INFORMATIK
BERICHTE

362 – 03/2012

GMOBench:
A Benchmark for Generic Moving Objects

Jianqiu Xu, Ralf Hartmut Güting

Fakultät für Mathematik und Informatik
Postfach 940
D-58084 Hagen

GMOBench: A Benchmark for Generic Moving Objects

Jianqiu Xu, Ralf Hartmut Güting

Database Systems for New Applications, Mathematics and Computer Science
FernUniversität Hagen, Germany
{jianqiu.xu,rhg}@fernuni-hagen.de

March 8, 2012

Abstract

In this paper, we introduce a benchmark called GMOBench that aims to evaluate the performance of a database system managing moving objects in different environments. In real life, people's movement can cover several environments rather than one, for example, *Indoor* $\rightarrow$ *Walk* $\rightarrow$ *Bus* $\rightarrow$ *Walk*. As a result, the complete trip needs to be managed by a database system in order to support novel queries. Since existing methods are limited to one environment, new technologies are developed in a database system that is able to manage generic moving objects. A meaningful analysis and evaluation of such a system necessitates a comprehensive benchmark. GMOBench is settled in a realistic scenario which comprises (1) a data generator with the capability of creating a scalable set of trips representing the complete movement of humans (both indoor and outdoor); (2) a set of carefully designed and new benchmark queries. The generator defines some parameters so that users can control the characteristics of results. We create the benchmark data in such a way that the dataset can mirror important characteristics and real world distributions of human mobility. Efficient data access methods are developed for query processing, and optimization techniques are proposed to improve the efficiency. We perform an extensive experimental study on comprehensive datasets to evaluate the performance of the system. The results demonstrate the effectiveness and efficiency of our approaches.

1 Introduction

Recently, the area of moving objects with different transportation modes has been an interesting topic due to novel applications on detecting outdoor transportation modes and advanced trip plannings or recommendations. Researchers try to infer motion modes such as walking, driving and cycling from raw GPS data in order to have more contextual information of a mobile user [41, 40, 26, 30]. In an advanced transportation system [6, 8], a realistic traveling plan includes a trip with different choices of modes and constraints with modes, e.g., less than two bus transfers. As a kind of human behavior, transportation modes are closely related to the movement. Recognizing such pieces of information can enrich the knowledge of a user's mobility. From the viewpoint of the database community, the management of continuously changing location data and transportation modes requires dedicated support from the underlying database system. However, existing techniques are only able to manage the data in one environment such as free space or road network, and cannot answer new queries on moving objects in multiple environments. The difficulty is to have a consistent and efficient location representation in all available environments such as road network, bus network and indoor. Based on a data model proposed in [36], we develop new techniques in a database system to manage moving objects that travel through different environments, for example, *Indoor* $\rightarrow$ *Walk* $\rightarrow$ *Bus* $\rightarrow$ *Walk*. To evaluate the performance of such a database system, a benchmark is needed to test the system under a wide range of queries regarding transportation modes and moving environments.

A benchmark [7, 11, 31, 13, 25, 29], consisting of a set of comprehensive and scalable datasets and a group of well defined queries, plays a crucial role in evaluating the functionality and performance of a database both for application users and developers. Additionally, a benchmark allows one to test a system's capabilities and helps determine its strengths or potential bottlenecks. In the field of moving objects databases, simulation is

widely accepted to provide synthetic data for designing and testing new data types and access methods due to the difficulty of getting a large amount of real data. In particular, real data might not be comprehensive enough to thoroughly evaluate the system in consideration. Consequently, researchers develop tools to create synthetic data in a realistic scenario. Nevertheless, existing data generators [33, 9, 28] and benchmarks [32, 12, 14] only deal with the movement in one environment without considering transportation modes. The data do not represent the complete trips of humans. It is also not easy to get the real data of moving objects with precise transportation modes including both outdoor and indoor. To solve the problem, new methods are needed to generate moving objects with multiple transportation modes in a realistic way. The previous works [36, 37, 38] are fundamental steps to the present benchmark where a data model is designed to represent generic moving objects by referencing to the underlying environments, and a data generator is developed to create all real world environments: road network, bus network, metro network, pavement areas and indoor.

In this paper, we propose a benchmark called GMOBench to evaluate the database performance by a broad range of novel queries on moving objects regarding transportation modes and mobile environments. We process the complete past movement (also called trajectory) and generate scalable and comprehensive datasets to simulate a variety of real life scenarios. It makes no sense to create human movement in a pure random fashion as usually there is an evident motivation to start a trip, accomplishing some tasks or performing an activity. To achieve the goal of realistic simulation, we define a set of rules to create trips with various properties to model human movement behavior in practice. For example, people's trajectories exhibit *regular patterns* [16] most of the time, e.g., commuting. On the weekend, based on the habits and preferences they may have some trips to interesting places such as home of friends, shopping malls, and cinemas. To perform an activity, people usually prefer nearby to distant places, e.g., look for the nearest hotel. We define three parameters location, time and transportation modes with the aim of letting benchmark users create the desired data by configuring different settings. Since moving objects are generated based on real life behavior, our benchmark datasets can be used for some other applications. For example, one can monitor the traffic condition (e.g., rush hour) by creating trips between home and work places during a certain time period. As these moving objects contain multiple transportation modes, one can analyze the data and test whether the traffic jam can be relieved by improving and adjusting the public transportation system. One can also investigate people's movement inside buildings by generating indoor moving objects.

