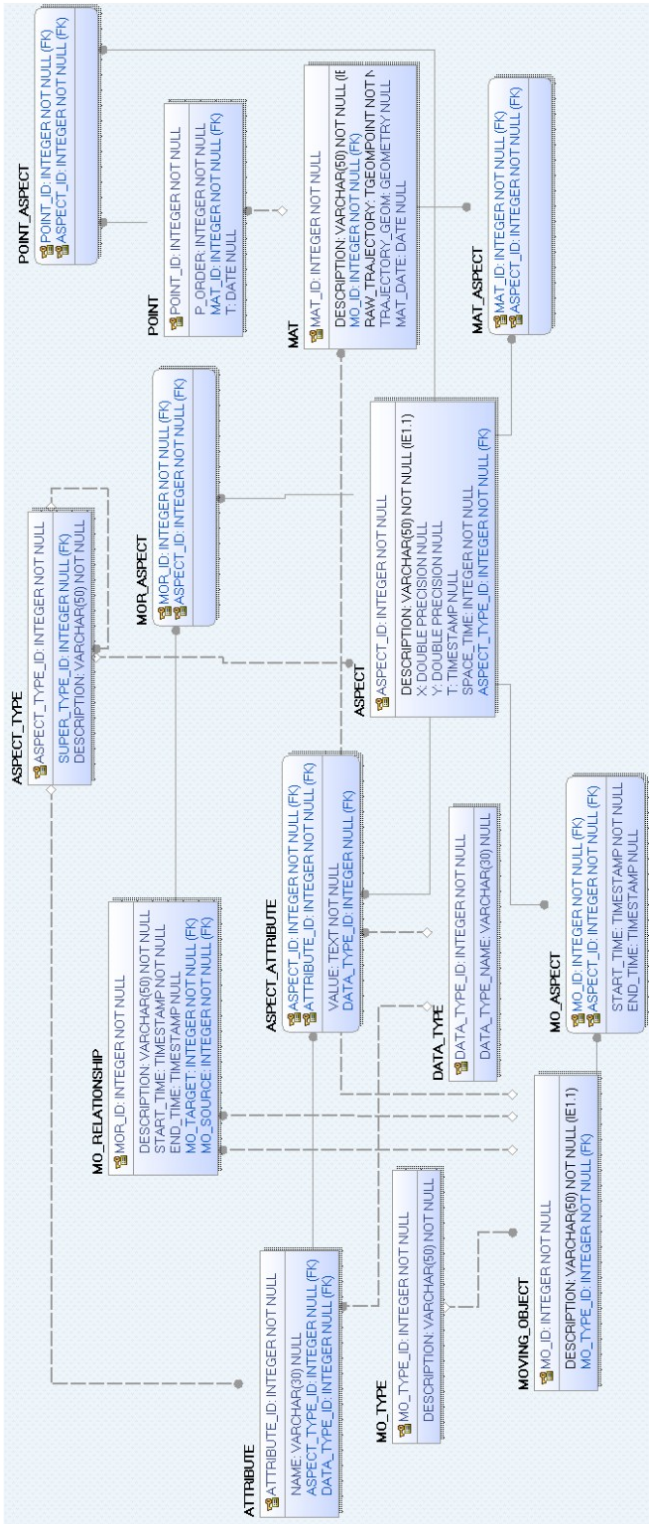
4.5 ORGANIZAÇÃO DO MASTERMOBILITYDB

Um esquema de BD no MASTERMoblityDB é uma coleção nomeada de objetos de BD, incluindo tabelas, visualizações, índices, sequências, tipos de dados e funções. Os esquemas permitem organizar objetos de BD em grupos lógicos, fornecendo uma maneira de separar diferentes partes de um BD e evitar conflitos de nomenclatura entre objetos.

Aqui estão alguns pontos-chave sobre esquemas em um BD no MASTERMoblityDB:

a) Namespace: os esquemas fornecem um espaço para nomes ou *namespace* para objetos de BD. Os objetos dentro de um esquema podem ser referenciados usando o nome do esquema como prefixo, o que ajuda a evitar conflitos de nomenclatura;

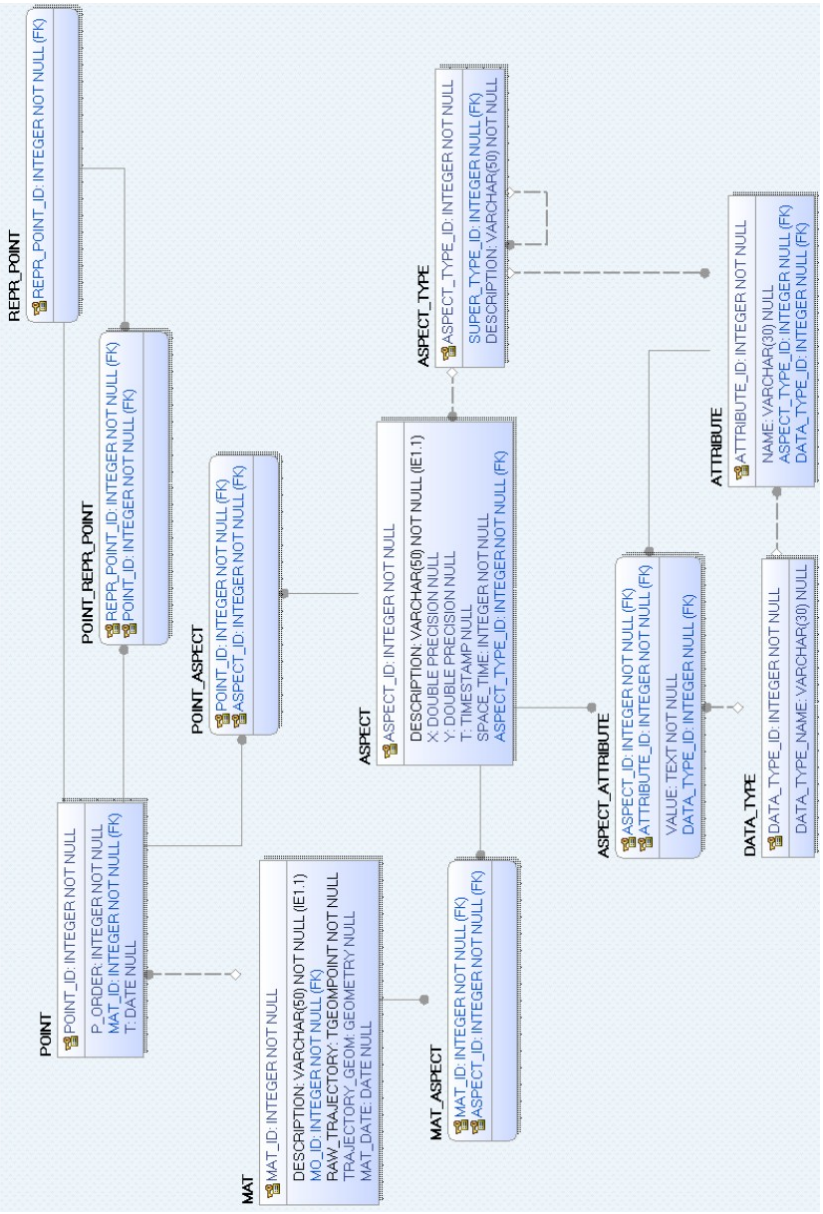
Figura 11 – Modelo físico do MASTERMobilityDB



b) Organização: Os esquemas ajudam a organizar os objetos do BD com base em seus relacionamentos lógicos;

c) Controle de acesso: Esquemas podem ser usados para controle de acesso. Você pode conceder ou revogar permissões no nível do esquema, permitindo controlar quais usuários ou funções têm acesso a partes específicas do BD;

Figura 12 – Modelo físico do MAT-SG



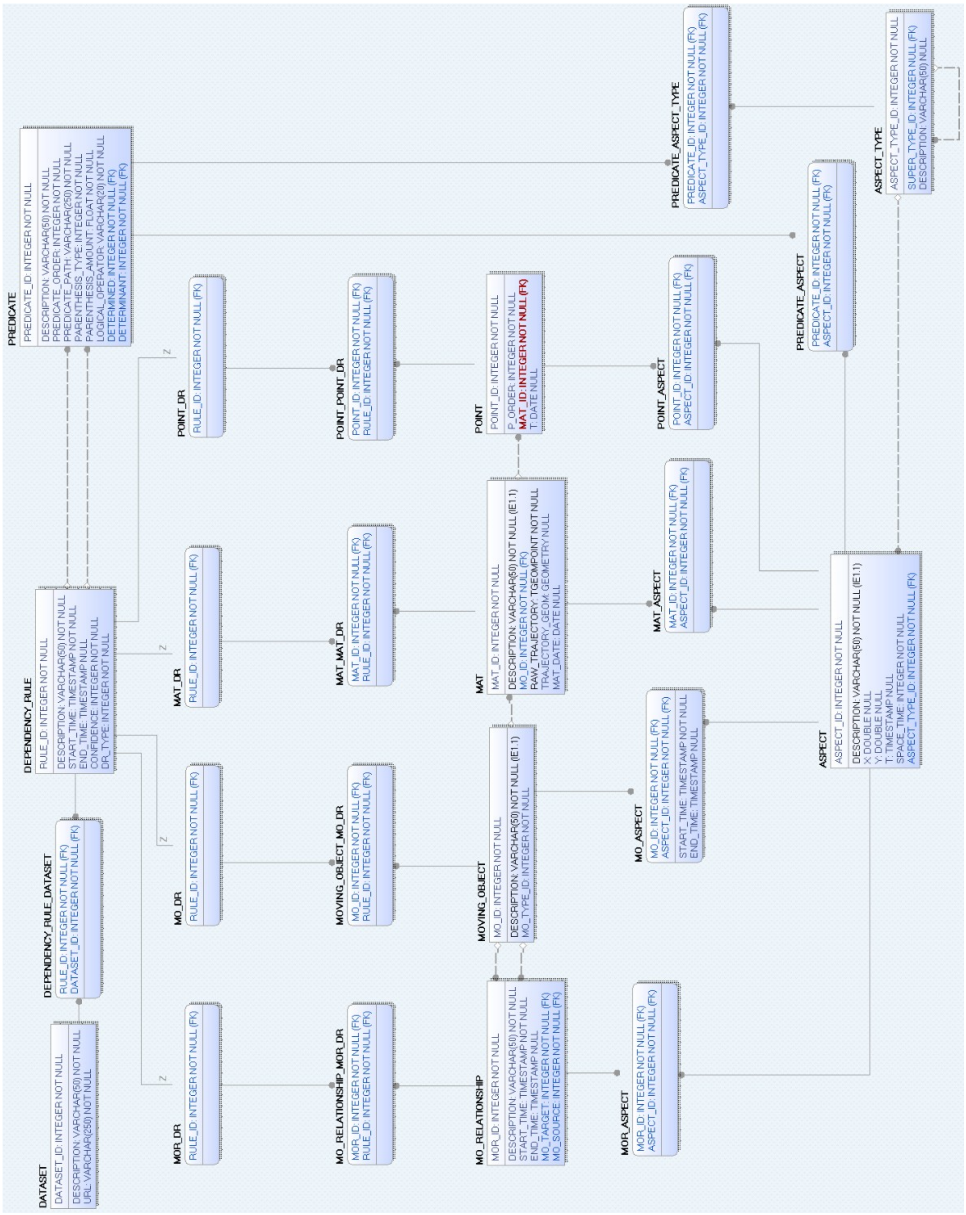
d) Caminho de pesquisa: o MobilityDB usa um caminho de pesquisa ou *search path* para determinar a ordem em que os esquemas são pesquisados ao procurar um objeto específico. O caminho de pesquisa é uma lista de nomes de esquemas e o BD procura objetos na ordem especificada pelo caminho de pesquisa;

e) Esquema padrão: quando um usuário se conecta ao BD, ele possui um esquema padrão. Se eles fizerem referência a um objeto sem especificar um esquema, o objeto será pesquisado no esquema padrão do usuário.

Para referenciar um objeto em um esquema, deve ser usada a sintaxe: <esquema>.<objeto>.

O MASTERMoblityDB implementa diversos objetos de modo a suportar efici-

Figura 13 – Modelo físico do MASTER-DR



entemente a persistência das MATs como tabelas, procedimentos, sequências, entre outros. Com o objetivo de organizar e gerenciar tais objetos foram definidos esquemas de modo a separar os objetos de acordo com a sua finalidade. Assim pode-se melhorar a clareza e controlar o acesso a diferentes partes do BD. A seguir temos a relação destes esquemas:

a) master: mantém os objetos(relações, funções, sequências, etc) do modelo MASTER;

b) master_dr: mantém os objetos(relações, funções, sequências, etc) da extensão MASTER-DR;

c) mat_sg: mantém os objetos(relações, funções, sequências, etc) da extensão MAT-SG;

d) partitions: No esquema partitions são mantidas as partições que segmentam as relações em objetos menores.

e) util: As rotinas utilitárias utilizadas principalmente nos procedimentos de extração, limpeza e carga (ETL) dos dados estão disponíveis no esquema util;

f) staging: o esquema staging não possui objetos e deve ser utilizado para registrar os procedimentos de ETL desenvolvidos pelo usuário de acordo com a aplicação.

4.6 TIPOS ABSTRATOS DE DADOS

Tipos de dados compostos referem-se a tipos abstratos que podem combinar vários atributos ou campos em uma única estrutura nomeada. Esses tipos compostos às vezes também são chamados de "tipos estruturados". Eles são úteis quando se deseja agrupar dados relacionados. Os tipos compostos fornecem uma maneira de organizar e estruturar dados de maneira significativa, facilitando o trabalho com conjuntos de informações relacionados.

Uma vez definido um tipo composto, ele pode ser usado como tipo de coluna em uma tabela ou como tipo de variável ou argumento em uma função. Tipos compostos também podem ser usados em funções para retornar vários valores. Por fim, um tipo composto pode ser incorporado em outro tipo composto e também podem ser definidos *arrays* de tipos compostos.

O MobilityDB herda do PostgreSQL uma vantagem interessante de trabalhar com tipos de dados compostos ou abstratos devido à sua forma simples e eficiente de converter valores para um determinado tipo de dados em uma consulta SQL. Um exemplo de tipo de dados composto é o tipo *aspect_type*, que possui seis atributos: aspect_id um número inteiro que identifica o aspecto, um caractere variável (description), dois valores flutuantes (x e y representando as coordenadas), um carimbo de data / hora (t), o flag space_time (inteiro) e aspect_type_id um número inteiro que é uma referência para uma chave na relação aspect_type correspondente.

Para implementar o MASTERMobilityDB na forma de uma API para tratamento de MATs, foi explorado um conjunto de tipos de dados alfanuméricos do BD PostgreSQL juntamente com os tipos de dados espaciais estáticos oferecidos pela opção Espacial Postgis e o tipo *tgeompoint* introduzido no MobilityDB para persistência de trajetórias brutas. Com base nesses tipos de dados de objetos espaço-temporais e no modelo físico do MASTER, o MASTERMobilityDB define uma série de tipos de dados, detalhados na Tabela 6, para suportar MATs e os mesmos são utilizados nas operações e nas funções definidas dentro da API. Trata-se de um conjunto de tipos compostos que espelha a estrutura do modelo físico do MASTER e das suas extensões.

Tabela 6 – Tipos abstratos de dados do MASTERMobilityDB

Extensão	Tipo de dados	Atributos
master	aspect_attribute_typ	aspect_id int, attribute_id int, value text, data_type_id int
master	aspect_typ	aspect_id int, description varchar(50), x float, y float, t timestamp, space_time int, aspect_type_id int
master	aspect_type_typ	aspect_type_id int, super_type_id int, description varchar(50)
master	attribute_typ	attribute_id int, name varchar(30), aspect_type_id int, data_type_id int
master	data_type_typ	data_type_id int, data_type_name varchar(30)
master-dr	dataset_typ	dataset_id int, description varchar(50), url varchar(250)
master-dr	dependency_rule_dataset_typ	rule_id int, dataset_id int
master-dr	dependency_rule_typ	rule_id int, description varchar(50), start_time timestamp, end_time timestamp, confidence int, dr_type int
master	mat_aspect_typ	mat_id int, aspect_id int
master-dr	mat_dr_typ	rule_id int
master-dr	mat_mat_dr_typ	mat_id int, rule_id int
master	mat_typ	mat_id int, description varchar(50), mo_id int, raw_trajectory tgeompoint, trajectory_geom geometry, mat_date date, aspect_a aspect_typ[]
master	mo_aspect_typ	mo_id int, aspect_id int, start_time timestamp, end_time timestamp
master-dr	mo_dr_typ	rule_id int
master-dr	mo_relationship_mor_dr_typ	mor_id int, rule_id int
master	mo_relationship_typ	mor_id int, description varchar(50), start_time timestamp, end_time timestamp, mo_target int, mo_source int, aspect_a aspect_typ[]
master	mo_type_typ	mo_type_id int, description varchar(50)
master	mor_aspect_typ	mor_id int, aspect_id int
master-dr	mor_dr_typ	rule_id int
master-dr	moving_object_mo_dr_typ	mo_id int, rule_id int
master	moving_object_typ	mo_id int, description varchar(50), mo_type_id int, aspect_a aspect_typ[]
master	point_aspect_typ	point_id int, aspect_id int
master-dr	point_dr_typ	rule_id int
master-dr	point_point_dr_typ	point_id int, rule_id int
mat-sg	point_repr_point_typ	point_id int, repr_point_id int
master	point_typ	point_id int, p_order int, mat_id int, t date, aspect_a aspect_typ[]
master-dr	predicate_aspect_typ	predicate_id int, aspect_id int
master	predicate_aspect_type_typ	predicate_id int, aspect_type_id int
master	predicate_typ	predicate_id int, description varchar(50), predicate_order int, predicate_path varchar(250), parenthesis_type int, parenthesis_amount float, logical_operator varchar(20), determinant int, determined int, aspect_a aspect_typ[]
mat-sg	repr_point_typ	repr_point_id int

Com base no modelo físico do MASTER foram desenvolvidos os tipos abstratos de dados e as operações, disponibilizados em uma API escrita em PL/SQL que pode ser consumida por aplicações que necessitem tratar com MATs.

4.7 OPERAÇÕES E FUNÇÕES

Um cenário de utilização simples para um usuário do MASTERMobilityDB é projetar e construir MATs em um esquema de BD objeto-relacional e construir um aplicativo por meio de transações com esse BD. Neste caso, onde o BD subjacente é o MobilityDB, para especificar o esquema do BD, a extensão MASTERMobilityDB manipula *scripts* de definição de dados (DDL) com o objetivo de estender a linguagem SQL com suporte para operações em trajetórias mencionadas anteriormente. Para cons-

Tabela 7 – Operações de criação e manipulação de dados no MASTERMobilityDB

Operação	Descrição
count()	retorna a contagem de tuplas na relação
create()	cria uma nova tupla na relação
create_many()	cria uma ou mais tuplas na relação
delete()	elimina uma tupla da relação
delete_all()	elimina todas as tuplas da relação
delete_by_id()	elimina uma tupla da relação com base na chave primária
delete_by_name()	elimina uma tuplas da relação com base no atributo nome ou descrição
find_all()	retorna todas as tuplas da relação
find_by_id()	retorna uma tupla da relação com base na chave primária
find_by_name()	retorna uma tupla da relação com base no atributo nome ou descrição
find_by_attribute()	retorna as tuplas da relação que possuem associados aspectos do tipo informado e cujo valor do atributo atende aos parâmetros informados
update()	atualiza uma tupla da relação

truir um aplicativo sobre esse BD que trate MATs, aspectos, objetos em movimento, consultar dados e manipular informações, o desenvolvedor do aplicativo escreve um programa fonte em alguma linguagem de programação como Java ou Python, onde ele pode incorporar scripts PL/SQL que invocam construtores de objetos e operações do MASTERMobilityDB. O poder da linguagem de programação com a funcionalidade de BD oferecida pelo PL/SQL estendido geram o executável da aplicação. Escrever procedimentos armazenados independentes que aproveitam a funcionalidade do MASTERMobilityDB e compilá-los com o compilador PL/SQL é outra maneira de construir um aplicativo orientado a trajetórias de múltiplos aspectos.

Com o objetivo de simplificar o acesso e facilitar a manipulação das MATs, foram desenvolvidas operações e funções dentro da extensão MASTERMobilityDB. Estas rotinas foram implementadas dentro dos esquemas master e util como procedimentos e funções escritas na linguagem procedural PL/SQL do MobilityDB, criados na instalação da extensão no BD. Algo que foi considerado na implementação, visto que o volume de dados tende a aumentar, foi a manipulação de várias tuplas em uma operação, como em *create_many* que insere várias tuplas na relação em uma chamada. A Tabela 7 apresenta as operações que tratam a persistência no modelo MASTER. As operações ali apresentadas são instanciadas para cada uma das entidades do MASTER como em moving_object_create_many ou aspect_find_by_id. Na Tabela 8 são apresentadas as operações implementadas para a relação *aspect*, onde tem-se o nome da operação, o tipo de retorno e a assinatura. Por fim no Apêndice A tem se a relação das operações e funções desenvolvidas dentro desta camada de persistência.

Algumas observações: *(i)* a notação [] indica que trata-se de um array de objetos; *(ii)* a notação SETOF indica que se trata de um conjunto de objetos do tipo especificado e; *(iiii)* c) os tipos moving_object_typ, mat_typ, point_typ e mo_relationship_typ possuem na sua estrutura um array de objetos aspect_a que guarda os aspectos associados.

Tabela 8 – Exemplo de operações para a relação *aspect*

Operação	Retorno	Parâmetros
aspect_count()	integer	
aspect_create()		INOUT p_aspect aspect_typ
aspect_create_many()		INOUT p_aspect aspect_typ[]
aspect_delete_many()		IN p_aspect_a aspect_typ[]
aspect_delete_all()		
aspect_delete_by_id()		IN p_aspect_id integer
aspect_delete_by_name()		IN p_description varchar
aspect_find_all()	SETOF aspect_typ	
aspect_find_by_id()	SETOF aspect_typ	IN p_aspect_id integer
aspect_find_by_name()	SETOF aspect_typ	IN p_description varchar
aspect_find_by_atribute()	SETOF aspect_typ	IN p_aspect_typ varchar, IN p_attribute_name, IN p_value
aspect_update()		INOUT p_aspect_a aspect_typ[]

4.8 ROTINAS UTILITÁRIAS

Com o objetivo de apoiar o desenvolvimento dos procedimentos para extração, limpeza e carga (ETL) dos dados no MASTERMobilityDB estão disponíveis algumas rotinas utilitárias dentro do esquema util do BD. Tais rotinas são instanciadas em tempo de instalação da extensão. A Tabela 9 mostra a relação das rotinas utilitárias desenvolvidas.

Em uma possível extensão do MASTERMobilityDB podem vir a ser desenvolvidas novas rotinas utilitárias para tarefas mais especializadas como a construção de conjuntos de dados (*DataFrames*) de MATs que poderão ser a entrada para a aplicação de técnicas de *machine learning* ou *datamining* sobre os mesmos.

4.9 ALTERNATIVAS PARA MELHORIA DE DESEMPENHO DO BD

Conforme dito anteriormente, com a explosão da Internet das Coisas e a enxurrada de big data gerados na Internet, agora é possível coletar enormes volumes de dados de movimento sobre pessoas, animais, objetos (automóveis, ônibus, drones), entre outros. Entretanto, a integração de dados de movimento e dados alfanuméricos e a transformação dessa massa de dados em informação útil pode ser uma tarefa difícil e dispendiosa caso não sejam tomados os devidos cuidados na implementação dos processos de extração, limpeza e carga. Nesse sentido o MobilityDB oferece funcionalidades que são muito úteis na carga e construção das trajetórias bem como nas consultas sobre as mesmas, tais como:

a) Particionamento de objetos: refere-se à divisão do que é logicamente uma grande tabela em partes físicas menores e mais gerenciáveis.

b) Tabelas temporárias: são eliminadas automaticamente no final de uma sessão ou, opcionalmente, no final da transação atual. Elas desempenham um papel importante na divisão de consultas complexas em consultas mais simples e de resolução mais rápida pelo BD.

Tabela 9 – Rotinas utilitárias para procedimentos de ETL

Operação	Parâmetros	Descrição
create_partitions_by_date	IN p_schemaname text, IN p_tablename text, IN p_startdate date, IN p_enddate date, IN p_columnname text, IN p_interval text, IN p_schemapart text	Cria as partições para a relação especificada por p_schemaname e p_tablename dentro do período definido por p_startdate e p_enddate utilizando a coluna p_columnname para distribuir os valores de acordo com o intervalo que pode ser 'day', 'week', 'month', etc. As partições são alocadas no esquema definido em p_schemapart cujo valor default é 'partitions'.
drop_partitions_by_date	IN p_schemaname text, IN p_tablename text, IN p_startdate date, IN p_enddate date, IN p_interval text, IN p_schemapart text	Elimina as partições da relação especificada por p_schemaname e p_tablename dentro do período definido por p_startdate e p_enddate de acordo com o intervalo que pode ser 'day', 'week', 'month', etc, alocadas no esquema definido em p_schemapart cujo valor default é 'partitions'.
disable_fks	IN p_schemaname text, IN p_tablename text	Desabilita as chaves estrangeiras da relação especificada por p_schemaname e p_tablename, com o objetivo de diminuir o tempo de carga dos registros.
enable_fks	IN p_schemaname text, IN p_tablename text	Habilita as chaves estrangeiras da relação especificada por p_schemaname e p_tablename, validando as mesmas contra a respectiva relação ancestral. Tal procedimento é mais eficiente do que a validação a cada tupla inserida durante a carga dos registros.
disable_indexes	IN p_schemaname text, IN p_tablename text	Desabilita os índices da relação especificada por p_schemaname e p_tablename, com o objetivo de diminuir o tempo de carga das tuplas.
enable_indexes	IN p_schemaname text, IN p_tablename text	Habilita os índices da relação especificada por p_schemaname e p_tablename. Tal procedimento é mais eficiente do que a atualização dos índices da relação a cada tupla inserida.
reset_sequence	IN p_schemaname text, IN p_tablename text	Redefine o valor inicial da sequência utilizada para a geração de chave primária da relação especificada por p_schemaname e p_tablename.
reset_sequences	IN p_schemaname text	Redefine o valor inicial das sequências utilizadas para a geração de chaves primárias das relações presentes no esquema especificado em p_schemaname.

c) Tabelas externas: podem ser usadas em consultas como uma tabela regular, mas uma tabela externa não possui armazenamento no BD. Para buscar dados da fonte externa ou transmitir dados para a tabela externa o BD faz uso de bibliotecas específicas chamadas *Foreign Data Wrappers* ou FDW.

d) Índices especializados: cada tipo de índice usa um algoritmo diferente que é mais adequado para diferentes tipos de cláusulas indexáveis, as quais utilizam uma lógica diferente do algoritmo *btree* utilizado em atributos alfanuméricos. Pode-se destacar os índices: espacial, espaço-temporal, hash, textual, entre outros.

Essas funcionalidades são detalhadas a seguir.

Tabela 10 – Particionamento da relação MAT

Relação	Tuplas	Partição	Tuplas
master.MAT	292940	partitions.MAT_2007_05_27	2000
		partitions.MAT_2007_05_28	11526
		partitions.MAT_2007_05_29	11598
		partitions.MAT_2007_05_30	11490
		partitions.MAT_2007_05_31	11719
		partitions.MAT_2007_06_01	11599
		partitions.MAT_2007_06_02	7053
		...	
		partitions.MAT_2007_06_25	60
		Total	292940

4.9.1 Particionamento de objetos

Para que relações grandes possam ser divididas em objetos físicos menores podem ser usados mecanismos de particionamento. Isso pode resultar em maior desempenho ao consultar e manipular tais relações. Especificamente, a relação MAT e a relação POINT são particionadas em tempo de carga das MATs usando particionamento por faixa, onde cada partição de MAT contém as trajetórias que começam em uma determinada data e hora (atributo MAT_DATE) e cada partição de POINT os pontos das trajetórias que foram registrados em uma data e hora (atributo T), sendo que os períodos podem variar como dia, semana, mês e assim por diante.

Para executar um particionamento usa-se o procedimento *util.create_partitions_by_date*, disponibilizado na criação da extensão do BD, descrito na Tabela 9. Com isso, são criadas automaticamente no esquema partitions do BD as partições de acordo com um intervalo de datas, gerando por exemplo, uma partição para cada dia dentro do período, como mostrado na Tabela 10. Por extensão, podem ser definidos índices nas partições da mesma forma que na relação original e uma vantagem importante do mecanismo de particionamento é que as restrições e os índices definidos nas relações particionadas são propagados automaticamente para as partições. Similarmente, consultas nas tabelas particionadas são propagadas para as partições, utilizando paralelismo para acesso às mesmas.

4.9.2 Tabelas temporárias

Uma tabela temporária no MobilityDB, como o próprio nome indica, é uma tabela que existe para uma determinada sessão e é eliminada automaticamente quando a sessão é fechada. Elas são particularmente úteis para armazenar resultados intermediários ou dados temporários dentro de um contexto específico. Aqui está uma descrição dos principais recursos das tabelas temporárias:

a) Tabelas Permanentes persistem no BD até serem descartadas explicitamente enquanto que tabelas temporárias, são descartadas automaticamente.

b) tabelas temporárias com escopo de sessão existem durante a sessão do usuário enquanto que tabelas temporárias com escopo de transação existem apenas durante a transação atual.

c) Uma tabela temporária não pertence a nenhum esquema.

d) Uma tabela temporária pode ser indexada.

Podem ser destacadas as seguintes vantagens das tabelas temporárias:

a) Isolamento e encapsulamento: As tabelas temporárias permitem encapsular e isolar dados dentro de uma sessão ou transação específica, evitando interferência com outros usuários ou processos;

b) Resultados intermediários: útil para armazenar resultados intermediários durante consultas complexas ou manipulações de dados. Isso pode melhorar o desempenho ao dividir operações complexas em etapas mais simples;

c) Redução da sobrecarga de bloqueio: Como as tabelas temporárias são específicas da sessão ou da transação, elas não causam contenção com outros usuários ou transações. Isso pode reduzir a sobrecarga de bloqueio e aumentar a simultaneidade.

d) Lógica de transação simplificada: as tabelas temporárias podem simplificar a lógica da transação, fornecendo uma maneira de armazenar e referenciar dados temporários dentro de uma transação. Isso pode tornar consultas complexas mais legíveis e gerenciáveis;

e) SQL dinâmico e consultas dinâmicas: tabelas temporárias são valiosas ao lidar com SQL dinâmico ou quando a estrutura do conjunto de resultados não é conhecida em tempo de compilação. Eles permitem criar e manipular tabelas dinamicamente;

f) Desempenho aprimorado em determinados cenários: Em alguns casos, o uso de tabelas temporárias pode levar a melhorias de desempenho, especialmente ao lidar com grandes conjuntos de dados ou manipulações complexas de dados. No entanto, isso depende do caso de uso específico e dos recursos de otimização do mecanismo de BD;

g) Evitando armazenamento de dados a longo prazo: as tabelas temporárias são ideais para situações em que se necessita armazenar dados temporariamente, mas não quer sobrecarregar o BD com necessidades de armazenamento de longo prazo. Eles fornecem uma maneira limpa e automática de gerenciar dados temporários.

No MASTERMobilityDB o uso de tabelas temporárias é bastante empregado em consultas complexas e/ou que manipulam um grande volume de tuplas que, quando divididas em consultas mais simples com seus resultados armazenados em tabelas temporárias indexadas, apresentam um desempenho superior e menor consumo de recursos.

4.9.3 Tabelas externas

Tabelas externas ou Foreign Data Wrappers (FDW) é um recurso prático do MobilityDB que permite que se possa trabalhar com dados remotos ou em outros formatos tais como: arquivos CSV, planilhas, documentos JSON e outros tipos de BDs. FDWs são extensões do BD que permitem aos usuários definir como se comunicar com fontes de dados externas, como diferentes BDs ou sistemas de arquivos, e definir tabelas externas que acessam tais fontes de dados. Tratam-se de extensões bastante flexíveis e que suportam diferentes fontes de dados constituindo uma ferramenta poderosa para integrar dados de diferentes fontes em um único BD.

A arquitetura do FDW é baseada na especificação SQL/MED (SQL Management of External Data) e permite aos usuários gerenciar tabelas externas de forma semelhante às tabelas locais. O recurso FDW requer a instalação da extensão específica e configurações apropriadas para criar servidores externos e tabelas externas. Isso significa dados de diferentes fontes podem ser consultadas e manipuladas em um único BD, sem precisar transferir os dados entre BDs ou sistemas.

Dentre os principais FDWs disponíveis podemos citar:

a) postgres_fdw: permite acessar dados de outro BD PostgreSQL, no mesmo servidor ou em um servidor remoto.

b) file_fdw: permite acessar dados armazenados em arquivos simples, como arquivos CSV ou TSV, como se fossem tabelas no BD.

c) oracle_fdw: permite acessar dados de um BD Oracle.

d) mysql_fdw: permite acessar dados de um BD MySQL.

e) mongodb_fdw: permite acessar dados de um BD MongoDB.

No MASTERMobilityDB o uso de tabelas externas é muito importante na extração de dados em arquivos CSV, por esta razão a extensão file_fdw é instalada automaticamente, além disso no processo de extração limpeza e carga (ETL), são configurados: a) o diretório onde os arquivos de entrada estão disponíveis através de um objeto servidor, b) um mapeamento de usuário com as devidas permissões e c) uma tabela externa para cada arquivo CSV.

Esta abordagem facilita o trabalho com os dados de entrada dispensando a necessidade de gerenciamento dos mesmos em tabelas na área de preparação (staging) dos dados, podendo ser combinada com o uso tabelas temporárias.

4.9.4 Índices especializados

No PostgreSQL o qual deriva o MobilityDB, GiST (Árvore de Pesquisa Generalizada) e SP-GiST (Árvore de Pesquisa Generalizada Particionada no Espaço) são dois tipos de estruturas de índice que oferecem recursos de indexação especializados para determinados tipos de dados. Outro tipo de índice chamado índice parcial foi utilizado na indexação dos valores dos atributos dos aspectos.

GiST é uma estrutura de índice genérica que pode ser usada para vários tipos de dados e métodos de indexação. É particularmente adequado para tipos de dados complexos e oferece suporte a uma ampla variedade de consultas. GiST é extensível e pode ser usado para criar índices para tipos de dados definidos pelo usuário. Isso o torna uma escolha versátil para tipos de dados não padronizados e estruturas de dados complexas. GiST é comumente usado para tipos de dados espaciais, como pontos, linhas e polígonos, permitindo indexação espacial eficiente e otimização de consulta. Também é usado para pesquisa de texto completo, onde pode agilizar pesquisas em grandes corpos de texto. GiST pode ser adaptado a vários tipos de dados e permite que os usuários definam seus próprios métodos de pesquisa personalizados. Adequado para dados multidimensionais, tornando-o uma boa escolha para índices espaciais. Entretanto, a implementação de um índice GiST personalizado pode ser complexa, e um design eficiente geralmente requer um conhecimento profundo do tipo de dados que está sendo indexado.