Complementary to the benchmark data generator, we design a set of queries to test a variety of operator constellations and data access methods. Two groups of queries are proposed where one deals with the underlying environments and the other considers moving objects. Most of the queries are not supported by existing methods for the reason that previous work is limited to one environment. Optimization strategies are developed to improve the query efficiency. The transportation modes of a moving object can be represented and stored in a compact way, using an integer. Indices are built on moving objects to accelerate the procedure of accessing the data. We present a thorough experimental study for performance evaluation with the proposed queries under comprehensive datasets. The results demonstrate the effectiveness and efficiency of the proposed optimization techniques. The evaluation is performed in a database system SECONDO [17], as to our knowledge there is not yet any other available DBMS that can represent generic moving objects and execute the given set of queries.

The rest of the paper is organized as follows: A concise overview of related work is presented in Section 2. In Section 3, we elaborate the configuration of benchmark data and the generating algorithm. Benchmark queries are defined in Section 4 and optimization techniques are developed in Section 5. We perform the experiments in Section 6 and conclude the paper in Section 7.

2 Related Work

2.1 Benchmarking

A benchmark proposed in [35] deals with 3-dimensional spatio-temporal data that require significant temporal processing and storage capabilities, and has provisions for evaluating the ability of a spatio-temporal database to

handle 3-dimensional data. The work expands on the Sequoia 2000 and Paradise benchmarks, and is oriented towards general operating system and database system performance comparison. In the context of moving objects databases, [32] proposes a benchmark that includes a database description and a group of representative SQL-based queries. Ten benchmark queries plus two operations for loading and updating data are proposed. The authors give an ER diagram of a database for location-based services where the entities include humans, buildings and roads. Humans visit buildings (shops) for their interests and requests on products. Each road stores two kinds of data: (1) a polyline; (2) the time when people pass the road. But the paper does not present a method to create the benchmark database and there is no performance evaluation. BerlinMOD [14] is a benchmark that uses the SECONDO DBMS [17] for generating moving object data. A scenario is simulated where a number of cars move within the road network of Berlin and sampled positions from such movements are used as the data. The method models a person's trips to and from work during the daytime on workdays as well as some additional trips in the evening and on the weekend. Long-term observations of moving objects are available, e.g., a month. A set of carefully selected SQL-based queries constitute the workload. However, these benchmarks only process moving objects and queries in one environment and do not consider transportation modes. Compared with the aforementioned work, our benchmark is general in the context of moving objects where the system manages trips passing through different environments and supports new queries on these data.

Benchmarking moving objects indices is studied in [22, 20, 12], focusing on location update, current and near future positions. [22] presents a benchmark termed DynaMark for dynamic spatial indexing, that is towards location-based services. Three types of queries are defined that form the basis of location-based service applications: proximity queries, kNN queries and sorted distance queries. A benchmark called COST [20] is concerned with the indexing of current and near-future moving objects positions and aims to evaluate the index ability to accommodate uncertain object positions. Three types of queries are investigated, timeslice query, window query and moving window query. In [12], three types of datasets are generated: (1) uniform distribution; (2) Gaussian distribution and (3) road-network-based datasets. The goal is to measure the overall index efficiency and to simulate certain real-world scenarios. The query workloads consist of the range query and the kNN query. A set of aspects is proposed for the performance evaluation such as data size, update frequency and buffer size.

2.2 Spatio-Temporal Data Generators

The so far developed tools have been using random functions and road networks to model different physical aspects of moving objects. A network-based moving objects generator is proposed in [9, 10] for the traffic application. Objects are created in a random way and appear and disappear when their destinations are reached. Important concepts of the generators are the maximum speed, the maximum edge capacity, etc. Two kinds of methods are used for setting positions and velocities for moving objects generation [28], uniform distribution and skewed distribution. GSTD [33], a widely used spatio-temporal generator, defines a set of parameters to control the generated trajectories: (1) the duration of an object instance; (2) the shift of objects and (3) the resizing of objects. Initialized by a certain distribution of points or rectangle objects, GSTD computes at every time step the next position and the shape of objects based on parametrized random functions. Later, the generator is extended to produce more realistic moving behavior such as group movement and obstructed movement, by introducing the notion of clustered movement and a new parameter [24]. G-TERD [34] is a generator for time evolving regional data in an unconstrained space and Oporto [27] is a realistic scenario generator for moving objects motivated by a specific application, fishing. A set of rules is defined to create indoor moving objects in [19, 39], like an object in a room can move to the hallway or move inside the room. In summary, these generators only consider a single environment and cannot generate moving objects passing through different environments.

There are also various spatio-temporal simulators for different applications. ST-ACTS [15] is a simulator that uses geo-statistical data sources and intuitive principles to model *social* and *geo-demographic* aspects of human mobility. The model is based on commercial source data describing some statistics of Denmark's population. Some principles are defined to govern the social aspects of mobility, e.g., home-work and home-school. STEPS [23] is a parametric mobility model for human mobility, which makes abstraction of spatio-temporal preferences in human mobility by using a power law to rule the movement. The work focuses on human geographic mobility and

defines the human mobility as a finite state. The mobility is modeled by a discrete-time Markov chain in which the transition probability distribution expresses a movement pattern. GAMMA [18] is a framework in which trajectory generation is treated as an optimization problem and solved by a genetic algorithm. Two examples are given to show how to configure the framework for different simulation objectives, in a cellular space and a real-life symbolic moving behavior. However, the aforementioned methods do not consider the detailed routing between locations, that is, how people move from one place to another. As a result, transportation modes and movement environments are not addressed.