SP-GiST é uma extensão do índice GiST projetada para lidar com estruturas de dados particionadas em espaço. É particularmente útil para indexar dados que podem ser divididos naturalmente em regiões não sobrepostas. SP-GiST indexa dados particionando o espaço do índice em regiões não sobrepostas, cada uma associada a uma chave específica. SP-GiST é frequentemente usado para estruturas de dados hierárquicas e de rede onde os dados podem ser divididos em partições não sobrepostas de forma eficiente. Adequado para dados que podem ser divididos em intervalos distintos, como intervalos de datas. SP-GiST foi projetado para particionar eficientemente o espaço de dados, proporcionando benefícios para certos tipos de consultas e estruturas de dados. Assim como o GiST, o SP-GiST é extensível, permitindo aos usuários definir métodos de pesquisa personalizados para tipos de dados específicos. Projetar e implementar índices SP-GiST personalizados pode ser complexo e exigir um conhecimento profundo da estrutura de dados que está sendo indexada.

O otimizador de consultas do BD pode usar automaticamente os índices GiST ou SP-GiST quando apropriado. Nenhum *hint* ou diretriz específica de otimização é necessária.

Índices GiST no MobilityDB foram implementados para otimizar o armazenamento e a recuperação de dados espaço-temporais. Os índices GiST são adequados para suportar consultas de intervalo e operações espaciais, que são comuns em apli-

cações relacionadas à mobilidade. Os índices SP-GiST também podem ser usados no MobilityDB, especialmente para cenários onde os dados podem ser particionados de forma eficiente de maneira particionada por espaço. No entanto, o uso específico de índices SP-GiST no MobilityDB pode depender da distribuição de dados e dos padrões de acesso. Os índices GiST e SP-GiST podem ser benéficos no MobilityDB para otimizar consultas de intervalo ao longo do tempo e operações espaciais. Por exemplo, encontrar trajetórias que se sobrepõem no espaço e no tempo. A integração do MobilityDB com os índices GiST e SP-GiST é particularmente importante ao lidar com objetos em movimento, onde a indexação eficiente das dimensões espaciais e temporais é crucial para o desempenho. No MobilityDB, para o tipo de dados *tgeompoint*, foram implementados os comportamentos GiST e SP-GiST personalizados para otimizar a indexação para trajetórias brutas.

No MASTERMobilityDB a otimização de consultas pode se dar no nível alfanumérico onde são utilizados índices alfanuméricos para a busca em relação aos aspectos e seus atributos funcionando como um filtro primário da consulta e índices GiST ou SP-GiST para acelerar as operações e as buscas espaciais.

Em resumo, os índices GiST e SP-GiST fornecem opções de indexação flexíveis e eficientes para uma variedade de tipos de dados, incluindo estruturas de dados espaciais e particionadas por espaço. São ferramentas valiosas para otimizar consultas em dados de mobilidade.

```
CREATE INDEX XIE2_MAT ON MAT USING GIST (RAW_TRAJECTORY) ;
```

Por fim, tem-se os índices parciais que são um recurso que permite criar um índice em um subconjunto de uma tabela, com base em uma condição especificada. No MASTERMobilityDB isto pode ser útil na criação de índices alfanuméricos, desconsiderando-se outros tipos de valores como no atributo VALUE da tabela ASPECT_ATRIBUTE.

```
CREATE INDEX XIE1_ASPECT_ATTRIBUTE ON ASPECT_ATTRIBUTE USING BTREE(VALUE)  WHERE DATA_TYPE = 1;
```

4.10 EXEMPLOS DE CONSULTAS UTILIZANDO O MASTERMOBILITYDB

Esta seção apresenta alguns exemplos de consultas sobre o *dataset* BerlinMOD que podem ser respondidas com a ajuda do MASTERMoblityDB, enfatizando o uso dos aspectos semânticos que enriquecem as MATs. O BerlinMod é um dataset de trajetórias semanticamente enriquecidas de veículos. Mais detalhes sobre este dataset são explicados na Seção 5.2.1.

Exemplo 5.1 Quantas viagens de carro ocorreram na segunda-feira pela manhã com duração entre 15 e 30 minutos?

```
SELECT COUNT(M. MAT_ID )
FROM MASTER. MOVING_OBJECT_FIND_BY_ATTRIBUTE( ' Vehicle ', ' t y p e ', ' passenger ' )  MO
   INNER JOIN MASTER.MAT M USING(MO_ID)
```

```
WHERE EXTRACT(DOW FROM M.MAT_DATE) = 1 --Monday
  AND EXTRACT(HOUR FROM M.MAT_DATE) BETWEEN 6 AND 12 --Morning
  AND DURATION(M.RAW_TRAJECTORY)
    BETWEEN INTERVAL '15 minutes' AND INTERVAL '30 minutes';
```

De acordo com o modelo físico do MASTER, devem ser selecionados os veículos de passageiros, e as respectivas viagens filtrando as mesmas através do operador extract do PostgreSQL que retorna o dia da semana e da função duration do MobilityDB que retorna a duração da trajetória.

Exemplo 5.2 Retornar o id e a velocidade máxima das trajetórias que incluem pelo menos a passagem por 5 Locais de interesse.

```
WITH POI_PASSED AS (
  SELECT P.MAT_ID, COUNT(P.POINT_ID) P_COUNT
  FROM MASTER.POINT_FIND_BY_ATTRIBUTE('POI', NULL, NULL) P
  GROUP BY P.MAT_ID
)
SELECT M.MAT_ID, MAXVALUE(SPEED(M.RAW_TRAJECTORY)) MAT_SPEED
FROM POI_PASSED P
  INNER JOIN MASTER.MAT M USING(MAT_ID)
WHERE P.P_COUNT >= 5;
```

Neste exemplo são selecionados para cada trajetória a quantidade de pontos que passam por algum POI, filtrando as trajetórias com pelo menos 5 pontos para as quais é calculada a velocidade máxima (operadores SPEED e MAXVALUE).

Exemplo 5.3 Retornar o id, a distancia percorrida, e a geometria das trajetórias que passaram pela sequência de locais de interesse: 81, 49.

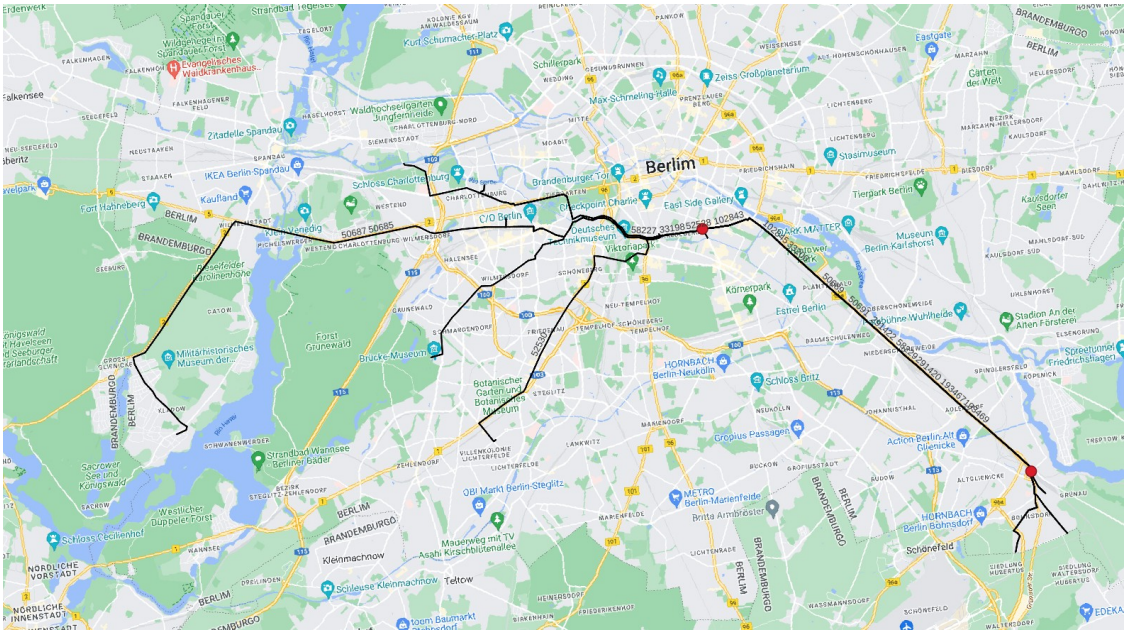
```
WITH POI_PASSED AS (
    SELECT P.MAT_ID, ARRAY_AGG(ASP.DESCRIPTION ORDER BY P.T) POIS
      , ST_UNION(ARRAY_AGG(ST_TRANSFORM(
        ST_SETSRID(ST_MAKEPOINT(ASP.X, ASP.Y), 4326),5676)
        ORDER BY P.T)) COORD
    FROM MASTER.POINT_FIND_BY_ATTRIBUTE('POI', NULL, NULL) P
      CROSS JOIN LATERAL UNNEST(P.ASPECT_A) AS 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['81','49'] AS VARCHAR(50)[])
  AND MAT_DATE BETWEEN '2007-06-11 00:00:00' AND '2007-06-11 23:59:59';
```

Aqui são selecionadas as MATs, cuja sequência de pontos contém o array ['81','49'], que são os identificadores dos pontos de interesse, juntamente com a sequência de locais visitados, a distancia percorrida sendo retornada pela função LENGTH e a geometria da trajetória pela função TRAJECTORY.

A Figura 14 mostra uma representação gráfica gerada pela ferramenta QGIS do resultado da consulta exemplo 5.3. As linhas em preto apresentam a geometria das trajetórias selecionadas e os pontos em vermelho representam os locais de interesse.

Exemplo 5.4. Produzir um histograma com a distância percorrida pelas viagens.

Figura 14 – Resultado da consulta exemplo 5.4




```
WITH BUCKETS (BUCKET_NO, BUCKET_RANGE)AS (
   SELECT 1 , FLOATRANGE ' [ 0 ,␣1 ) ' UNION
   SELECT 2 , FLOATRANGE ' [ 1 ,␣2 ) ' UNION
   SELECT 3 , FLOATRANGE ' [ 2 ,␣5 ) ' UNION
   SELECT 4 , FLOATRANGE ' [ 5 ,␣1 0 ) ' UNION
   SELECT 5 , FLOATRANGE ' [ 1 0 ,␣5 0 ) ' UNION
   SELECT 6 , FLOATRANGE ' [ 5 0 ,␣100) '  ) ,
HISTOGRAM AS (
   SELECT B .BUCKET_NO, B .BUCKET_RANGE,COUNT(M. MAT_ID ) AS FREQ
   FROM BUCKETS B
      LEFT OUTER JOIN master . mat  m
         ON LENGTH(m. r a w _ t r a j e c t o r y ) 1E3 <@ B.BUCKET_RANGE
   GROUP BY B.BUCKET_NO, B.BUCKET_RANGE
   ORDER BY B.BUCKET_NO, B .BUCKET_RANGE )
SELECT H.BUCKET_NO, H.BUCKET_RANGE, H.FREQ,
   REPEAT( 'O ' , ( FREQ : : FLOAT / MAX(FREQ) OVER ( ) * 30 ) : : INT ) AS BAR
FROM HISTOGRAM H;
```

Na listagem abaixo pode ser verificada a distribuição das viagens de acordo com a distância em Km.

```
bucket_no | bucket_range | f r e q      | b a r                                |
---------+------------+------+------------------------------+
        1 | [ 0 , 1 )          |152404|OOOOOOOOOOOOOOOOOOOOOOOOOOOOOO|
        2 | [ 1 , 2 )          |  10492|OO                                    |
        3 | [ 2 , 5 )          |  39212|OOOOOOOO                              |
        4 | [ 5 , 1 0 )        |  22751|OOOO                                  |
        5 | [ 1 0 , 5 0 )      |  68079|OOOOOOOOOOOOO                         |
        6 | [ 5 0 , 1 0 0 )    |      2 |                                      |
```

Por fim, mostra-se o plano de execução destacando o uso das partições da relação MAT.

```
WindowAgg  ( c o s t =93055.43..93073.30  rows=6  w i d t h =76)
->  GroupAggregate  ( c o s t =93055.43..93073.07  rows=6  w i d t h =44)
   Group Key :  ( 1 ) , ( ' [ 0 , 1 ) ' : : f l o a t r a n g e )
```

```
->  Sort  (cost=93055.43..93059.83  rows=1758  width=40)
    SortKey: (1), ('[0,1)'::floatrange)
    ->  Nested Loop  Left Join  (cost=0.18..92960.60 rows=1758  width=40)
      Join Filter: ((length(m.raw_trajectory)'1000'::double precision) <@ ('[0,1)'::floatrange))
      ->  HashAggregate  (cost=0.18..0.24 rows=6  width=36)
          Group Key:  (1), ('[0,1)'::floatrange)
          ->  Append  (cost=0.00..0.15 rows=6  width=36)
            ->  Result (cost=0.00..0.01 rows=1  width=36)
            ->  Result (cost=0.00..0.01 rows=1  width=36)
            ->  Result (cost=0.00..0.01 rows=1  width=36)
            ->  Result (cost=0.00..0.01 rows=1  width=36)
            ->  Result (cost=0.00..0.01 rows=1  width=36)
            ->  Result (cost=0.00..0.01 rows=1  width=36)
      ->  Materialize (cost=0.00..58539.98 rows=292941  width=36)
          ->  Append  (cost=0.00..57075.27 rows=292941  width=36)
            ->  Seq Scan on mat_2007_05_26 m_1  (cost=0.00..0.00 rows=1  width=36)
            ->  Seq Scan on mat_2007_05_27 m_2  (cost=0.00..83.00 rows=2000  width=36)
            ->  Seq Scan on mat_2007_05_28 m_3  (cost=0.00..2254.26 rows=11526  width=36)
            ->  Seq Scan on mat_2007_05_29 m_4  (cost=0.00..2259.98 rows=11598  width=36)
            ->  Seq Scan on mat_2007_05_30 m_5  (cost=0.00..2254.90 rows=11490  width=36)
            ->  Seq Scan on mat_2007_05_31 m_6  (cost=0.00..2289.19 rows=11719  width=36)
            ->  Seq Scan on mat_2007_06_01 m_7  (cost=0.00..2284.99 rows=11599  width=36)
            ->  Seq Scan on mat_2007_06_02 m_8  (cost=0.00..1228.50 rows=7053  width=36)
            ->  Seq Scan on mat_2007_06_03 m_9  (cost=0.00..1236.95 rows=7295  width=36)
            ->  Seq Scan on mat_2007_06_04 m_10  (cost=0.00..2254.70 rows=11470  width=36)
            ->  Seq Scan on mat_2007_06_05 m_11  (cost=0.00..2314.20 rows=11820  width=36)
            ->  Seq Scan on mat_2007_06_06 m_12  (cost=0.00..2268.61 rows=11661  width=36)
            ->  Seq Scan on mat_2007_06_07 m_13  (cost=0.00..2304.26 rows=11826  width=36)
            ->  Seq Scan on mat_2007_06_08 m_14  (cost=0.00..2297.42 rows=11642  width=36)
            ->  Seq Scan on mat_2007_06_09 m_15  (cost=0.00..1223.61 rows=7261  width=36)
            ->  Seq Scan on mat_2007_06_10 m_16  (cost=0.00..1222.18 rows=7118  width=36)
            ->  Seq Scan on mat_2007_06_11 m_17  (cost=0.00..2273.40 rows=11549  width=36)
            ->  Seq Scan on mat_2007_06_12 m_18  (cost=0.00..2281.34 rows=11634  width=36)
            ->  Seq Scan on mat_2007_06_13 m_19  (cost=0.00..2304.48 rows=11748  width=36)
            ->  Seq Scan on mat_2007_06_14 m_20  (cost=0.00..2336.32 rows=11832  width=36)
            ->  Seq Scan on mat_2007_06_15 m_21  (cost=0.00..2264.45 rows=11545  width=36)
            ->  Seq Scan on mat_2007_06_16 m_22  (cost=0.00..1220.29 rows=7329  width=36)
            ->  Seq Scan on mat_2007_06_17 m_23  (cost=0.00..1274.71 rows=7571  width=36)
            ->  Seq Scan on mat_2007_06_18 m_24  (cost=0.00..2244.95 rows=11495  width=36)
            ->  Seq Scan on mat_2007_06_19 m_25  (cost=0.00..2303.50 rows=11650  width=36)
            ->  Seq Scan on mat_2007_06_20 m_26  (cost=0.00..2327.94 rows=11794  width=36)
            ->  Seq Scan on mat_2007_06_21 m_27  (cost=0.00..2283.46 rows=11546  width=36)
            ->  Seq Scan on mat_2007_06_22 m_28  (cost=0.00..2273.88 rows=11688  width=36)
            ->  Seq Scan on mat_2007_06_23 m_29  (cost=0.00..1233.02 rows=7313  width=36)
            ->  Seq Scan on mat_2007_06_24 m_30  (cost=0.00..1201.08 rows=7108  width=36)
            ->  Seq Scan on mat_2007_06_25 m_31  (cost=0.00..10.60 rows=60  width=36)
```

O próximo capítulo apresenta uma avaliação experimental realizada sobre o MASTERMobilityDB.

5 AVALIAÇÃO EXPERIMENTAL

Este capítulo é dedicado aos experimentos para avaliar o desempenho do MASTERMobilityDB e ao atendimento aos requisitos funcionais desta camada de persistência proposta. Inicialmente será apresentado o protocolo experimental dos testes e em seguida os resultados.

Conforme a implementação do MASTERMobilityDB, descrita no capítulo anterior, acredita-se que o mesmo atende aos requisitos funcionais definidos na seção 4.1, para validar esta hipótese foram realizados testes de aceitação dos nove requisitos funcionais através da verificação dos critérios de aceite dos mesmos, por meio da coleta de evidências.

Além do teste de aceitação é necessário realizar os testes de desempenho do MASTERMobilityDB em comparação com o modelo de trajetórias simbólicas apresentado pelo BD Secondo frente a um alto volume de dados de entrada. Nesta avaliação foi utilizado um conjuntos de dados conhecido: BerlinMOD (DÜNTGEN; BEHR; GÜTING, 2009) no qual dados de pontos em movimento são amostrados de automóveis simulados circulando na rede viária da capital alemã, Berlim.