The goal in [21] is to provide a friend-finder service. The considered places are not general, including only home and entertainment places. Also the observed time periods are limited to Friday and Saturday nights. The method sets some parameters for the simulation, like source, destination, and starting time. In order to have a realistic model of distributions, a survey is prepared to collect the data of real users based on interviews of more than 300 people. In the simulation, home places are distributed almost uniformly on the map, with a minor concentration on central zones of the city.

3 Benchmark Data

3.1 Data Model

We let the *space* for generic moving objects be covered by a set of infrastructures (environments), each of which corresponds to an environment and contains its possible transportation modes. A notation is defined for each infrastructure, listed in Table 1. Transportation modes are summarized in Def. 3.1.

$Space$	I_{rn}: Road Network	*Car, Taxi, Bike*
	I_{rbo}: Region-based Outdoor	*Walk*
	I_{bn}: Bus Network	*Bus*
	I_{mn}: Metro Network	*Metro*
	I_{indoor}: Indoor	*Indoor*

Table 1: Components for Space

Definition 3.1 *Transportation Mode*
TM = {*Car, Bus, Walk, Indoor, Metro, Taxi, Bike*}

Each infrastructure consists of a set of infrastructure objects (IFOBs) representing available places for moving objects. For example, streets and roads constitute I_{rn} and polygons representing pavement areas compose I_{rbo}. The bus network I_{bn} comprises bus routes, bus stops and moving buses. The location of a moving object is represented by referencing to the underlying IFOBs. We give the definition below.

Definition 3.2 *Generic Location*
$D_{\underline{genloc}} = \{(oid, (loc_1, loc_2)) | oid \in D_{\underline{int}}, loc_1, loc_2 \in D_{\underline{real}}\}$

A generic location consists of two attributes with oid being an IFOB id and (loc_1, loc_2) describing the relative position according to that object. The representation has different semantics according to the infrastructure characteristic. For example, in a road network oid is a route identifier and $(loc_1, \perp)$ records the relative location on the route. Given a location in I_{rbo}, oid maps to a polygon and (loc_1, loc_2) represents the location inside the polygon.

We denote a generic moving object by $mo = < u_1, u_2, ..., u_n >$, that is a sequence of temporal units ordered by time where each unit defines the movement during a time interval. In detail, we have
$u_i = (i, gl_1, gl_2, m)$ $(gl_1, gl_2 \in D_{\underline{genloc}} \wedge gl_1.oid = gl_2.oid, m \in$ TM)

where i denotes the time interval, gl_1, gl_2 are the start and end locations, respectively, and m is the transportation mode. We assume that the object moves linearly during i so that the positions between gl_1 and gl_2 are calculated by a linear function. Consider such an example movement: *Car* $\rightarrow$ *Walk* $\rightarrow$ *Indoor*. The units of mo will record ids for (1) roads; (2) pavement areas; (3) rooms. The precise locations are identified by gl_1 (gl_2).

The method represents a moving object in a compact way. Two units u_i, u_j are merged into one if they fulfill the conditions: (1) $u_i.i$ and $u_j.i$ are *adjacent*; (2) $u_i.m$ = $u_j.m$; (3) u_i and u_j reference to the same IFOB and the linear functions are the same. Some examples are shown in the following. (1) locations for a car (taxi or bicycle) moving on a road segment with a constant speed are compressed into one unit denoting (i) the road id and (ii) the positions of endpoints for such a segment; (2) locations for a bus (metro) traveler are not explicitly recorded but reference to the bus (metro) and this can avoid the redundant data for people who take the same bus. Consequently, the storage size for moving objects is greatly reduced.

3.2 Movement Principles

In order to create realistic benchmark data, we define a set of movement rules that can reflect the characteristics and distributions of human behavior in practice. Human movement has a certain distribution in terms of time and location, and in most cases people move from one place to another with the objective of performing a certain activity at the target place. We define in total four movement rules that consider both (1) *physical* and (2) *social* aspects of mobility. The former considers a geographical impact, e.g., location, distance, while the latter regards human community behavior, preferences or habits. We believe that both *physical* and *social* factors are essential for creating the benchmark data and are able to mirror key features of human movement. The four rules are defined as follows.

- **MR1**. This rule is to represent the *regular* movement, which usually has a periodic pattern. A large majority of people go to work in the morning and come back in the evening. There are additional trips between work places during the office time, as people may travel to another place for business or conference meeting. Evidently, these regular trips occur on weekdays in most cases.

- **MR2**. Movement in a *single* environment. For example, people walk around in the city center (pedestrian areas) for shopping on the weekend, and a clerk moves from his office room to the conference room. This rule is used to generate a short trip limited to one environment.

- **MR3**. Motivated by the famous *nearest neighbor* query, we create a trip from the query location to the closest qualified location. For instance, a person wants to find the nearest hospital. If the target place is a short distance away from the query location, the traveler can go by walking. Otherwise, he may wish to travel by the public transportation system. In this case, the closest bus stop is found and then he travels by bus. Another example could be a car searching the nearest gas station. Trips generated by this method are based on the physical aspect, i.e., distance.

- **MR4**. A trip is triggered by the purpose of visiting a point of interest. The concept of an interesting location is general, a personal apartment, a sightseeing place, a restaurant, etc. On the weekend, people may visit friends, go shopping or meet in a park.

Compared with **MR1**, trips defined by **MR2**, **MR3** and **MR4** can be considered as irregular movement, but they occur frequently in daily life. There is a large amount of such trips, especially for **MR3**. An extensive study that has been done on *NN* queries in moving object databases, confirms that this query is fairly common and widely used in daily life, motivating us to define a rule for the movement performing such an activity. Trips generated by **MR1**, **MR3** and **MR4** may pass through different environments, leading to multiple transportation modes.

The trips created by above rules reflect the most common movement of humans, leading to the generated data in a realistic scenario. The purpose of defining precise rules is to generate the data in a well defined method and a

clear motivation. The method is flexible and one can add more rules to generate desired and interesting trips, e.g., a traveler passes a sequence of places at the minimum cost.

3.3 Parameters

To create a generic moving object, three parameters have to be configured: (1) **Location**; (2) **Time**; (3) **Transportation Modes**.

Location plays a key role for creating trips for the reason that it specifies where the trip starts and ends. No restrictions are made for the start and end locations of a traveler, i.e., they can be in any environment defined in Table 1. In particular, the location is related to an IFOB, e.g., a road or a bus. If the start and end locations are in distinct environments, a trip involving different transportation modes is created. In the following, we show how the location parameter is specified to create desired trips. Stated in Sec. 3.1, the whole space is partitioned into a set of infrastructures each of which consists of a set of IFOBs. A unique number is assigned to each IFOB, and each infrastructure has a range of integers denoting its element ids.

Definition 3.3 *Integer Sets for IFOB Ids*

Let IO_ID=$\{$ID_R, ID_P, ID_B, ID_M, ID_IN$\}$ be the overall integer set for all IFOB ids and the following two conditions hold:

(i) $\forall\, U \in IO\_ID, U \subset D_{\underline{int}}$;

(ii) $\forall\, U, V \in IO\_ID, U \neq V \Rightarrow U \cap V = \varnothing$.

Each element in IO_ID is a set storing integers that represent the ids of IFOBs in the corresponding infrastructure. An element is denoted by a symbol ID plus the first (two) character(s) of the environment name. For example, ID_R stores all road and street ids in I_{rn}, and ID_IN records building ids in I_{indoor}. There is no overlapping between the integer sets from different infrastructures. Since the location representation (Def. 3.2 in Sec. 3.1) records an IFOB id, one can set an integer range denoting certain IFOBs. The range can determine the environment for the start or end location. The parameter is defined below.

Definition 3.4 *Location Parameter*

A pair $l(l_s, l_e)$ denotes the start and end locations of a trip with the conditions:

(i) $l_s = (gl_s, c_s)$, $l_e = (gl_e, c_e)$ $(gl_s, gl_e \in D_{\underline{genloc}}, c_s, c_e \subseteq IO\_ID)$;

(ii) $gl_s.oid \in c_s \wedge gl_e.oid \in c_e$.

The location parameter is represented by a pair of objects defining the start and end locations as well as the constraints, c_s and c_e. They are two sets each of which designates an integer range for the location. That is, the environment for gl_s (gl_e) is determined by c_s (c_e). For example, if c_s = ID_R, the start location is on a road. If c_s = ID_P, gl_s is located in the pavement area. For the case c_s = c_e = ID_IN, there are two possibilities: (1) gl_s and gl_e are in the same building; (2) gl_s and gl_e are in different buildings. Case (1) is simple. Case (2) implies a trip from one building to another, e.g., from home to office.

By defining c_s and c_e, one can set up the location in a flexible way. $c_s(c_e)$ can be either (i) a range indicating a set of ids or (ii) a set with a single value denoting a specific object id. For case (i), a concrete value belonging to the given set needs to be specified in order to let the trip start from a precise IFOB. We let such a value be randomly chosen from the set. For example, if c_s = ID_R, a stochastic road is selected. For case (ii), the method is able to create a trip starting from (ending at) a specific IFOB, e.g., a road or a building. In the above two cases, when a concrete IFOB is determined, we randomly choose a location belonging to the IFOB and let it be the accurate start (end) location. For instance, if a road is chosen, we let a stochastic position on the road be the start (end) location of a trip. If a building is selected, a random location inside a room is chosen.

Time. In principle, a trip can start at any time instant. But human movement has a certain time distribution that most trips occur in the range [6:00, 22:00].

Definition 3.5 *Time Parameter*
Let $Hour = \{0, 1, ..., 23\}$ *and* $Min = \{0, 1, ..., 59\}$ *be two sets of integers.*
The start time of a trip is defined as a four-tuple $t(h, min, hc, minc)$ *where*
(i) $h \in hc \subseteq Hour$*;*
(ii) $min \in minc \subseteq Min$*.*

The two attributes h and min define the start time of a trip and each value is set according to its corresponding constraint. We have hc for h and $minc$ for min. One can generate the desired trip on time distribution by configuring hc and $minc$. For example, to create a trip from home to work place in the morning, the time parameter can be set by $hc = \{6, 7, 8\}$ and $minc = Min$. We let the concrete value for h and min be uniformly distributed in the defined sets hc and $minc$. This method is general and effective, and is able to satisfy different requirements as one can shrink or enlarge the ranges for hc and $minc$ to get certain distribution. For example, we can set $hc = \{8\}$ and $minc = \{0, 1, ..., 30\}$ to create a trip whose start time is between 8am and 8:30am.

Modes. People have different choices on vehicles for their traveling, by car or using public transportation system. If the traveling distance is short, the mode *Walk* or *Bike* can also suffice. A trip can contain a single mode or multiple modes. The value depends on the start and end locations to some extent. Regarding the locations, we make the following assumptions for the transportation modes involved by a trip.