Todos os tempos de computação mencionados e a avaliação dos requisitos funcionais neste capítulo foram alcançados em um computador Intel Core I7 2.8 GHz com 16 GB de memória principal e HD SSD de 512 GB, rodando Linux Ubuntu 22.04. O MasterMobilityDB foi testado na sua versão 1.0 enquanto que para o BD Secondo foi utilizada a versão 4.4.0. Os tempos de execução foram determinados pela execução de diversas consultas envolvendo aspectos semânticos e espaço-temporais variando a amostra dos dados.

5.1 TESTES DE ACEITAÇÃO

Segundo (COHN, 2004), critérios de aceite são as condições que um produto de software ou um projeto deve cumprir para ser aceito por um usuário, cliente ou outros interessados. Eles são uma parte crucial dos requisitos e são usados para estabelecer os limites e requisitos de uma funcionalidade ou tarefa. Os critérios de aceitação servem como uma lista detalhada e precisa de condições sob as quais um requisito é considerado como atendido, sendo uma maneira de garantir que todos os envolvidos tenham um entendimento comum do que é esperado. Eles ajudam a esclarecer o que os desenvolvedores devem construir antes de começarem a trabalhar, orientam os testadores sobre o que testar e auxiliam os interessados a entenderem o que receberão. Isso garante que o produto final atenda às necessidades dos usuários e esteja em conformidade com os requisitos especificados no início do projeto. Eles desempenham um papel chave em metodologias Ágeis, mas são aplicáveis em vários processos de gerenciamento de projetos e desenvolvimento de software.

Ainda segundo (COHN, 2004), testes de aceitação são uma fase crítica no ciclo de desenvolvimento de software, na qual se verifica se o sistema atende aos critérios de aceite estabelecidos, garantindo assim que as necessidades e requisitos do usuário final sejam satisfeitos antes do lançamento do produto. Esses testes são realizados numa etapa posterior ao desenvolvimento, e servem como uma validação final de que o software está pronto para ser entregue.

Na tabela 11 tem-se a relação dos requisitos funcionais do MASTERMobilityDB, os critérios de aceite, a verificação se o mesmo atende ou não e as evidências para o aceite.

A seguir são apresentados os testes de aceitação onde são mostrados: (i) os requisitos funcionais do MASTERMobilityDB, (ii) os critérios de aceite, (iii) a verificação se o mesmo atende ou não e (iv) as evidências para o aceite.

Tabela 11 – Testes de aceitação do MASTERMobilityDB

Teste:	**Teste 01**
Requisito:	R1
Crit. aceite:	Dado um conjunto de dados espaço-temporais de entrada de objetos em movimento, quando tais dados forem transformados em trajetórias então o MASTERMobilityDB deve oferecer um tipo de dados e funcionalidades para a persistência e a recuperação da geometria da trajetória.
Verificado:	Sim
Explicação:	O MASTERMobilityDB possui o tipo MAT_TYP que guarda a geometria da trajetória no atributo TRAJECTORY_GEOM como pode ser verificado no modelo físico na Figura 11. A rotina TGEOMPOINT_SEQ persiste a trajetória com base nos dados espaço-temporais do objeto em movimento. A função TRAJECTORY do MobilityDB retorna a geometria da trajetória.

Teste:	**Teste 02**
Requisito:	R2
Crit. aceite:	Dada uma MAT persistida no banco de dados, quando o usuário acessar a mesma, então o sistema deve apresentar primitivas espaciais para a manipulação da geometria da trajetória.
Verificado:	Sim

Explicação:	O MASTERMobilityDB herda tanto do MobilityDB quanto do PostGis diversas primitivas para manipulação da geometria da trajetória (seção 2.7), que é persistida no tipo MAT_TYP, no atributo TRAJECTORY_GEOM

Teste:	**Teste 03**
Requisito:	R3
Crit. aceite:	Dada uma MAT persistida no banco de dados, quando o usuário consultar a mesma, então o MASTERMobilityDB deve prover índices espaciais para otimizar o acesso a mesma.
Verificado:	Sim
Explicação:	O MASTERMobilityDB possui os índices especializados GIST e SP-GIST que oferecem recursos para a indexação de trajetórias e da sua geometria, diminuindo o tempo de acesso, como pode ser verificado na seção 4.9.4.

Teste:	**Teste 04**
Requisito:	R4
Crit. aceite:	Dado o modelo de representação MASTER quando o mesmo for implementado em um banco de dados então este BD deve permitir a implementação do MASTER através de tipos abstratos de dados.
Verificado:	Sim
Explicação:	Com base no modelo físico do MASTER, o MASTERMobilityDB define uma série de tipos abstratos de dados no BD MobilityDB para suportar as MATs, os pontos, os *moving objects*, os aspectos, os atributos, entre outros (seção 4.6) e os mesmos são utilizados nas operações e nas funções definidas dentro da API do MASTERMobilityDB.

Teste:	**Teste 05**
Requisito:	R5
Crit. aceite:	Dada uma trajetória de múltiplos aspectos quando a mesma for persistida no banco de dados então o MASTERMobilityDB deve persistir as informações semânticas da MAT.
Verificado:	Sim
Explicação:	Os tipos abstratos de dados e as operações (seção 4.7) do MASTERMobilityDB tratam da persistência da MAT (tipo MAT_TYP), dos seus aspectos (tipo ASPECT_TYPE) e dos seus atributos(ATTRIBUTE_TYP), que compõem a parte semântica da MAT.

Teste:	**Teste 06**
Requisito:	R6
Crit. aceite:	Dada uma trajetória de múltiplos aspectos quando a mesma for persistida no banco de dados então o MASTERMobilityDB deve persistir tanto a trajetória bruta quanto as informações semânticas, integradas.
Verificado:	Sim
Explicação:	O tipo MAT_TYP do MASTERMobilityDB, possui o atributo RAW_TRAJECTORY onde é mantida a trajetória bruta associada a trajetória de múltiplos aspectos. O tipo POINT_TYP trata com os pontos que possuem pelo menos um aspecto associado, sendo desnecessário informar os pontos da trajetória que não possuem anotações semânticas visto que todos os pontos da MAT fazem parte da trajetória bruta.

Teste:	**Teste 07**
Requisito:	R7
Crit. aceite:	Dado o modelo de representação MASTER, quando o mesmo for implementado em um banco de dados então este BD tem que estar em uma versão estável.
Verificado:	Sim
Explicação:	O BD MobilityDB que suporta o MASTERMobilityDB se encontra na versão 1.0 que foi validada pela comunidade e encontra-se estável conforme pôde ser verificado nos testes de desempenho que estão relatados na seção 5.2 deste capítulo. Ou seja, o MobilityDB não é mais um protótipo.

Teste:	**Teste 08**
Requisito:	R8
Crit. aceite:	Dado o modelo de representação MASTER, quando o mesmo for implementado em um banco de dados então este BD tem que ser de código aberto.
Verificado:	Sim

Explicação:	Tanto O BD MobilityDB quanto o MASTERMobilityDB são softwares de código aberto podendo os fontes serem baixados do Github, respectivamente nos endereços: https://github.com/MobilityDB/MobilityDB e https://github.com/ffeller/MasterMobilityDB.

Teste:	**Teste 09**
Requisito:	R9
Crit. aceite:	Dada uma trajetória de múltiplos aspectos, quando a mesma for persistida no banco de dados, então o MASTERMobilityDB deve permitir a persistência de um número variável de atributos e de diferentes tipos de dados.
Verificado:	Sim
Explicação:	O MASTERMobilityDB possui os tipos ASPECT_TYP e ATTRIBUTE_TYP que persistem, respectivamente, os aspectos e os atributos das MATs e cada MAT pode ter um número variável de aspectos que por sua vez podem ter um número variável de atributos de diferentes tipos de dados.

Tendo-se os testes de aceitação concluídos pode-se afirmar com certeza que o MASTERMobilityDB atende aos requisitos funcionais da camada de persistência propostos na seção 4.1.

5.2 TESTES DE DESEMPENHO

Segundo (COHN, 2004), os testes de desempenho são uma categoria de testes realizados para avaliar a velocidade, capacidade de resposta, estabilidade, escalabilidade e uso de recursos de um aplicativo de software ou sistema sob uma carga de trabalho específica. Estes testes são cruciais para compreender como um sistema se comporta em condições normais e de pico, garantindo que cumpre os critérios de desempenho exigidos pelo negócio e pelos seus utilizadores finais. Os testes de desempenho ajudam a identificar gargalos, limitações e possíveis áreas de melhoria no aplicativo antes de ele ser implantado em produção, reduzindo assim o risco de problemas de desempenho que afetam a experiência do usuário.

Nesta seção é apresentado o conjunto de dados BerlinMOD que foi utilizado no comparativo do MASTERMobilityDB com o Secondo. O BerlinMOD trata com dados sintéticos de viagens com veículos.

5.2.1 BerlinMOD

Os benchmarks se tornaram o método padrão para comparar diferentes SGBDs. Cada benchmark consiste em um conjunto de dados bem definido e uma carga de trabalho, geralmente um conjunto de consultas. Embora estruturas de dados, estruturas de índices e diferentes implementações de operadores possam ser comparadas separadamente de outros componentes, seu impacto no desempenho do banco de dados torna-se mais claro quando são testados todos juntos em um sistema real. Os benchmarks beneficiam os pesquisadores ao simplificar a configuração e a descrição dos experimentos usados para avaliar a eficiência das invenções propostas: as propriedades dos dados são bem definidas e estudadas, o risco de introdução de vieses é minimizado e torna-se mais fácil repetir experimentos. BerlinMOD é um benchmark para apoiar pesquisas no contexto de sistemas históricos de bancos de dados de objetos móveis/sistemas de bancos de dados de trajetória (DÜNTGEN; BEHR; GÜTING, 2009).

Embora o uso de dados de objetos móveis (MOD) coletados do mundo real seja amplamente considerado preferível, existem sérios problemas com a quantidade e a natureza de tais dados. Muitas vezes, as trajetórias são curtas, são poucas ou questões legais — como a privacidade dos dados ou os direitos de autor — tornam praticamente impossível a sua utilização e distribuição gratuitas. Portanto, BerlinMOD utiliza dados artificiais criado por um gerador de dados. O gerador é implementado como um script do BD Secondo e permite análise e modificação. Com configurações padrão, os dados são obtidos por observação de longo prazo (28 dias) de 2.000 posições simuladas de veículos. Os movimentos criados são representativos do comportamento dos colaboradores, que se deslocam entre a casa e o local de trabalho e realizam algumas deslocações adicionais (para visitas, compras, desporto, etc.). A simulação utiliza dados de linhas geográficas e uma combinação de estatísticas sobre localizações residenciais e de empregadores e regiões que descrevem distritos estatísticos para criar destinos representativos para viagens.

O conjunto de dados BerlinMOD[1] consiste de 292940 viagens de veículos realizadas dentro da cidade de Berlim. Tais viagens simulam o movimento de pessoas para casa, trabalho e lazer onde cada viagem foi modelada como uma trajetória de múltiplos aspectos. Seguindo a abordagem do MASTER, cada MAT contem: a) a trajetória bruta da viagem, b) os pontos anotados associados com as regiões que a trajetória cruzou, c) os locais mais próximos da trajetória como restaurantes, bares, praças, parques entre outros, se a distância for de até 1 km. As regiões, as ruas e os locais de interesse são modelados como aspectos. Os veículos além de serem modelados como objetos em movimento também são modelados como aspectos com os seus atributos como marca, modelo e placa. Por fim tem-se as relações: Licences, Instants e Periods

[1] https://secondo-database.github.io/BerlinMOD/BerlinMOD.html

que definem faixas de valores utilizados para expressar predicados de consulta em relação às viagens. As relações possuem índices em atributos tradicionais, espaciais, temporais ou espaçotemporais.

O número de trajetórias coletadas por veículo varia de 85 a 197 com média de 146 sendo que cada trajetória bruta possui entre 1 e 1899 pontos com média de 382,25 pontos e o total de pontos anotados por trajetória varia de 1 a 28 com uma média de 6,5 pontos.

5.2.2 Tipos de consulta para o BerlinMOD

BerlinMOD fornece 17 consultas de intervalo e ponto formuladas em SQL. As consultas aplicam combinações de predicados, operações e agregações padrão, temporais, espaciais e espaço-temporais simples a bastante complexos ao conjunto de dados. Ambas as consultas de seleção e junção são abordadas. A seleção permite testar uma ampla gama de estruturas de índices, métodos de acesso e implementações de operadores espaço-temporais. Para obter uma visão geral dos possíveis tipos de consulta, distinguem-se cinco aspectos das propriedades da consulta:

a) Identidade do objeto (conhecido/desconhecido): aqui distinguem-se entre consultas que começam com um objeto conhecido (p. ex. "O objeto X ...") e consultas onde não se conhece nenhum objeto em questão antecipadamente (p. ex "Quais objetos fazem ..."), respectivamente.

b) Dimensão (padrão / temporal / espacial / espaço-temporal): este critério refere-se à(s) dimensão(ões) utilizadas na consulta. "padrão"significa que nenhum atributo ou condição temporal, espacial ou espaço-temporal é motivo de preocupação para esta consulta. Na representação baseada em viagens a placa do automóvel, não é mais uma chave (um carro com uma determinada placa fará várias viagens), ou é uma chave fora da relação que contém as viagens. Neste caso, o número da placa deve ser conectado às viagens por meio de uma operação de junção.

c) Intervalo de consulta (ponto/intervalo/ilimitado): esta propriedade determina a presença/tamanho do intervalo de consulta.

d) Tipo de condição (objeto único/relações entre objetos): este aspecto é sobre se o predicado de consulta depende de uma condição referente a apenas um único objeto (como "comprimento (o1) < 10,0") ou se depende de uma relação entre dois ou mais objetos (como "o1 = o2"ou "o1 passes o2").

e) Agregação (sim/não): este atributo indica se o resultado é computado por algum tipo de agregação (soma, contagem, etc.).

Com o objetivo de endereçar os diversos tipos de consultas espaço-temporais e semânticas foi definido um conjunto de 17 consultas submetidas ao Secondo e

ao MASTERMobilityDB carregados com o dataset BerlinMOD. Estas consultas são listadas a seguir.

a) Query_1: Quais são os modelos dos veículos com placas que estão em Licences?
b) Query_2: Quantos veículos existem que são automóveis de passageiros?
c) Query_3: Onde estiveram os veículos com placas que estão em Licences em cada instante de Instants?
d) Query_4: Quais números de placas pertencem aos veículos que passaram nos locais de interesse - POIs?
e) Query_5: Qual a distância mínima entre locais onde estiveram 10 veículos sorteados de Licences e outros 10 veículos sorteados de Licences?
f) Query_6: Quais são os pares de placas de "caminhões"que estão a uma distância de 10 m ou menos um do outro?
g) Query_7: Quais são os números das placas dos carros de “passageiros” que alcançaram os POIs antes de todos os carros de “passageiros” durante o período completo de observação?
h) Query_8: Quais são as distâncias totais percorridas pelos veículos com placas de Licences durante os períodos de Periods?
i) Query_9: Qual é a maior distância percorrida por um veículo durante cada um dos períodos de Periods?
j) Query_10: Quando e onde os veículos com placas de Licences encontraram outros veículos (distância < 3m) e quais são estas últimas placas?
k) Query_11: Quais veículos passaram por um ponto dos POIs em um dos instantes de Instants?
l) Query_12: Quais veículos se encontraram em um ponto dos POIs em um instante de Instants?
m) Query_13: Quais veículos circularam dentro de uma das regiões durante os períodos de Periods?
n) Query_14: Quais veículos viajaram dentro de uma das regiões em um dos instantes de Instants?
o) Query_15: Quais veículos passaram por um POI durante um período de Periods?
p) Query_16: Liste os pares de placas para veículos, o primeiro em Licences, o segundo em Licences, onde os veículos correspondentes estão presentes em uma região durante um período de Períods, mas não se encontram naquele momento.

Tabela 12 – Seleção do tipo de consulta para o BerlinMOD

Query	Id Objeto	Dimensão	Intervalo	Tipo condição	Agregação
1	Conhecido	Padrão	Ponto	Único	Não
2	Desconhecido	Padrão	Ponto	Único	Sim
3	Conhecido	Temporal	Ponto	Único	Não
4	Desconhecido	Espacial	Ponto	Relação	Não
5	Conhecido	Espacial	Ilimitado	Relação	Não
6	Conhecido	Espaço-temporal	Ilimitado	Relação	Não
7	Conhecido	Espacial	Ilimitado	Relação	Não
8	Conhecido	Temporal	Intervalo	Único	Não
9	Desconhecido	Temporal	Intervalo	Único	Sim
10	Conhecido	Espaço-temporal	Intervalo	Relação	Não
11	Conhecido	Espaço-temporal	Ponto	Único	Não
12	Desconhecido	Espaço-temporal	Ponto	Relação	Não
13	Desconhecido	Espaço-temporal	Intervalo	Único	Não
14	Desconhecido	Espaço-temporal	Intervalo	Único	Não
15	Desconhecido	Espaço-temporal	Intervalo	Único	Não
16	Desconhecido	Espaço-temporal	Intervalo	Relação	Não
17	Desconhecido	Espaço-temporal	Ilimitado	Relação	Não

q) Query_17: Quais POIs foram visitados por um número máximo de veículos diferentes?

Na tabela 12 tem-se a seleção das características das consultas para o dataset BerlinMOD.

5.3 PROTOCOLO EXPERIMENTAL PARA O BENCHMARK UTILIZANDO O BERLINMOD

Para realizar os experimentos com o dataset BerlinMOD primeiramente foi necessário construir os procedimentos de ETL com base nas operações e funções oferecidas pelo MASTERMobilityDB com o objetivo de construir as MATs dos objetos em movimento, no caso, as viagens de carro dentro da cidade de Berlim. Tais procedimentos foram convertidos para o BD Secondo utilizando a extensão para construção de trajetórias simbólicas de modo que se pudesse realizar os comparativos.