- Assumption 1. If l_s and l_e are located in the same building, we define the movement to be inside such a building. That is, the mode is only *Indoor*.
- Assumption 2. If l_s and l_e belong to the same outdoor environment such as I_{rn} or I_{rbo}, then the mode is single and determined by the environment, e.g., *Car* in I_{rn}.

Based on the two assumptions, the case that a trip contains different transportation modes occurs when (1) l_s and l_e are located in different buildings; or (2) l_s and l_e belong to different infrastructures. Otherwise, the modes are determined. Recalling Def. 3.1, we do the following partition on TM, seeing below.

Definition 3.6 *Partition of Transportation Modes*
TM = A $\cup$ *B where*
(i) A = $\{$*Walk*$\}$*;*
(ii) B = B1 $\cup$ *B2* $\cup$ *B3 where*
B1=$\{$*Indoor*$\}$*, B2=*$\{$*Bus, Metro, Taxi*$\}$*, B3=*$\{$*Car, Bike*$\}$*.*

TM is first partitioned into two groups A and B where A contains only one element. The modes in B are further grouped, thus we can distinguish between (1) indoor and outdoor; (2) public transportation modes and private vehicles. We define *Walk* to be the only mode that connects to another different mode, e.g., *Walk* $\rightarrow$ *Car*, *Indoor* $\rightarrow$ *Walk* $\rightarrow$ *Bus*. The connection between two modes such as *Car* $\rightarrow$ *Indoor* and *Indoor* $\rightarrow$ *Bus* is not allowed, implying there is no mode switch between elements from B. Usually, people do not directly change from *Bus* to *Car*, or from *Indoor* to *Taxi*. A short distance of walking is required. This is consistent with the result in [40] where the authors infer transportation modes from raw GPS data and find that the walk segment is up to 99% as the transition between different transportation modes. Table 2 gives the transition matrix for TM. A is transitive to all the others, while B1, B2, B3 are only transitive to themselves and A.

Definition 3.7 *Transportation Modes Parameter*
The mode parameter is a set $mc \subset$ *TM to specify the modes that are to be included by a trip.*

See some instances: (1) mc = B1, movement inside a building; (2) mc = A $\cup$ B2, take the bus and a short distance walking; (3) mc = A $\cup$ B1 $\cup$ B3. Note that if the mode is already determined by l_s and l_e (the two assumptions above), mc is decided and does not have to be specified.

	A	B1	B2	B3
A	1	1	1	1
B1	1	1	0	0
B2	1	0	1	0
B3	1	0	0	1

Table 2: Transportation Mode Transition

3.4 Configure Parameters

In this part, we show how to set the three parameters to create trips simulating human movement. At first, we divide the integer set ID_IN representing buildings in I_{indoor} into three subgroups {H, W, SE}, where H refers to the set for personal houses, W denotes the ids for work buildings such as office buildings, universities, and SE (Shopping and Entertainment) is for buildings like shopping malls, cinemas. If a trip starts from or ends in the indoor environment (a building), we use the subgroup to determine the type of the building. H, W and SE are also used to create certain trips according to the rule. For example, we can set the constraint in the location parameter by H $\cup$ W for **MR1** to represent regular movement between home and work. The trip representing friends visiting on the weekend (**MR4**) can set H for both start and end locations.

A trip is an optimal route with respect to the minimum cost (e.g., distance, time) from one location to another and the result is used to create a generic moving object. The start and end locations can be in one environment or in different environments, where the latter involves multiple transportation modes. For each rule, a set of movement instances is specified to create concrete trips. Table 3 lists some instances for each rule as well as the parameter settings. For simplicity, we only show the constraint value for each parameter. Note that in some cases transportation modes are in fact decided by the location (see **MR2**). Otherwise, mc is variable and can be specified according to the requirement. For example, people can drive by car or use the public transportation system to travel between home and work. The values in the table are possible settings.

Rule	Movement Instance	c_s, c_e	hc	mc
MR1	travel from home to work	H, W	$[6, 9]$	A $\cup$ B1 $\cup$ B2
	business travel between working places	W, W	$[9, 16]$	A $\cup$ B1 $\cup$ B3
MR2	people walk around in the city center	ID_P, ID_P	$[10, 18]$	A
	a clerk walks from one office to another	W,W	$[9, 16]$	B1
MR3	a car searches the nearest restaurant	ID_R, ID_IN	$[10, 20]$	A $\cup$ B3
	a traveler looks for the nearest bus stop	ID_P, ID_B	$[9, 20]$	A
MR4	go to a shopping mall from home	H, SE	$[10, 20]$	A $\cup$ B1 $\cup$ B3
	visit friends	H, H	$[10, 20]$	A $\cup$ B2

Table 3: Example Movements and Parameter Settings

3.5 Trip Generation Algorithm

We outline the algorithm that generates a trip with different transportation modes. First, a set of graphs is introduced. A trip is created based on the shortest path (SP) where the start and end locations can be located in different environments, resulting in a set of sub trips in several infrastructures. In the sequel, different trip planning algorithms are needed, e.g., indoor navigation, trip planning for pedestrians, routing in bus network. Fig. 1(a) lists all infrastructure graphs.

Second, to create a trip passing through a set of environments, location mapping technique is required to convert positions from one system to another. Consider the case that a pedestrian searches the nearest bus stop.