Tendo-se os dados consolidados em cada plataforma, o experimento consistiu basicamente na aplicação das 17 consultas envolvendo operadores alfanuméricos, espaçotemporais e de trajetórias no MASTERMobilityDB e no Secondo. As medições são realizadas variando-se o percentual da amostra dos dados de entrada: 10%, 40%, 70% e 100% em relação a quantidade de trajetórias carregadas, como pode ser visto na tabela 13. Com base na relação completa, são criadas aleatoriamente 4 subrelações com cardinalidades variando de acordo com o percentual da amostra.

Tabela 13 – Tamanho da amostra e dos dados de entrada

Amostra	Qtd MATs	Tam MASTER	Tam Secondo	Diferença	Percentual
10%	29294	0,30GB	3,30GB	-3,00GB	-1000,00%
40%	117176	1,13GB	14,61GB	-13,48GB	-1189,50%
70%	205058	3,99GB	24,86GB	-20,87GB	-523,06%
100%	292940	9,35GB	39,45GB	-30,10GB	-321,93%

Na tabela 13, tem-se para cada percentual de amostra, a quantidade de trajetórias no BD e o tamanho em GB dos BDs MASTERMobilityDB e Secondo.

A seqüência de passos para a execução do benchmark é a seguinte:

a) É recuperado um percentual, que pode ser 10%, 40%, 70% ou 100%, das MATs persistidas no banco de dados.

b) Gera-se uma nova relação com os dados sorteados.

c) Aplica-se o benchmark contendo as 17 consultas para que execute com as MATs que foram sorteadas, no qual os tempos de execução das consultas são computados.

d) Repete-se o procedimento até chegar ao percentual de 100

5.4 RESULTADOS DA EXECUÇÃO DO BENCHMARK BERLINMOD

Através do conjunto de consultas BerlinMOD foi desenvolvido um benchmark para testar o MASTERMobilityDB em comparação com o BD Secondo, com suas álgebras de extensão para trajetórias simbólicas. O benchmark foi realizado com consultas executáveis traduzidas manualmente no mesmo PC que foi usado para gerar os dados de benchmark. Esta seção relata a série de experimentos conduzida com dados sintéticos baseados no BerlinMOD.

Os resultados da execução do benchmark são apresentados em cinco tabelas: 14, 15, 16, 17 e 18, cada uma contendo os tempos de execução para um percentual de amostra dos dados de entrada. A tabela 14 apresenta os tempos de carga para as trajetórias e as tabelas 15, 16, 17 e 18 possuem o mesmo formato apresentando: (i) o nome da consulta executada, (ii) o tempo de resposta no MASTERMobilityDB, (iii) o tempo de resposta no BD Secondo, (iv) a diferença de tempo entre as duas soluções, e (v) a diferença de tempo em percentual. Deve-se observar que o valores negativos na quarta e na quinta colunas indicam que o tempo do MASTERMobilityDB foi inferior ao Secondo e um desempenho superior do primeiro.

Os gráficos de tempo de execução correspondentes à aplicação das consultas mencionadas na seção 5.1 a essas subrelações estão representados na Figura 15. Os resultados da abordagem utilizando o MASTERMobilityDB são ilustrados no diagrama

Tabela 14 – Tempos de carga em (s) das trajetórias

Amostra %	MASTERMobilityDB	Secondo	Diferença	Percentual
10%	416,33	369,23	47,09	12,75%
40%	1692,19	1319,07	373,12	28,29%
70%	3597,46	2712,15	885,31	32,64%
100%	3322,46	4824,54	1502,082	45,21%

Tabela 15 – Tempos de execução das consultas em (s) para 10% das trajetórias

Query #	MASTERMobilityDB	Secondo	Diferença	Percentual
Query 1	0,04	0,05	-0,01	-35,95%
Query 2	0,02	0,05	-0,03	-160,07%
Query 3	0,09	0,08	0,01	15,02%
Query 4	0,19	3,95	-3,76	-1998,24%
Query 5	0,15	3,79	-3,65	-2487,01%
Query 6	0,43	33,01	-32,58	-7553,66%
Query 7	0,81	1,55	-0,74	-91,80%
Query 8	0,04	0,10	-0,06	-143,20%
Query 9	5,29	163,06	-157,77	-2982,29%
Query 10	0,08	16,97	-16,89	-20868,28%
Query 11	0,31	0,07	0,24	76,81%
Query 12	0,03	0,09	-0,06	-232,98%
Query 13	0,10	1,63	-1,52	-1468,82%
Query 14	0,72	0,20	0,52	72,14%
Query 15	0,16	0,16	0,00	2,31%
Query 16	0,22	6,94	-6,72	-3027,56%
Query 17	0,27	1,57	-1,30	-491,32%

à esquerda, enquanto os gráficos à direita são referentes a aplicação das consultas no Secondo.

Para facilitar a visualização dos tempos de execução dividiram-se as consultas em três grupos: a) grupo 1: tempo de execução até 1,6s; b) grupo 2: tempo de execução até 45s e. c) grupo 3: tempo de execução até 3000s. No primeiro grupo observou-se um comportamento linear nos dois BDs com uma pequena diferença na Query_14 a favor do MASTERMobilityDB e na Query_3 para o Secondo. No segundo grupo, diferenças no tempo de execução foram observadas a favor do MASTERMobilityDB na ordem de até 10 para 1, comportamento justificado em função do particionamento de objetos disponível no MASTERMobilityDB. Por fim, no terceiro grupo, onde foram observadas as maiores diferenças no tempo de execução, aconteceu pelo uso de índices nas operações espaciais aplicadas nas trajetórias no MASTERMobilityDB.

Uma primeira limitação das trajetórias simbólicas do BD Secondo é a repre-

Tabela 16 – Tempos de execução das consultas em (s) para 40% das trajetórias

Query #	**MASTERMobilityDB**	**Secondo**	**Diferença**	**Percentual**
Query 1	0,04	0,04	-0,01	-18,94%
Query 2	0,02	0,04	-0,02	-128,47%
Query 3	0,18	0,11	0,07	41,63%
Query 4	0,35	19,73	-19,37	-5472,76%
Query 5	0,29	19,11	-18,82	-6512,55%
Query 6	3,40	588,02	-584,61	-17172,40%
Query 7	1,80	8,45	-6,65	-368,26%
Query 8	0,07	0,22	-0,15	-215,18%
Query 9	23,14	738,96	-715,82	-3093,75%
Query 10	0,16	221,67	-221,51	-139499,22%
Query 11	0,11	0,07	0,04	36,18%
Query 12	0,03	0,08	-0,05	-158,75%
Query 13	0,38	4,98	-4,60	-1214,05%
Query 14	0,76	0,44	0,33	42,80%
Query 15	0,19	0,26	-0,08	-40,75%
Query 16	0,30	16,42	-16,12	-5387,25%
Query 17	1,24	6,84	-5,60	-452,19%

Tabela 17 – Tempos de execução das consultas em (s) para 70% das trajetórias

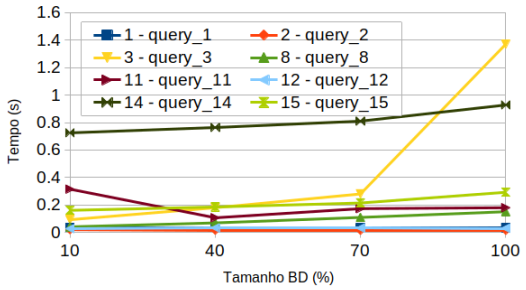
Query #	**MASTERMobilityDB**	**Secondo**	**Diferença**	**Percentual**
Query 1	0,04	0,02	0,01	35,70%
Query 2	0,02	0,02	-0,01	-41,62%
Query 3	0,28	0,09	0,19	68,93%
Query 4	0,51	20,10	-19,59	-3858,12%
Query 5	0,45	15,63	-15,18	-3371,93%
Query 6	7,11	1.253,52	-1.246,40	-17519,48%
Query 7	2,67	8,28	-5,61	-210,67%
Query 8	0,11	0,23	-0,12	-110,64%
Query 9	38,80	1.131,39	-1.092,59	-2815,70%
Query 10	0,24	701,91	-701,67	-293420,81%
Query 11	0,17	0,11	0,07	38,21%
Query 12	0,03	0,10	-0,07	-224,34%
Query 13	0,39	9,60	-9,21	-2367,00%
Query 14	0,81	0,87	-0,06	-7,38%
Query 15	0,21	0,44	-0,23	-107,33%
Query 16	0,30	28,67	-28,37	-9374,62%
Query 17	1,24	13,97	-12,73	-1027,26%

Tabela 18 – Tempos de execução das consultas em (s) para 100% das trajetórias

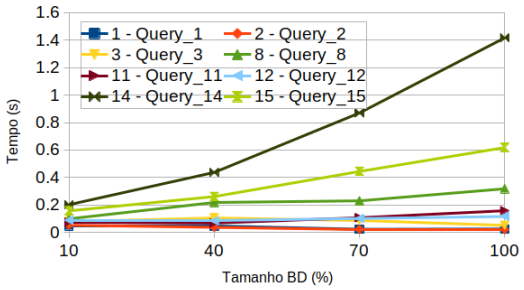
Query #	MASTERMobilityDB	Secondo	Diferença	Percentual
Query 1	0,04	0,03	0,01	29,08%
Query 2	0,02	0,02	0,00	-30,13%
Query 3	1,37	0,05	1,32	96,28%
Query 4	0,62	16,24	-15,62	-2525,59%
Query 5	0,85	20,70	-19,85	-2333,63%
Query 6	14,96	2.549,96	-2.535,00	-16950,54%
Query 7	3,93	17,30	-13,36	-339,91%
Query 8	0,15	0,32	-0,17	-110,96%
Query 9	65,43	1.692,15	-1.626,72	-2486,28%
Query 10	0,40	1.502,05	-1.501,66	-379564,18%
Query 11	0,18	0,16	0,02	12,09%
Query 12	0,03	0,12	-0,09	-288,52%
Query 13	0,52	12,84	-12,33	-2389,77%
Query 14	0,93	1,42	-0,49	-52,73%
Query 15	0,29	0,62	-0,33	-111,52%
Query 16	0,46	38,68	-38,22	-8375,88%
Query 17	1,34	18,51	-17,18	-1284,15%

sentação da informação semântica como um rótulo. Em vez disso, as informações semânticas são modeladas em termos de aspectos que podem ter um esquema arbitrário definido por um aspecto e seus atributos. Outra limitação é que uma trajetória simbólica associa apenas uma informação semântica por vez para uma trajetória. Por exemplo, se considerarmos locais de interesse e dias da semana como informação semântica, devem ser criadas duas trajetórias simbólicas, uma para o local e outra para o dia da semana, cada uma representada como um atributo em uma relação. Para uma consulta que envolve múltiplos aspectos no Secondo, todos os intervalos de tempo devem ser pesquisados para verificar onde todos os aspectos se mantêm juntos, enquanto o MASTERMobilityDB é capaz de acessar diretamente qualquer aspecto e seus atributos, seja para os objetos em movimento, os pontos, e para a MAT. Essas limitações do modelo de dados do BD Secondo na prática, apresentam resultados de desempenho piores que o MASTERMobilityDB.

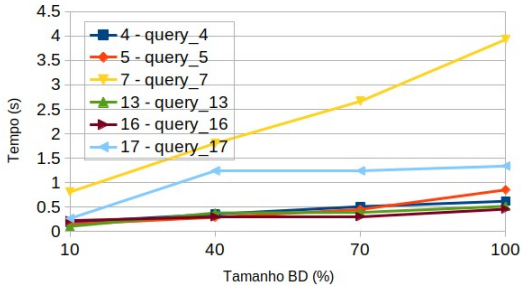
Tais resultados mostram que o MASTERMobilityDB superou o BD Secondo principalmente quando se aumentou o tamanho da amostra, onde o último também apresentou muitas instabilidades durante a execução das consultas, como falhas de alocação de memória, vazamentos de memória e geração de dumps em arquivo core do S.O. Linux. O benefício dos índices especializados, tabelas temporárias e o particionamento foram fundamentais para estes resultados. Com isto conclui-se que o MASTERMobilityDB é mais apropriado para tratar com grandes volumes de MATs,



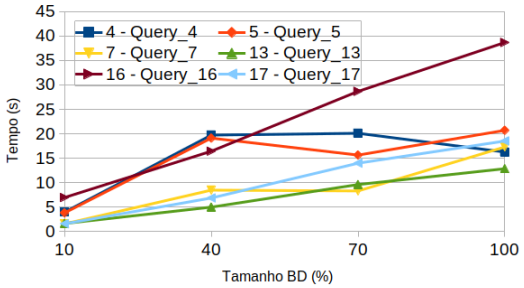
(a) MASTERMobilityDB



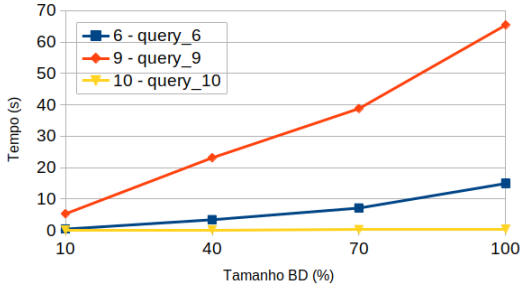
(b) Secondo



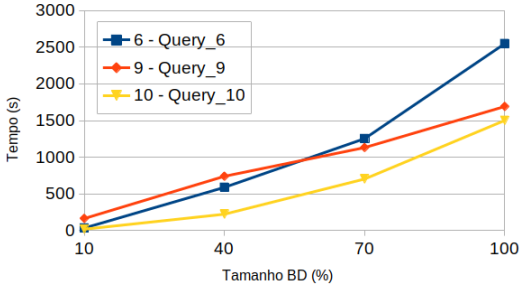
(c) MASTERMobilityDB



(d) Secondo



(e) MASTERMobilityDB



(f) Secondo

Figura 15 – Experimentos BerlinMOD

visto que este se mostrou mais estável e com desempenho visivelmente superior. Destaca-se a favor do Secondo os tempos para a carga dos conjuntos de dados e geração das MATs uma vez que este processo pode ser feito ignorando-se o log de transações do BD.

6 CONCLUSÃO E TRABALHOS FUTUROS

6.1 CONCLUSÃO

O gerenciamento e a análise de dados de mobilidade surgiram na última década como um domínio de pesquisa muito ativo abordando diversos aspectos como clusterização, integração, mineração, indexação, persistência e privacidade. Enquanto pesquisas anteriores se concentravam no processamento de trajetórias brutas coletadas de sensores, dispositivos GPS e similares, pesquisas recentes se concentram em métodos para enriquecer uma trajetória com informações mais contextualizadas e orientadas para aplicações. A adição de semântica aos dados de movimento trouxe um enorme potencial para aplicativos inovadores que permitem análises de fenômenos relacionados à mobilidade. Por exemplo, compreender por que e como as pessoas e os animais se deslocam, que lugares visitam e para quais propósitos, quais são suas atividades e quais recursos usam, é de extrema importância para a tomada de decisão, em particular para as autoridades públicas responsáveis pela gestão da sociedade.

Dentro deste contexto este trabalho endereça uma questão importante que é a persistência de MATs, cujo objetivo é armazenar informações semânticas sobre mobilidade, onde a representação e a consulta das MATs são consideradas a partir de uma perspectiva de banco de dados. No desenvolvimento deste trabalho primeiramente foram introduzidos os conceitos e definições que permitem uma compreensão clara das MATs. Estendeu-se esta tarefa inicial para abranger os comportamentos de objetos em movimento, bem como a contextualização dos mesmos dentro da Big Data. O modelo MASTER e as suas extensões para regras de dependência e sumarização de trajetórias foram revisitados, enfatizando a flexibilidade de representação dos mesmos. A pesquisa bibliográfica realizada foi de vital importância para o entendimento do estado atual da pesquisa em bancos de dados para trajetórias e fundamental no apoio à escolha do MobilityDB. O projeto *MASTERMobilityDB* visa cobrir todas as características do modelo MASTER, através de uma implementação simples e eficiente que servirá para futuros projetos de pesquisa em MATs.

Consistindo em informações semânticas dependentes do tempo relacionadas ao movimento de uma entidade, as MATs e o modelo MASTER oferecem uma representação mais intuitiva e com economia de recursos, apropriada para muitos domínios de aplicação. As MATs devem ser consideradas como uma extensão do modelo para bases de dados de objetos móveis e trajetórias brutas implementado no BD MobilityDB. Além das operações herdadas do MobilityDB, Postgres e PostGis, foi desenvolvido um conjunto de tipos abstratos de dados e uma API escrita em PL/SQL que simplifica a interação com o modelo de representação MASTER oferecendo um conjunto de rotinas que tratam o acesso e a manipulação das MATs. Todos esses componentes encapsulados em uma extensão do banco de dados facilitando a sua instalação.

Conforme já mencionado anteriormente, além dos dados geométricos brutos, outros dados dependentes do tempo relacionados ao movimento são registrados e/ou derivados para a maioria das aplicações, que tendem a aumentar em volume ao longo do tempo. Tendo isto em mente além da persistência das MATs este trabalho abordou questões importantes relacionadas ao desempenho e a estabilidade da solução como o uso de índices especializados Gist e SP-Gist, o particionamento de relações aliado ao paralelismo do banco de dados e a divisão de consultas complexas em consultas menores através do uso de tabelas temporárias. Com estes recursos combinados foram obtidos resultados de desempenho muito relevantes através dos experimentos realizados.