To answer such a query, at first the location of the qualified bus stop is mapped to the pavement. This is because the representation of a bus stop is not simply a point but contains some other information such as the bus route id, the relative order on the route. Then, the shortest path from the query location to the bus stop is returned where the path is located in the walking area. Since a walking segment is the part connecting movements in different environments, the location mapping is actually between I_{rbo} and the other infrastructures. We denote the mapping by $M(I_{rbo}, I')$ ($I' \in \{I_{rn}, I_{bn}, I_{mn}, I_{indoor}\}$).

Third, a set of speed values are defined. We summarize them in Fig. 1(b). Each road is assigned a value as the maximum speed allowed for cars. Such a value is also used for the bus moving on the road. v_m is the speed for metros. The walking speed for both indoor and outdoor is specified by v_w. Without loss of generality, we use the first movement instance of **MR1** in Table 3 to describe the procedure of the algorithm.

Notation	Meaning
G_{rn}	Road Graph
G_{rbo}	Pavement Graph
G_{bn}	Bus Network Graph
G_{mn}	Metro Network Graph
G_{indoor}	Indoor Graph

(a) Infrastructure Graphs

Name	Meaning
S_c	car speed values
v_m	metro speed
v_w	walking speed

(b) Speed Values

Figure 1: Algorithm Parameters

Step 1: Assuming that the traveler uses the bus network, the corresponding graphs $\{G_{rbo}, G_{bn}, G_{indoor}\}$ are selected according to transportation mode parameters.

Step 2: Let s be the home location of the traveler and the nearest bus stop to s is found. $M(I_{rbo}, I_{bn})$ returns the pavement location for the bus stop, denoted by bs. By running the SP algorithm on G_{rbo}, a walking path l_1 from s to bs is obtained. We create an object $< l_1$, *Walk*$>$ and put it into the path set P.

Step 3: We execute the SP algorithm on G_{bn} to find a bus connection to the destination stop. Let l_2 be the bus path and the pair $< l_2$, *Bus* $>$ is inserted into P. If a short walking is included for changing the bus, such a path is also put into P with the mode *Walk*.

Step 4: $M(I_{rbo}, I_{bn})$ maps the destination stop to the pavement location be. Again, we use G_{rbo} to find the SP from be to e, the building entrance (maps to I_{rbo}). l_3 denotes such a walking path and we insert $< l_3$, *Walk*$>$ into P.

Step 5: We run the SP algorithm on G_{indoor} to find an indoor shortest path l_4 from e to the final destination, e.g., an office room. The last sub path as well as the transportation mode is collected and we put $< l_4$, *Indoor*$>$ into P.

Step 6: For each $p_i \in P$, we take the corresponding speed value from Fig. 1(b) and create the sub movement. In the end, we perform the union on all sub trips to get the complete trip.

4 Benchmark Queries

We present a set of interesting queries to form the benchmark workload, categorized into two groups. One requests data from infrastructures and the other deals with generic moving objects. We provide the formulation for all queries by using an SQL-like language in the Appendix A.

Infrastructure Queries.

- **Q1.** Find out all metros passing through the city center.
- **Q2.** Given a building, find all bus stops that are within a radius of 300 meters.
- **Q3.** Which streets does Bus No. 12 pass by?

- **Q4.** Where can I change between bus route No. 16 and No. 38? Where can I change between metro route No. 2 and No. 8?

The first three are concerned with interactions between different infrastructures as users may compare IFOBs from different environments. In a public transportation system, travelers need to know the place where they can switch between different routes.

Queries on Generic Moving Objects.

We consider the following aspects: (1) referenced IFOBs by moving objects; (2) transportation modes; (3) time intervals; (4) spatial data and locations.

- **Q5.** At 8am on Monday, who sits in the bus No. 32? At 8am on Monday, who sits in the metro No. 2?

This query deals with travelers who take the bus (metro) from a specific route at the given time and covers three aspects above: (1), (2) and (3). The query result is not the case for a particular bus or metro, but all available buses (metros). At the query time, there may be several buses (metros) moving on the route.

- **Q6.** What is the percentage of people traveling by public transportation vehicles?

For this query, we define the the modes $\{$*Bus*, *Metro*, *Taxi*$\}$ to be the case of traveling by public transportation vehicles.

- **Q7.** Where does *Bobby* walk during his trip? And how long does he walk?
- **Q8.** Find out all people passing room 312 at the office building between 9am and 11am on Monday.

To answer **Q7, Q8**, we need to get a sub trip according to the transportation mode.

- **Q9.** Who arrived by taxi at the university on Friday?
- **Q10.** Who entered bus No.3 at bus stop "University" on Tuesday afternoon?

The above two queries deal with interactions between different environments. To answer these queries, one should find the place where people change transportation modes.

- **Q11.** Find out all people walking through zone A and zone B on Saturday between 10am and 3pm.
- **Q12.** Did anyone who was on floor H-5 of the office building between 2pm and 5pm take a bus to the train station on Friday?
- **Q13.** Did bus No. 35 pass by any bicycle traveler on Monday?

Q13 considers the distance between two moving objects with different transportation modes. It is interesting to discover such a relationship as the two objects move in different environments. The bus parameter means a group of buses which all belong to the route No. 35 but with different departure time, instead of a particular bus trip.

- **Q14.** Did someone spend more than 15 minutes on waiting for the bus at the bus stop Cinema on Saturday?
- **Q15.** How many people visit the cinema on Saturday?
- **Q16.** Find out all people staying at room 154 in the university for more than one hour on Thursday.

Besides the outdoor trips, indoor moving objects can also be traced.