Para fins de demonstração, foram fornecidos exemplos de aplicação que fornecem uma visão das capacidades das técnicas propostas. Além disso, avaliamos o MASTERMobilityDB em uma grande série de experimentos que revisam o desempenho do BD para MATs. As consultas foram executadas no conjuntos de dados BerlinMOD com dados sintéticos de viagens. Além disso, ilustramos as vantagens das MATs em comparação com o modelo de trajetórias simbólicas oferecido pelo BD Secondo, em relação ao desempenho e formulação de consultas.

6.2 TRABALHOS FUTUROS

O MASTERMobilityDB foi implementado de forma que seja uma camada de persistência de MATs que servirá de base para futuros projetos de pesquisa. Tendo sido superada esta etapa, são apontadas como direções futuras de pesquisa: a) a implementação do cálculo das trajetórias representativas (MAT-SG) que, conforme já foi dito, tratam o agrupamento e as semelhanças em coleções de MATs, b) a implementação das regras de dependência para MATs conforme especificado no MASTER-DR.

Um novo projeto de pesquisa baseado no MASTERMobilityDB trata de regras de integridade as quais serão muito úteis para melhorar a qualidade e corretude dos dados de MATs. Uma possibilidade para esta implementação é a criação uma camada de restrições de integridade na persistência do BD que compreende: a) a definição de classes relevantes padrões a serem respeitados nos dados de MATs, b) o projeto das classes de regras de integridade e implementação como operações do MASTERMobilityDB e c) avaliação da sobrecarga versus a melhoria da qualidade dos dados de MATs.

Dados de mobilidade possuem múltiplas aplicações onde destacam-se *bigdata analytics* e mineração de dados. Trabalhos futuros que atuem na criação de novas rotinas utilitárias no MASTERMobilityDB que auxiliam na aplicação de métodos, como clusterização e classificação - relacionados a tais aplicações, são de grande valor. Dentre estas rotinas utilitárias podemos citar a criação de ETLs a partir do modelo MASTER gerando *dataframes* e *datasets*.

REFERÊNCIAS

AGGARWAL, Charu. Managing and Mining Sensor Data. *In*: [*S.l.*: *s.n.*], jan. 2013. p. 1–8. ISBN 978-1-4614-6308-5.

ALVARES, Luis Otávio; BOGORNY, Vania; KUIJPERS, Bart; MACÊDO, José Antônio Fernandes de; MOELANS, Bart; VAISMAN, Alejandro A. A model for enriching trajectories with semantic geographical information. *In*: 15TH ACM International Symposium on Geographic Information Systems, ACM-GIS 2007, November 7-9, 2007, Seattle, Washington, USA, Proceedings. [*S.l.*]: ACM, 2007. P. 22.

BOGORNY, Vania *et al.* Weka-STPM: A Software Architecture and Prototype for Semantic Trajectory Data Mining and Visualization. **T. GIS**, v. 15, p. 227–248, 2011.

BOGORNY, Vania; RENSO, Chiara; AQUINO, Artur Ribeiro de; LUCCA SIQUEIRA, Fernando de; ALVARES, Luis Otávio. CONSTAnT - A Conceptual Data Model for Semantic Trajectories of Moving Objects. **Trans. GIS**, v. 18, n. 1, p. 66–88, 2014.

BRANDOLI, Bruno *et al.* From multiple aspect trajectories to predictive analysis: a case study on fishing vessels in the Northern Adriatic sea. **GeoInformatica**, mar. 2022.

BULBUL, Tanyel; TAYLOR, John; OLGUN, Guney. A Case Study of Embedding Real-time Infrastructure Sensor Data to BIM. *In*: CONSTRUCTION Research Congress 2014. [*S.l.*]: American Society of Civil Engineers, mai. 2014. p. 269–278.

COHN, Mike. **User stories applied: For agile software development**. [*S.l.*]: Addison-Wesley Professional, 2004.

DÜNTGEN, Christian; BEHR, Thomas; GÜTING, Ralf Hartmut. BerlinMOD: a benchmark for moving object databases. **VLDB J.**, v. 18, n. 6, p. 1335–1368, 2009.

FILETO, Renato *et al.* The Baquara2 Knowledge-based Framework for Semantic Enrichment and Analysis of Movement Data. **Data & Knowledge Engineering**, v. 98, p. 104–122, 2015.

FRANK, Eibe; HALL, Mark; HOLMES, Geoffrey; KIRKBY, Richard; PFAHRINGER, Bernhard; WITTEN, Ian; TRIGG, Len. Weka-A Machine Learning Workbench for Data Mining. *In*: DATA Mining and Knowledge Discovery Handbook. [*S.l.*]: Springer, jul. 2010. p. 1269–1277.

GÓMEZ, Leticia *et al.* Analytical queries on semantic trajectories using graph databases. **Transactions in GIS**, v. 23, p. 1078–1101, jul. 2019.

GÜTING, Ralf Hartmut. Moving Object. *In*: LIU, Ling; ÖZSU, M. Tamer (Ed.). **Encyclopedia of Database Systems, Second Edition**. [*S.l.*]: Springer, 2018.

GÜTING, Ralf Hartmut; BEHR, Thomas; DÜNTGEN, Christian. SECONDO: A Platform for Moving Objects Database Research and for Publishing and Integrating Research Implementations. **IEEE Data Eng. Bull.**, v. 33, n. 2, p. 56–63, 2010.

GÜTING, Ralf Hartmut; SCHNEIDER, Markus. **Moving Objects Databases**. [*S.l.*]: Morgan Kaufmann, 2005. ISBN 0-12-088799-1.

GÜTING, Ralf Hartmut; VALDÉS, Fabio; DAMIANI, Maria Luisa. Symbolic Trajectories. **ACM Trans. Spatial Algorithms Syst.**, v. 1, n. 2, 7:1–7:51, 2015.

KHALID, Muhammad; CRUZ, Christophe; GINHAC, Dominique. Exploiting Semantic Trajectories Using HMMs and BIM for Worker Safety in Dynamic Environments. *In*: 2018 International Conference on Computational Science and Computational Intelligence (CSCI). [*S.l.*]: IEEE, dez. 2018. p. 525–530.

KITCHENHAM, Barbara. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. 2004, p. 1–26, 2004.

LAGO MACHADO, Vanessa; MELLO, Ronaldo; BOGORNY, Vania. A Method for Summarizing Trajectories with Multiple Aspects. *In*: DATABASE and Expert Systems Applications. [*S.l.*]: Springer International Publishing, jul. 2022. ISBN 978-3-031-12422-8.

LEE, Jae-Gil; HAN, Jiawei; WHANG, Kyu-Young. Trajectory Clustering: A Partition-and-Group Framework. *In*: PROCEEDINGS of the 2007 ACM SIGMOD International Conference on Management of Data. Beijing, China: Association for Computing Machinery (ACM), 2007. (SIGMOD '07), p. 593–604.

LEW, Dong June; YOO, Kihyun; NAM, Kwang Woo. DeepVQL: Deep Video Queries on PostgreSQL. VLDB Endowment, v. 16, n. 12, p. 3910–3913, ago. 2023. ISSN 2150-8097.

LI, Huan *et al.* Vita: a versatile toolkit for generating indoor mobility data for real-world buildings. **VLDB Endowment**, v. 9, p. 1453–1456, set. 2016.

LIU, Feng *et al.* Characterizing activity sequences using Profile Hidden Markov Models. **Expert Systems with Applications**, v. 42, mar. 2015.

MERATNIA, Nirvana; BY, Rolf de. Spatiotemporal Compression Techniques for Moving Point Objects. *In*: ADVANCES in Database Technology - EDBT 2004. [*S.l.*]: Springer Berlin Heidelberg, mar. 2004. v. 2992, p. 765–782.

MIR, Usama; ABBASI, Ubaid; YANG, Yang; BHATTI, Zeeshan Ahmed; MIR, Talha. Spatial Big Data and Moving Objects: A Comprehensive Survey. **IEEE Access**, v. 6, p. 58835–58857, 2018.

NOGUEIRA, Tales *et al.* FrameSTEP: A Framework for Annotating Semantic Trajectories Based on Episodes. **Expert Systems with Applications**, v. 92, p. 533–545, 2018.

NOGUEIRA, Tales Paiva; MARTIN, Hervé. Querying semantic trajectory episodes. *In*: 4TH ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems, MobiGIS 2015. [*S.l.*]: ACM, 2015. p. 23–30.

NOUREDDINE, Hassan *et al.* A hierarchical indoor and outdoor model for semantic trajectories. **Transactions in GIS**, v. 26, p. 214–235, 2021.

PANAGIOTAKIS, Costas; PELEKIS, Nikos; KOPANAKIS, Ioannis; RAMASSO, Emmanuel; THEODORIDIS, Yannis. Segmentation and Sampling of Moving Object Trajectories Based on Representativeness. **IEEE Transactions on Knowledge and Data Engineering**, v. 24, n. 7, p. 1328–1343, 2012.

PELEKIS, Nikos; FRENTZOS, Elias; GIATRAKOS, Nikos; THEODORIDIS, Yannis. HERMES: A trajectory DB engine for mobility-centric applications. **Int. J. Knowl. Based Organ**, v. 5, p. 19–41, abr. 2015.

PELEKIS, Nikos; THEODORIDIS, Yannis. **Mobility data management and exploration**. [*S.l.*]: Springer, 2014.

PORTELA, Tarlis Tortelli; CARVALHO, Jonata Tyska; BOGORNY, Vania. HiPerMovelets: high-performance movelet extraction for trajectory classification. **International Journal of Geographical Information Science**, Taylor e Francis, v. 36, n. 5, p. 1012–1036, 2022. eprint: https://doi.org/10.1080/13658816.2021.2018593

RENSO, Chiara; SPACCAPIETRA, Stefano; ZIMÁNYI, Esteban. **Mobility Data: Modeling, Management, and Understanding**. Cambridge: Cambridge University Press, 2013.

SANTOS MELLO, Ronaldo dos; BOGORNY, Vania; ALVARES, Luis Otávio; SANTANA, Luiz Henrique Zambom; FERRERO, Carlos Andres; FROZZA, Angelo Augusto; SCHREINER, Geomar Andre; RENSO, Chiara. MASTER: A multiple aspect view on trajectories. **Trans. GIS**, v. 23, n. 4, p. 805–822, 2019.

SANTOS MELLO, Ronaldo dos; SCHREINER, Geomar Andre; ALCHINI, Cristian Alexandre; SANTOS, Gustavo Goncalves dos; BOGORNY, Vania; RENSO, Chiara. Dependency Rule Modeling for Multiple Aspects Trajectories. *In*: GHOSE, Aditya K.; HORKOFF, Jennifer; SOUZA, Vitor E. Silva; PARSONS, Jeffrey; EVERMANN, Joerg (Ed.). **Conceptual Modeling - 40th International Conference, ER 2021, Virtual Event, October 18-21, 2021, Proceedings**. [*S.l.*]: Springer, 2021. v. 13011. (Lecture Notes in Computer Science), p. 123–132.

SPACCAPIETRA, Stefano; PARENT, Christine; DAMIANI, Maria Luisa; MACÊDO, José Antônio Fernandes de; PORTO, Fábio; VANGENOT, Christelle. A conceptual view on trajectories. **Data Knowl. Eng.**, v. 65, n. 1, p. 126–146, 2008.

TAMILMANI, Rajesh *et al.* Modelling and Analysis of Semantically Enriched Simplified Trajectories Using Graph Databases. **Advances in Cartography and GIScience of the ICA**, v. 1, p. 1–8, 2019.

TAMILMANI, Rajesh; STEFANAKIS, Emmanuel. Enriched geometric simplification of linear features. **GEOMATICA**, v. 71, p. 3–19, mar. 2017.

TORRES, Yenier *et al.* Stop-and-move sequence expressions over semantic trajectories. **International Journal of Geographical Information Science**, v. 35, p. 1–26, jul. 2020.

WANNOUS, Rouaa; MALKI, Jamal; BOUJU, Alain; VINCENT, Cécile. Time Integration in Semantic Trajectories Using an Ontological Modelling Approach. *In*: NEW Trends in Databases and Information Systems. [*S.l.*]: Springer Berlin Heidelberg, jan. 2013. v. 185, p. 187–198. ISBN 978-3-642-32518-2.

WIKIPÉDIA. **Diagrama de Voronoy — Wikipédia, a enciclopédia livre**. [*S.l.*: *s.n.*], 2022. http://pt.wikipedia.org/w/index.php?title=Diagrama_de_Voronoy&oldid=63124122. Accessed: 2022-01-30.

XU, Jianqiu; LU, Hua; BAO, Zhifeng. A Query Optimizer for Range Queries over Multi-Attribute Trajectories. **ACM Trans. Intell. Syst. Technol.**, Association for Computing Machinery, New York, NY, USA, v. 14, n. 1, jan. 2023. ISSN 2157-6904.

ZHANG, Hengcai; LU, Feng; XU, Jianqiu. Modeling and Querying Moving Objects with Social Relationships. **ISPRS Int. J. Geo Inf.**, v. 5, n. 7, p. 121, 2016.

ZIMANYI, Esteban *et al.* MobilityDB: A Mobility Database Based on PostgreSQL and PostGIS. **ACM Transactions on Database Systems**, v. 45, p. 1–42, dez. 2020.

ZIMÁNYI, Esteban; SAKR, Mahmoud Attia; LESUISSE, Arthur; BAKLI, Mohamed S. MobilityDB: A Mainstream Moving Object Database System. *In*: AREF, Walid G.; BERTOLOTTO, Michela; BOUROS, Panagiotis; JENSEN, Christian S.; MAHMOOD, Ahmed R.; NØRVÅG, Kjetil; SACHARIDIS, Dimitris; SARWAT, Mohamed (Ed.). **Proceedings of the 16th International Symposium on Spatial and Temporal Databases, SSTD 2019, Vienna, Austria, August 19-21, 2019**. [*S.l.*]: ACM, 2019. p. 206–209.

APÊNDICE A – LISTAGEM DAS OPERAÇÕES E FUNÇÕES DO MASTERMOBILITYDB

Tabela 19 – Listagem das operações implementadas no MASTERMobilityDB

Schema	Name	Return	Comments
master	aspect_attribute_count()	integer	Counts ocurrences on ASPECT_ATTRIBUTE
master	aspect_attribute_create(INOUT p_aspect_attribute aspect_attribute_typ)		Creates one tuple on ASPECT_ATTRIBUTE from a ASPECT_ATTRIBUTE_TYP object
master	aspect_attribute_create_many(INOUT p_aspect_attribute_a aspect_attribute_typ[])		Creates tuples on ASPECT_ATTRIBUTE from an array of ASPECT_ATTRIBUTE_TYP objects
master	aspect_attribute_delete(IN p_aspect_attribute_a aspect_attribute_typ[])		Delete ocurrences on ASPECT_ATTRIBUTE based on array of ASPECT_ATTRIBUTE_TYP objects
master	aspect_attribute_delete_all()		Delete all ocurrences on ASPECT_ATTRIBUTE
master	aspect_attribute_find_all()	SETOF aspect_attribute_typ	Find all ocurrences on ASPECT_ATTRIBUTE
master	aspect_attribute_find_by_id(p_aspect_id integer, p_attribute_id integer)	SETOF aspect_attribute_typ	Find all ocurrences on ASPECT_ATTRIBUTE based on primary key
master	aspect_attribute_update(IN p_aspect_attribute_a aspect_attribute_typ[])		Updates all ocurrences on ASPECT_ATTRIBUTE based on array of ASPECT_ATTRIBUTE_TYP objects
master	aspect_count()	integer	Counts ocurrences on ASPECT
master	aspect_create(INOUT p_aspect aspect_typ)		Creates one tuple on ASPECT from a ASPECT_TYP object
master	aspect_create_many(INOUT p_aspect_a aspect_typ[])		Creates tuples on ASPECT from an array of ASPECT_TYP objects
master	aspect_delete(IN p_aspect_a aspect_typ[])		Delete ocurrences on ASPECT based on array of ASPECT_TYP objects
master	aspect_delete_all()		Delete all ocurrences on ASPECT
master	aspect_find_all()	SETOF aspect_typ	Find all ocurrences on ASPECT
master	aspect_find_by_attribute(p_aspect_type text, p_attribute text, p_value text)	SETOF aspect_typ	Find all ocurrences on ASPECT based on aspect type and attribute value
master	aspect_find_by_id(p_aspect_id integer)	SETOF aspect_typ	Find all ocurrences on ASPECT based on primary key
master	aspect_find_by_name(p_description character varying)	SETOF aspect_typ	Find all ocurrences on ASPECT based on name or description
master	aspect_type_count()	integer	Counts ocurrences on ASPECT_TYPE
master	aspect_type_create(INOUT p_aspect_type aspect_type_typ)		Creates one tuple on ASPECT_TYPE from a ASPECT_TYPE_TYP object
master	aspect_type_create_many(INOUT p_aspect_type_a aspect_type_typ[])		Creates tuples on ASPECT_TYPE from an array of ASPECT_TYPE_TYP objects
master	aspect_type_delete(IN p_aspect_type_a aspect_type_typ[])		Delete ocurrences on ASPECT_TYPE based on array of ASPECT_TYPE_TYP objects
master	aspect_type_delete_all()		Delete all ocurrences on ASPECT_TYPE

Schema	Name	Return	Comments
master	aspect_type_find_all()	SETOF aspect_type_typ	Find all ocurrences on ASPECT_TYPE
master	aspect_type_find_by_id(p_aspect_type_id integer)	SETOF aspect_type_typ	Find all ocurrences on ASPECT_TYPE based on primary key
master	aspect_type_find_by_name(p_description character varying)	SETOF aspect_type_typ	Find all ocurrences on ASPECT_TYPE based on name or description
master	aspect_type_update(IN p_aspect_type_a aspect_type_typ[])		Updates all ocurrences on ASPECT_TYPE based on array of ASPECT_TYPE_TYP objects
master	aspect_update(IN p_aspect_a aspect_typ[])		Updates all ocurrences on ASPECT based on array of ASPECT_TYP objects
master	attribute_count()	integer	Counts ocurrences on ATTRIBUTE
master	attribute_create(INOUT p_attribute attribute_typ)		Creates one tuple on ATTRIBUTE from a ATTRIBUTE_TYP object
master	attribute_create_many(INOUT p_attribute_a attribute_typ[])		Creates tuples on ATTRIBUTE from an array of ATTRIBUTE_TYP objects
master	attribute_delete(IN p_attribute_a attribute_typ[])		Delete ocurrences on ATTRIBUTE based on array of ATTRIBUTE_TYP objects
master	attribute_delete_all()		Delete all ocurrences on ATTRIBUTE
master	attribute_find_all()	SETOF attribute_typ	Find all ocurrences on ATTRIBUTE
master	attribute_find_by_id(p_attribute_id integer)	SETOF attribute_typ	Find all ocurrences on ATTRIBUTE based on primary key
master	attribute_find_by_name(p_name character varying)	SETOF attribute_typ	Find all ocurrences on ATTRIBUTE based on name or description
master	attribute_update(IN p_attribute_a attribute_typ[])		Updates all ocurrences on ATTRIBUTE based on array of ATTRIBUTE_TYP objects
master	data_type_count()	integer	Counts ocurrences on DATA_TYPE
master	data_type_create(INOUT p_data_type data_type_typ)		Creates one tuple on DATA_TYPE from a DATA_TYPE_TYP object
master	data_type_create_many(INOUT p_data_type_a data_type_typ[])		Creates tuples on DATA_TYPE from an array of DATA_TYPE_TYP objects
master	data_type_delete(IN p_data_type_a data_type_typ[])		Delete ocurrences on DATA_TYPE based on array of DATA_TYPE_TYP objects
master	data_type_delete_all()		Delete all ocurrences on DATA_TYPE
master	data_type_find_all()	SETOF data_type_typ	Find all ocurrences on DATA_TYPE
master	data_type_find_by_id(p_data_type_id integer)	SETOF data_type_typ	Find all ocurrences on DATA_TYPE based on primary key
master	data_type_find_by_name(p_data_type_name character varying)	SETOF data_type_typ	Find all ocurrences on DATA_TYPE based on name or description
master	data_type_update(IN p_data_type_a data_type_typ[])		Updates all ocurrences on DATA_TYPE based on array of DATA_TYPE_TYP objects
master_dr	dataset_count()	integer	Counts ocurrences on DATASET
master_dr	dataset_create(INOUT p_dataset dataset_typ)		Creates one tuple on DATASET from a DATASET_TYP object
master_dr	dataset_create_many(INOUT p_dataset_a dataset_typ[])		Creates tuples on DATASET from an array of DATASET_TYP objects
master_dr	dataset_delete(IN p_dataset_a dataset_typ[])		Delete ocurrences on DATASET based on array of DATASET_TYP objects
master_dr	dataset_delete_all()		Delete all ocurrences on DATASET

Schema	Name	Return	Comments
master_dr	dataset_find_all()	SETOF dataset_typ	Find all ocurrences on DATASET
master_dr	dataset_find_by_id(p_dataset_id integer)	SETOF dataset_typ	Find all ocurrences on DATASET based on primary key
master_dr	dataset_update(IN p_dataset_a dataset_typ[])		Updates all ocurrences on DATASET based on array of DATASET_TYP objects
master_dr	dependency_rule_count()	integer	Counts ocurrences on DEPENDENCY_RULE
master_dr	dependency_rule_create(INOUT p_dependency_rule dependency_rule_typ)		Creates one tuple on DEPENDENCY_RULE from a DEPENDENCY_RULE_TYP object
master_dr	dependency_rule_create_many(INOUT p_dependency_rule_a dependency_rule_typ[])		Creates tuples on DEPENDENCY_RULE from an array of DEPENDENCY_RULE_TYP objects
master_dr	dependency_rule_dataset_count()	integer	Counts ocurrences on DEPENDENCY_RULE_DATASET
master_dr	dependency_rule_dataset_create(INOUT p_dependency_rule_dataset dependency_rule_dataset_typ)		Creates one tuple on DEPENDENCY_RULE_DATASET from a DEPENDENCY_RULE_DATASET_TYP object
master_dr	dependency_rule_dataset_create_many(INOUT p_dependency_rule_dataset_a dependency_rule_dataset_typ[])		Creates tuples on DEPENDENCY_RULE_DATASET from an array of DEPENDENCY_RULE_DATASET_TYP objects
master_dr	dependency_rule_dataset_delete(IN p_dependency_rule_dataset_a dependency_rule_dataset_typ[])		Delete ocurrences on DEPENDENCY_RULE_DATASET based on array of DEPENDENCY_RULE_DATASET_TYP objects
master_dr	dependency_rule_dataset_delete_all()		Delete all ocurrences on DEPENDENCY_RULE_DATASET
master_dr	dependency_rule_dataset_find_all()	SETOF dependency_rule_dataset_typ	Find all ocurrences on DEPENDENCY_RULE_DATASET
master_dr	dependency_rule_dataset_find_by_id(p_rule_id integer, p_dataset_id integer)	SETOF dependency_rule_dataset_typ	Find all ocurrences on DEPENDENCY_RULE_DATASET based on primary key
master_dr	dependency_rule_dataset_update(IN p_dependency_rule_dataset_a dependency_rule_dataset_typ[])		Updates all ocurrences on DEPENDENCY_RULE_DATASET based on array of DEPENDENCY_RULE_DATASET_TYP objects
master_dr	dependency_rule_delete(IN p_dependency_rule_a dependency_rule_typ[])		Delete ocurrences on DEPENDENCY_RULE based on array of DEPENDENCY_RULE_TYP objects
master_dr	dependency_rule_delete_all()		Delete all ocurrences on DEPENDENCY_RULE
master_dr	dependency_rule_find_all()	SETOF dependency_rule_typ	Find all ocurrences on DEPENDENCY_RULE
master_dr	dependency_rule_find_by_id(p_rule_id integer)	SETOF dependency_rule_typ	Find all ocurrences on DEPENDENCY_RULE based on primary key
master_dr	dependency_rule_update(IN p_dependency_rule_a dependency_rule_typ[])		Updates all ocurrences on DEPENDENCY_RULE based on array of DEPENDENCY_RULE_TYP objects
master	mat_aspect_count()	integer	Counts ocurrences on MAT_ASPECT

Schema	Name	Return	Comments
master	mat_aspect_create(INOUT p_mat_aspect mat_aspect_typ)		Creates one tuple on MAT_ASPECT from a MAT_ASPECT_TYP object
master	mat_aspect_create_many(INOUT p_mat_aspect_a mat_aspect_typ[])		Creates tuples on MAT_ASPECT from an array of MAT_ASPECT_TYP objects
master	mat_aspect_delete(IN p_mat_aspect_a mat_aspect_typ[])		Delete ocurrences on MAT_ASPECT based on array of MAT_ASPECT_TYP objects
master	mat_aspect_delete_all()		Delete all ocurrences on MAT_ASPECT
master	mat_aspect_find_all()	SETOF mat_aspect_typ	Find all ocurrences on MAT_ASPECT
master	mat_aspect_find_by_id(p_mat_id integer, p_aspect_id integer)	SETOF mat_aspect_typ	Find all ocurrences on MAT_ASPECT based on primary key
master	mat_aspect_update(IN p_mat_aspect_a mat_aspect_typ[])		Updates all ocurrences on MAT_ASPECT based on array of MAT_ASPECT_TYP objects
master	mat_count()	integer	Counts ocurrences on MAT
master	mat_create(INOUT p_mat mat_typ)		Creates one tuple on MAT from a MAT_TYP object
master	mat_create_many(INOUT p_mat_a mat_typ[])		Creates tuples on MAT from an array of MAT_TYP objects
master	mat_delete(IN p_mat_a mat_typ[])		Delete ocurrences on MAT based on array of MAT_TYP objects
master	mat_delete_all()		Delete all ocurrences on MAT
master_dr	mat_dr_count()	integer	Counts ocurrences on MAT_DR
master_dr	mat_dr_create(INOUT p_mat_dr mat_dr_typ)		Creates one tuple on MAT_DR from a MAT_DR_TYP object
master_dr	mat_dr_create_many(INOUT p_mat_dr_a mat_dr_typ[])		Creates tuples on MAT_DR from an array of MAT_DR_TYP objects
master_dr	mat_dr_delete(IN p_mat_dr_a mat_dr_typ[])		Delete ocurrences on MAT_DR based on array of MAT_DR_TYP objects
master_dr	mat_dr_delete_all()		Delete all ocurrences on MAT_DR
master_dr	mat_dr_find_all()	SETOF mat_dr_typ	Find all ocurrences on MAT_DR
master_dr	mat_dr_find_by_id(p_rule_id integer)	SETOF mat_dr_typ	Find all ocurrences on MAT_DR based on primary key
master_dr	mat_dr_update(IN p_mat_dr_a mat_dr_typ[])		Updates all ocurrences on MAT_DR based on array of MAT_DR_TYP objects
master	mat_find_all()	SETOF mat_typ	Find all ocurrences on MAT
master	mat_find_by_attribute(p_aspect_type text, p_attribute text, p_value text)	SETOF mat_typ	Find all ocurrences on MAT based on aspect type and attribute value
master	mat_find_by_id(p_mat_id integer)	SETOF mat_typ	Find all ocurrences on MAT based on primary key
master	mat_find_by_mo_id(p_mo_id integer)	SETOF mat_typ	Find all ocurrences on MAT based on MO_ID
master	mat_find_by_name(p_description character varying)	SETOF mat_typ	Find all ocurrences on MAT based on name or description
master_dr	mat_mat_dr_count()	integer	Counts ocurrences on MAT_MAT_DR
master_dr	mat_mat_dr_create(INOUT p_mat_mat_dr mat_mat_dr_typ)		Creates one tuple on MAT_MAT_DR from a MAT_MAT_DR_TYP object

Schema	Name	Return	Comments
master_dr	mat_mat_dr_create_many(INOUT p_mat_mat_dr_a mat_mat_dr_typ[])		Creates tuples on MAT_MAT_DR from an array of MAT_MAT_DR_TYP objects
master_dr	mat_mat_dr_delete(IN p_mat_mat_dr_a mat_mat_dr_typ[])		Delete ocurrences on MAT_MAT_DR based on array of MAT_MAT_DR_TYP objects
master_dr	mat_mat_dr_delete_all()		Delete all ocurrences on MAT_MAT_DR
master_dr	mat_mat_dr_find_all()	SETOF mat_mat_dr_typ	Find all ocurrences on MAT_MAT_DR
master_dr	mat_mat_dr_find_by_id(p_mat_id integer, p_rule_id integer)	SETOF mat_mat_dr_typ	Find all ocurrences on MAT_MAT_DR based on primary key
master_dr	mat_mat_dr_update(IN p_mat_mat_dr_a mat_mat_dr_typ[])		Updates all ocurrences on MAT_MAT_DR based on array of MAT_MAT_DR_TYP objects
master	mat_update(IN p_mat_a mat_typ[])		Updates all ocurrences on MAT based on array of MAT_TYP objects
master	mo_aspect_count()	integer	Counts ocurrences on MO_ASPECT
master	mo_aspect_create(INOUT p_mo_aspect mo_aspect_typ)		Creates one tuple on MO_ASPECT from a MO_ASPECT_TYP object
master	mo_aspect_create_many(INOUT p_mo_aspect_a mo_aspect_typ[])		Creates tuples on MO_ASPECT from an array of MO_ASPECT_TYP objects
master	mo_aspect_delete(IN p_mo_aspect_a mo_aspect_typ[])		Delete ocurrences on MO_ASPECT based on array of MO_ASPECT_TYP objects
master	mo_aspect_delete_all()		Delete all ocurrences on MO_ASPECT
master	mo_aspect_find_all()	SETOF mo_aspect_typ	Find all ocurrences on MO_ASPECT
master	mo_aspect_find_by_id(p_mo_id integer, p_aspect_id integer)	SETOF mo_aspect_typ	Find all ocurrences on MO_ASPECT based on primary key
master	mo_aspect_update(IN p_mo_aspect_a mo_aspect_typ[])		Updates all ocurrences on MO_ASPECT based on array of MO_ASPECT_TYP objects
master_dr	mo_dr_count()	integer	Counts ocurrences on MO_DR
master_dr	mo_dr_create(INOUT p_mo_dr mo_dr_typ)		Creates one tuple on MO_DR from a MO_DR_TYP object
master_dr	mo_dr_create_many(INOUT p_mo_dr_a mo_dr_typ[])		Creates tuples on MO_DR from an array of MO_DR_TYP objects
master_dr	mo_dr_delete(IN p_mo_dr_a mo_dr_typ[])		Delete ocurrences on MO_DR based on array of MO_DR_TYP objects
master_dr	mo_dr_delete_all()		Delete all ocurrences on MO_DR
master_dr	mo_dr_find_all()	SETOF mo_dr_typ	Find all ocurrences on MO_DR
master_dr	mo_dr_find_by_id(p_rule_id integer)	SETOF mo_dr_typ	Find all ocurrences on MO_DR based on primary key
master_dr	mo_dr_update(IN p_mo_dr_a mo_dr_typ[])		Updates all ocurrences on MO_DR based on array of MO_DR_TYP objects
master	mo_relationship_count()	integer	Counts ocurrences on MO_RELATIONSHIP
master	mo_relationship_create(INOUT p_mo_relationship mo_relationship_typ)		Creates one tuple on MO_RELATIONSHIP from a MO_RELATIONSHIP_TYP object
master	mo_relationship_create_many(INOUT p_mo_relationship_a mo_relationship_typ[])		Creates tuples on MO_RELATIONSHIP from an array of MO_RELATIONSHIP_TYP objects

Schema	Name	Return	Comments
master	mo_relationship_delete(IN p_mo_relationship_a mo_relationship_typ[])		Delete ocurrences on MO_RELATIONSHIP based on array of MO_RELATIONSHIP_TYP objects
master	mo_relationship_delete_all()		Delete all ocurrences on MO_RELATIONSHIP
master	mo_relationship_find_all()	SETOF mo_relationship_typ	Find all ocurrences on MO_RELATIONSHIP
master	mo_relationship_find_by_attribute(p_aspect_type text, p_attribute text, p_value text)	SETOF point_typ	Find all ocurrences on MO_RELATIONSHIP based on aspect type and attribute value
master	mo_relationship_find_by_id(p_mor_id integer)	SETOF mo_relationship_typ	Find all ocurrences on MO_RELATIONSHIP based on primary key
master	mo_relationship_find_by_name(p_description character varying)	SETOF mo_relationship_typ	Find all ocurrences on MO_RELATIONSHIP based on name or description
master_dr	mo_relationship_mor_dr_count()	integer	Counts ocurrences on MO_RELATIONSHIP_MOR_DR
master_dr	mo_relationship_mor_dr_create(INOUT p_mo_relationship_mor_dr mo_relationship_mor_dr_typ)		Creates one tuple on MO_RELATIONSHIP_MOR_DR from a MO_RELATIONSHIP_MOR_DR_TYP object
master_dr	mo_relationship_mor_dr_create_many(INOUT p_mo_relationship_mor_dr_a mo_relationship_mor_dr_typ[])		Creates tuples on MO_RELATIONSHIP_MOR_DR from an array of MO_RELATIONSHIP_MOR_DR_TYP objects
master_dr	mo_relationship_mor_dr_delete(IN p_mo_relationship_mor_dr_a mo_relationship_mor_dr_typ[])		Delete ocurrences on MO_RELATIONSHIP_MOR_DR based on array of MO_RELATIONSHIP_MOR_DR_TYP objects
master_dr	mo_relationship_mor_dr_delete_all()		Delete all ocurrences on MO_RELATIONSHIP_MOR_DR
master_dr	mo_relationship_mor_dr_find_all()	SETOF mo_relationship_mor_dr_typ	Find all ocurrences on MO_RELATIONSHIP_MOR_DR
master_dr	mo_relationship_mor_dr_find_by_id(p_mor_id integer, p_rule_id integer)	SETOF mo_relationship_mor_dr_typ	Find all ocurrences on MO_RELATIONSHIP_MOR_DR based on primary key
master_dr	mo_relationship_mor_dr_update(IN p_mo_relationship_mor_dr_a mo_relationship_mor_dr_typ[])		Updates all ocurrences on MO_RELATIONSHIP_MOR_DR based on array of MO_RELATIONSHIP_MOR_DR_TYP objects
master	mo_relationship_update(IN p_mo_relationship_a mo_relationship_typ[])		Updates all ocurrences on MO_RELATIONSHIP based on array of MO_RELATIONSHIP_TYP objects
master	mo_type_count()	integer	Counts ocurrences on MO_TYPE
master	mo_type_create(INOUT p_mo_type mo_type_typ)		Creates one tuple on MO_TYPE from a MO_TYPE_TYP object
master	mo_type_create_many(INOUT p_mo_type_a mo_type_typ[])		Creates tuples on MO_TYPE from an array of MO_TYPE_TYP objects
master	mo_type_delete(IN p_mo_type_a mo_type_typ[])		Delete ocurrences on MO_TYPE based on array of MO_TYPE_TYP objects
master	mo_type_delete_all()		Delete all ocurrences on MO_TYPE
master	mo_type_find_all()	SETOF mo_type_typ	Find all ocurrences on MO_TYPE
master	mo_type_find_by_id(p_mo_type_id integer)	SETOF mo_type_typ	Find all ocurrences on MO_TYPE based on primary key