- **Q17.** Did someone spend more than one hour traveling by bus (metro)?
- **Q18.** Find out all people changing from bus No. A to bus No. B at stop X. Find out all people changing from metro No. A to metro No. B at stop Y.

In a public transportation system, knowing the place where people do the transfer is meaningful to analyze and investigate the schedule and route distribution.

- **Q19.** Find out all trips starting from zone A and ending in zone B by public transportation vehicles with the length of the walking path being less than 300 meters.

We search trips by considering the start and end locations as well as transportation modes. A and B are defined to be a set of locations, represented by regions. They can be areas with high road density, implying a large number of residences. The query is helpful for travel recommendation. Meaningful knowledge can be discovered from past movements.

- **Q20.** Find the top k bus (metro) routes with high passenger flow for all workdays.
- **Q21.** Find the top k road segments with high traffic during the rush hour for all workdays.

By analyzing the histories of movements, routes with high passenger flow can be discovered and the schedule in a public transportation system can be adjusted to improve the traffic. For **Q21**, the traffic value of a road segment is set as the number of moving objects passing by during the rush hour ([7:00, 9:00] $\cup$ [16:00, 18:00]), where the following transportation modes are considered: $\{$*Car*, *Taxi*, *Bicycle*, *Bus*$\}$.

5 Optimization

5.1 Transportation Mode Access

By observing and analyzing involved operators for queries, there is a frequently called procedure in the benchmark workload. That is checking whether a transportation mode is included by a moving object mo. Since a large part of queries specifies a transportation mode (e.g., **Q7**, **Q9**, **Q16**), one needs to find all qualified moving objects. To get the result, one option is to sequentially scan all units of mo, comparing the mode attribute to see if the value is identical to the argument. However, this yields a linear searching for each mo. The ability to determine the existence of a mode in a fast way can reduce the overall running time. We optimize the procedure as follows. An integer IM (32 bits) is assigned to each mo to denote the involved transportation modes where a bit indicates whether a mode exists or not. Each transportation mode is assigned a value as an index to locate the corresponding bit in IM. We mark the bit *true* if the mode exists and *false* if it does not exist. Suppose that the least significant (right-most) seven bits are used for the modes in Def. 3.1. We assign 0 for *Car* as the bit index, 1 for *Bus*, and so on. A moving object with the following sequence of modes *Indoor* $\rightarrow$ *Walk* $\rightarrow$ *Bus* $\rightarrow$ *Walk* defines the value 14 (binary 0001110). We calculate IM for each mo in advance and store it as an attribute along with mo in a tuple. Later, if a query needs to determine the existence of a mode, both the query mode and the integer are taken as input. The index for the bit denoting the mode is calculated, and the stored integer is examined.

Assume that there are N trips stored in the database and one trip contains M units on average. The original method needs NM times of unit access to find all qualified trips. With the optimization technique, we can achieve the result by N integer comparisons.

5.2 Index on Units

Recall that a moving object is represented by a set of units arranged in a linear order on time. To retrieve a sub trip with a certain mode, at present we need to linearly traverse all units to check the value and return the

qualified data. For instance, to answer **Q5** the movement with the mode *Bus* is returned, and in **Q7** walking trips are specified. For travelers who take public vehicles, one might get the sub trip on a particular bus (metro), seeing **Q10** and **Q18**. Obviously, the sequential scan yields poor performance for large data. This motivates us to build an index in order to access units according to transportation modes in a fast way.

Let $I = \{(m, l, h) | m \in \text{TM}, l, h \in D_{\underline{int}}\}$ be the index built on the units of mo. Each element in I consists of three attributes where m records the transportation mode and l, h are entries pointing to the start and end locations of a sequence of units with the mode m. That is, each element maps to a sub movement in mo with a single mode. For example, we have such an example movement

$mo = <$ (*Indoor*)$_1$, (*Indoor*)$_2$, ... , (*Indoor*)$_{10}$,
(*Walk*)$_{11}$, (*Walk*)$_{12}$, ... , (*Walk*)$_{15}$,
(*Bus*)$_{16}$, (*Walk*)$_{17}$, ... , (*Walk*)$_{20}>$.

The index built on mo is I = {(*Indoor*, 1, 10), (*Walk*, 11, 15), (*Bus*, 16, 16), (*Walk*, 17, 20)}. Although I is still a list structure, the quantity of elements depends on the number of modes in mo, usually quite few. Each element locates the range of units with a certain mode. Consequently, given a mode m we first scan the index to determine the positions of qualified units and then access them to get the concrete data. The advantage is a large number of units that do not fulfill the condition can be skipped by visiting I.

5.3 Projecting the Movement

The above two methods apply to almost all queries on moving objects, whereas we also develop a technique to improve the efficiency of a specific query (**Q13**) that originally needs a long processing time. To answer such a query, the procedure maps the following moving objects into free space: (1) moving objects with the mode *Bike* and (2) a set of buses belonging to one route, with the aim of computing the distance in the same environment. The distance value is represented by a moving real which consists of a sequence of units. Each unit describes the distance function in a time interval. We briefly introduce the procedure of calculating such a value. Let mo_B and mo_b denote a *Bike* traveler and a bus, respectively. Each object is represented by a moving point that contains a set of pieces, each describing a linear movement from one location to another during a time interval. Since the object moves in free space, the location is identified by a point. The algorithm first refines the units from both mo_B and mo_b to get two subsets that have overlapping time intervals, and then sequentially scans the subsets to calculate the value between each pair of qualified pieces (time intervals have intersections). This method deteriorates over long time and suffers from poor performance if the refined subset contains a large number of elements. For example, if mo_B and mo_b have almost the same period (perhaps a long time interval), then the procedure accesses nearly all units of mo_B and mo_b to compute the distance.