Schema	Name	Return	Comments
master	mo_type_find_by_name(p_description character varying)	SETOF mo_type_typ	Find all ocurrences on MO_TYPE based on name or description
master	mo_type_update(IN p_mo_type_a mo_type_typ[])		Updates all ocurrences on MO_TYPE based on array of MO_TYPE_TYP objects
master	mor_aspect_count()	integer	Counts ocurrences on MOR_ASPECT
master	mor_aspect_create(INOUT p_mor_aspect mor_aspect_typ)		Creates one tuple on MOR_ASPECT from a MOR_ASPECT_TYP object
master	mor_aspect_create_many(INOUT p_mor_aspect_a mor_aspect_typ[])		Creates tuples on MOR_ASPECT from an array of MOR_ASPECT_TYP objects
master	mor_aspect_delete(IN p_mor_aspect_a mor_aspect_typ[])		Delete ocurrences on MOR_ASPECT based on array of MOR_ASPECT_TYP objects
master	mor_aspect_delete_all()		Delete all ocurrences on MOR_ASPECT
master	mor_aspect_find_all()	SETOF mor_aspect_typ	Find all ocurrences on MOR_ASPECT
master	mor_aspect_find_by_id(p_mor_id integer, p_aspect_id integer)	SETOF mor_aspect_typ	Find all ocurrences on MOR_ASPECT based on primary key
master	mor_aspect_update(IN p_mor_aspect_a mor_aspect_typ[])		Updates all ocurrences on MOR_ASPECT based on array of MOR_ASPECT_TYP objects
master_dr	mor_dr_count()	integer	Counts ocurrences on MOR_DR
master_dr	mor_dr_create(INOUT p_mor_dr mor_dr_typ)		Creates one tuple on MOR_DR from a MOR_DR_TYP object
master_dr	mor_dr_create_many(INOUT p_mor_dr_a mor_dr_typ[])		Creates tuples on MOR_DR from an array of MOR_DR_TYP objects
master_dr	mor_dr_delete(IN p_mor_dr_a mor_dr_typ[])		Delete ocurrences on MOR_DR based on array of MOR_DR_TYP objects
master_dr	mor_dr_delete_all()		Delete all ocurrences on MOR_DR
master_dr	mor_dr_find_all()	SETOF mor_dr_typ	Find all ocurrences on MOR_DR
master_dr	mor_dr_find_by_id(p_rule_id integer)	SETOF mor_dr_typ	Find all ocurrences on MOR_DR based on primary key
master_dr	mor_dr_update(IN p_mor_dr_a mor_dr_typ[])		Updates all ocurrences on MOR_DR based on array of MOR_DR_TYP objects
master	moving_object_count()	integer	Counts ocurrences on MOVING_OBJECT
master	moving_object_create(INOUT p_moving_object moving_object_typ)		Creates one tuple on MOVING_OBJECT from a MOVING_OBJECT_TYP object
master	moving_object_create_many(INOUT p_moving_object_a moving_object_typ[])		Creates tuples on MOVING_OBJECT from an array of MOVING_OBJECT_TYP objects
master	moving_object_delete(IN p_moving_object_a moving_object_typ[])		Delete ocurrences on MOVING_OBJECT based on array of MOVING_OBJECT_TYP objects
master	moving_object_delete_all()		Delete all ocurrences on MOVING_OBJECT
master	moving_object_find_all()	SETOF moving_object_typ	Find all ocurrences on MOVING_OBJECT
master	moving_object_find_by_attribute(p_aspect_type text, p_attribute text, p_value text)	SETOF moving_object_typ	Find all ocurrences on MOVING_OBJECT based on aspect type and attribute value

Schema	Name	Return	Comments
master	moving_object_find_by_id(p_mo_id integer)	SETOF moving_object_typ	Find all ocurrences on MOVING_OBJECT based on primary key
master	moving_object_find_by_name(p_description character varying)	SETOF moving_object_typ	Find all ocurrences on MOVING_OBJECT based on name or description
master_dr	moving_object_mo_dr_count()	integer	Counts ocurrences on MOVING_OBJECT_MO_DR
master_dr	moving_object_mo_dr_create(INOUT p_moving_object_mo_dr moving_object_mo_dr_typ)		Creates one tuple on MOVING_OBJECT_MO_DR from a MOVING_OBJECT_MO_DR_TYP object
master_dr	moving_object_mo_dr_create_many(INOUT p_moving_object_mo_dr_a moving_object_mo_dr_typ[])		Creates tuples on MOVING_OBJECT_MO_DR from an array of MOVING_OBJECT_MO_DR_TYP objects
master_dr	moving_object_mo_dr_delete(IN p_moving_object_mo_dr_a moving_object_mo_dr_typ[])		Delete ocurrences on MOVING_OBJECT_MO_DR based on array of MOVING_OBJECT_MO_DR_TYP objects
master_dr	moving_object_mo_dr_delete_all()		Delete all ocurrences on MOVING_OBJECT_MO_DR
master_dr	moving_object_mo_dr_find_all()	SETOF moving_object_mo_dr_typ	Find all ocurrences on MOVING_OBJECT_MO_DR
master_dr	moving_object_mo_dr_find_by_id(p_mo_id integer, p_rule_id integer)	SETOF moving_object_mo_dr_typ	Find all ocurrences on MOVING_OBJECT_MO_DR based on primary key
master_dr	moving_object_mo_dr_update(IN p_moving_object_mo_dr_a moving_object_mo_dr_typ[])		Updates all ocurrences on MOVING_OBJECT_MO_DR based on array of MOVING_OBJECT_MO_DR_TYP objects
master	moving_object_update(IN p_moving_object_a moving_object_typ[])		Updates all ocurrences on MOVING_OBJECT based on array of MOVING_OBJECT_TYP objects
master	point_aspect_count()	integer	Counts ocurrences on POINT_ASPECT
master	point_aspect_create(INOUT p_point_aspect point_aspect_typ)		Creates one tuple on POINT_ASPECT from a POINT_ASPECT_TYP object
master	point_aspect_create_many(INOUT p_point_aspect_a point_aspect_typ[])		Creates tuples on POINT_ASPECT from an array of POINT_ASPECT_TYP objects
master	point_aspect_delete(IN p_point_aspect_a point_aspect_typ[])		Delete ocurrences on POINT_ASPECT based on array of POINT_ASPECT_TYP objects
master	point_aspect_delete_all()		Delete all ocurrences on POINT_ASPECT
master	point_aspect_find_all()	SETOF point_aspect_typ	Find all ocurrences on POINT_ASPECT
master	point_aspect_find_by_id(p_point_id integer, p_aspect_id integer)	SETOF point_aspect_typ	Find all ocurrences on POINT_ASPECT based on primary key
master	point_aspect_update(IN p_point_aspect_a point_aspect_typ[])		Updates all ocurrences on POINT_ASPECT based on array of POINT_ASPECT_TYP objects
master	point_count()	integer	Counts ocurrences on POINT
master	point_create(INOUT p_point point_typ)		Creates one tuple on POINT from a POINT_TYP object

Schema	Name	Return	Comments
master	point_create_many(INOUT p_point_a point_typ[])		Creates tuples on POINT from an array of POINT_TYP objects
master	point_delete_all()		Delete all ocurrences on POINT
master_dr	point_dr_count()	integer	Counts ocurrences on POINT_DR
master_dr	point_dr_create(INOUT p_point_dr point_dr_typ)		Creates one tuple on POINT_DR from a POINT_DR_TYP object
master_dr	point_dr_create_many(INOUT p_point_dr_a point_dr_typ[])		Creates tuples on POINT_DR from an array of POINT_DR_TYP objects
master_dr	point_dr_delete(IN p_point_dr_a point_dr_typ[])		Delete ocurrences on POINT_DR based on array of POINT_DR_TYP objects
master_dr	point_dr_delete_all()		Delete all ocurrences on POINT_DR
master_dr	point_dr_find_all()	SETOF point_dr_typ	Find all ocurrences on POINT_DR
master_dr	point_dr_find_by_id(p_rule_id integer)	SETOF point_dr_typ	Find all ocurrences on POINT_DR based on primary key
master_dr	point_dr_update(IN p_point_dr_a point_dr_typ[])		Updates all ocurrences on POINT_DR based on array of POINT_DR_TYP objects
master	point_find_all()	SETOF point_typ	Find all ocurrences on POINT
master	point_find_by_attribute(p_aspect_type text, p_attribute text, p_value text)	SETOF point_typ	Find all ocurrences on POINT based on aspect type and attribute value
master	point_find_by_id(p_point_id integer)	SETOF point_typ	Find all ocurrences on POINT based on primary key
master	point_update(IN p_point_a point_typ[])		Updates all ocurrences on PREDICATE_ASPECT_TYPE based on array of POINT_TYP objects
master_dr	point_point_dr_count()	integer	Counts ocurrences on POINT_POINT_DR
master_dr	point_point_dr_create(INOUT p_point_point_dr point_point_dr_typ)		Creates one tuple on POINT_POINT_DR from a POINT_POINT_DR_TYP object
master_dr	point_point_dr_create_many(INOUT p_point_point_dr_a point_point_dr_typ[])		Creates tuples on POINT_POINT_DR from an array of POINT_POINT_DR_TYP objects
master_dr	point_point_dr_delete(IN p_point_point_dr_a point_point_dr_typ[])		Delete ocurrences on POINT_POINT_DR based on array of POINT_POINT_DR_TYP objects
master_dr	point_point_dr_delete_all()		Delete all ocurrences on POINT_POINT_DR
master_dr	point_point_dr_find_all()	SETOF point_point_dr_typ	Find all ocurrences on POINT_POINT_DR
master_dr	point_point_dr_find_by_id(p_point_id integer, p_rule_id integer)	SETOF point_point_dr_typ	Find all ocurrences on POINT_POINT_DR based on primary key
master_dr	point_point_dr_update(IN p_point_point_dr_a point_point_dr_typ[])		Updates all ocurrences on POINT_POINT_DR based on array of POINT_POINT_DR_TYP objects
mat_sg	point_repr_point_count()	integer	Counts ocurrences on POINT_REPR_POINT
mat_sg	point_repr_point_create(INOUT p_point_repr_point point_repr_point_typ)		Creates one tuple on POINT_REPR_POINT from a POINT_REPR_POINT_TYP object
mat_sg	point_repr_point_create_many(INOUT p_point_repr_point_a point_repr_point_typ[])		Creates tuples on POINT_REPR_POINT from an array of POINT_REPR_POINT_TYP objects

Schema	Name	Return	Comments
mat_sg	point_repr_point_delete(IN p_point_repr_point_a point_repr_point_typ[])		Delete ocurrences on POINT_REPR_POINT based on array of POINT_REPR_POINT_TYP objects
mat_sg	point_repr_point_delete_all()		Delete all ocurrences on POINT_REPR_POINT
mat_sg	point_repr_point_find_all()	SETOF point_repr_point_typ	Find all ocurrences on POINT_REPR_POINT
mat_sg	point_repr_point_find_by_id(p_point_id integer, p_repr_point_id integer)	SETOF point_repr_point_typ	Find all ocurrences on POINT_REPR_POINT based on primary key
mat_sg	point_repr_point_update(IN p_point_repr_point_a point_repr_point_typ[])		Updates all ocurrences on POINT_REPR_POINT based on array of POINT_REPR_POINT_TYP objects
master_dr	predicate_aspect_count()	integer	Counts ocurrences on PREDICATE_ASPECT
master_dr	predicate_aspect_create(INOUT p_predicate_aspect predicate_aspect_typ)		Creates one tuple on PREDICATE_ASPECT from a PREDICATE_ASPECT_TYP object
master_dr	predicate_aspect_create_many(INOUT p_predicate_aspect_a predicate_aspect_typ[])		Creates tuples on PREDICATE_ASPECT from an array of PREDICATE_ASPECT_TYP objects
master_dr	predicate_aspect_delete(IN p_predicate_aspect_a predicate_aspect_typ[])		Delete ocurrences on PREDICATE_ASPECT based on array of PREDICATE_ASPECT_TYP objects
master_dr	predicate_aspect_delete_all()		Delete all ocurrences on PREDICATE_ASPECT
master_dr	predicate_aspect_find_all()	SETOF predicate_aspect_typ	Find all ocurrences on PREDICATE_ASPECT
master_dr	predicate_aspect_find_by_id(p_predicate_id integer, p_aspect_id integer)	SETOF predicate_aspect_typ	Find all ocurrences on PREDICATE_ASPECT based on primary key
master_dr	predicate_aspect_type_count()	integer	Counts ocurrences on PREDICATE_ASPECT_TYPE
master_dr	predicate_aspect_type_create(INOUT p_predicate_aspect_type predicate_aspect_type_typ)		Creates one tuple on PREDICATE_ASPECT_TYPE from a PREDICATE_ASPECT_TYPE_TYP object
master_dr	predicate_aspect_type_create_many(INOUT p_predicate_aspect_type_a predicate_aspect_type_typ[])		Creates tuples on PREDICATE_ASPECT_TYPE from an array of PREDICATE_ASPECT_TYPE_TYP objects
master_dr	predicate_aspect_type_delete(IN p_predicate_aspect_type_a predicate_aspect_type_typ[])		Delete ocurrences on PREDICATE_ASPECT_TYPE based on array of PREDICATE_ASPECT_TYPE_TYP objects
master_dr	predicate_aspect_type_delete_all()		Delete all ocurrences on PREDICATE_ASPECT_TYPE
master_dr	predicate_aspect_type_find_all()	SETOF predicate_aspect_type_typ	Find all ocurrences on PREDICATE_ASPECT_TYPE
master_dr	predicate_aspect_type_find_by_id(p_predicate_id integer, p_aspect_type_id integer)	SETOF predicate_aspect_type_typ	Find all ocurrences on PREDICATE_ASPECT_TYPE based on primary key

Schema	Name	Return	Comments
master_dr	predicate_aspect_type_update(IN p_predicate_aspect_type_a predicate_aspect_type_typ[])		Updates all ocurrences on PREDICATE_ASPECT_TYPE based on array of PREDICATE_ASPECT_TYPE_TYP objects
master_dr	predicate_aspect_update(IN p_predicate_aspect_a predicate_aspect_typ[])		Updates all ocurrences on PREDICATE_ASPECT based on array of PREDICATE_ASPECT_TYP objects
master_dr	predicate_count()	integer	Counts ocurrences on PREDICATE
master_dr	predicate_create(INOUT p_predicate predicate_typ)		Creates one tuple on PREDICATE from a PREDICATE_TYP object
master_dr	predicate_create_many(INOUT p_predicate_a predicate_typ[])		Creates tuples on PREDICATE from an array of PREDICATE_TYP objects
master_dr	predicate_delete(IN p_predicate_a predicate_typ[])		Delete ocurrences on PREDICATE based on array of PREDICATE_TYP objects
master_dr	predicate_delete_all()		Delete all ocurrences on PREDICATE
master_dr	predicate_find_all()	SETOF predicate_typ	Find all ocurrences on PREDICATE
master_dr	predicate_find_by_attribute(p_aspect_type text, p_attribute text, p_value text)	SETOF point_typ	Find all ocurrences on PREDICATE based on aspect type and attribute value
master_dr	predicate_find_by_id(p_predicate_id integer)	SETOF predicate_typ	Find all ocurrences on PREDICATE based on primary key
master_dr	predicate_update(IN p_predicate_a predicate_typ[])		Updates all ocurrences on PREDICATE based on array of PREDICATE_TYP objects
mat_sg	repr_point_count()	integer	Counts ocurrences on REPR_POINT
mat_sg	repr_point_create(INOUT p_repr_point repr_point_typ)		Creates one tuple on REPR_POINT from a REPR_POINT_TYP object
mat_sg	repr_point_create_many(INOUT p_repr_point_a repr_point_typ[])		Creates tuples on REPR_POINT from an array of REPR_POINT_TYP objects
mat_sg	repr_point_delete(IN p_repr_point_a repr_point_typ[])		Delete ocurrences on REPR_POINT based on array of REPR_POINT_TYP objects
mat_sg	repr_point_delete_all()		Delete all ocurrences on REPR_POINT
mat_sg	repr_point_find_all()	SETOF repr_point_typ	Find all ocurrences on REPR_POINT
mat_sg	repr_point_find_by_id(p_repr_point_id integer)	SETOF repr_point_typ	Find all ocurrences on REPR_POINT based on primary key
mat_sg	repr_point_update(IN p_repr_point_a repr_point_typ[])		Updates all ocurrences on REPR_POINT based on array of REPR_POINT_TYP objects
util	create_partitions_by_date(IN p_schemaname text, IN p_tablename text, IN p_startdate date, IN p_enddate date, IN p_columnname text, IN p_interval text, IN p_schemapart text DEFAULT 'partitions'::text)		Create the partitions for the specified table based on the selected interval

Schema	Name	Return	Comments
util	disable_fks(IN p_schema_name character varying, IN p_table_name character varying)		Disables all primary keys for the specified table
util	disable_indexes(IN p_schema_name character varying, IN p_table_name character varying)		Disables all indexes for the specified table
util	drop_partitions_by_date(IN p_schemaname text, IN p_tablename text, IN p_startdate date, IN p_enddate date, IN p_interval text, IN p_schemapart text DEFAULT 'partitions'::text)		Remove partitions for the specified table based on period and interval
util	duplicate_table(IN p_schemaname text, IN p_tablename text, IN p_targetschema text, IN p_targettable text, IN p_percenttuples integer, IN p_startdate date, IN p_enddate date, IN p_columnname text, IN p_interval text, IN p_schemapart text)		Duplicates a sample of rows of the specified table into a new table
util	enable_fks(IN p_schema_name character varying, IN p_table_name character varying)		Enable all foreign keys for the specified table
util	enable_indexes(IN p_schema_name character varying, IN p_table_name character varying)		Enable all indexes for the specified table
util	get_partition_by_date(p_schema_name text, p_table_name text, p_date date, p_schemapart text DEFAULT 'partitions'::text)	text	Returns the partition name for the specified table based on the date parameter
util	reset_sequence(IN p_schema_name character varying, IN p_table_name character varying)		Resets the sequence value for the specified table's primary key to the maximum value
util	reset_sequences(IN p_schema_name character varying)		Resets all sequences at the schema specified