Let us consider the places where mo_B and mo_b can be located. All roads are available for mo_B, but mo_b only moves along the pre-defined bus route. Evidently, the case that mo_b passes mo_B can only occur when mo_B is moving on the road segment that the bus route maps to. In other cases, it is not necessary to calculate the distance even when the time intervals of mo_B and mo_b are overlapping. For example, mo_b can not pass mo_B if they are far away from each other. Consequently, before running the costly algorithm of calculating the distance between two moving points, we first project the *Bike* movement to road segments that the bus route maps to. This leads to a small number of units for a moving point as unqualified movement pieces are pruned. Usually, the road segments on which a bike traveler moves and a bus route do not have too many intersections. Given a bus route, we map it to a set of road segments each of which is denoted by (rid, l_s, l_e), where rid ($\in D_{\underline{int}}$) indicates the road id and l_s, l_e ($\in D_{\underline{real}}$) refer to the start and end locations of such a road segment, respectively. Recalling the location representation in Def. 3.2 of Sec. 3.1, the value oid corresponds to a road if the moving object is a bike traveler, resulting in fast access to the road segment.

6 Performance Evaluation

This section shows extensive experimental results of the evaluation. The implementation is developed in an extensible database system Secondo [17] and programmed in C/C++ and Java. A standard PC (AMD 3.0 GHz, 4 GB memory, 2TB disk) running Suse Linux (kernel version 11.3) is used. We utilize the tool MWGen [38] to create all infrastructure data based on real road datasets and public floor plans (e.g., [5, 3, 4]). Two road datasets are used, Berlin [1] and Houston [2]. The tool takes roads and floor plans as input and generates pavement areas, bus network, metro network and a set of buildings. The infrastructures are referenced by moving objects for location representation.

6.1 Experimental Setup

Our benchmark generator provides a set of configuration parameters that allow a user to create the desired datasets, including (1) the total size of moving objects; (2) the number of trips according to each movement rule; (3) distribution of transportation modes; (4) start and end locations; (5) time distribution. In the following, we present the setting in our experiment. We simulate one week movement. All trips are created based on the rules presented in Section 3.2 and the distribution is listed in Fig. 2(a). We add one more pattern to create moving objects each of which visits several places, e.g., home $\rightarrow$ shopping $\rightarrow$ restaurant $\rightarrow$ home. The time distribution of each rule is shown in Fig. 2(b).

Name	Percentage
MR1	40%
MR2	10%
MR3	10%
MR4	30%
Long Trips	10%

(a) Rule Distribution

Name	Time Periods	Days
MR1	[6:30, 19:00]	Mon-Fri
MR2	[9:00, 17:00]	Mon-Sat
MR3	[9:00, 17:00]	Mon-Sat
MR4	[10:00, 21:00]	Mon-Sun
Long Trips	[6:30, 20:00]	Mon-Sat

(b) Time Distribution

Name	Berlin	Houston
MR1, **MR4**	*Indoor* + *Walk* + { *Bike* : 5%, *Car* : 50%, *Bus* : 20%, *Metro* : 20%, *Taxi* : 5% }	*Indoor* + *Walk* + { *Bike* : 5%, *Car* : 60%, *Bus* : 15%, *Metro* : 15%, *Taxi* : 5% }
MR2	*Walk*: 40%; *Indoor*: 60%	*Walk*: 40%; *Indoor*: 60%
MR3	*Indoor* + *Walk* + { *Car* : 50%, *Bus* : 35%, *Taxi* : 15% }	*Indoor* + *Walk* + { *Car* : 60%, *Bus* : 28%, *Taxi* : 12% }
Long Trips	*Indoor* + *Walk* + *Car*: 100%	*Indoor* + *Walk* + *Car*: 100%

(c) Transportation Modes Distribution

Figure 2: Benchmark Generator Parameters

We summarize the distribution of transportation modes in Fig. 2(c). All trips except those from **MR2** include both indoor and walking movement. **MR2** defines the movement in one environment (Region-based Outdoor or Indoor), leading to the determined mode. For trips based on **MR3**, we let part of them travel by car and the other use public transportation vehicles. Since the trip is motivated by nearest neighbor query, we find that usually the distance from the query location to the target place is not very far. A distance threshold is defined for the resulting path to determine whether the traveler will go on foot directly or by bus (taxi). This applies to the case that the query user is a pedestrian, not for a car. We do not include the mode *Metro* because usually such a mode is not contained in a short distance trip. For a long distance journey that visits more than one place, we let the modes be *Indoor + Walk + Car*. For some rules, we let the two cities have different distributions on transportation modes.

6.2 Datasets

Fig. 3 lists all datasets that we use for the experiment. All infrastructure data are shown in Fig. 3(a) and the detailed information of buildings is reported in Fig.3(b). We create a set of buildings of different types based on their floor plans to simulate a city environment. Besides the static IFOBs, there are buses and metros represented by moving objects. In both bus and metro networks, we define a schedule for each route to create trips. For the bus schedule, there are more trips on the day $\in$ {Mon, Tue, Wed, Thu, Fri, Sat} than on Sunday. For metros, the schedules are the same for the whole week. The detailed information is shown in Fig. 3(c). Moving objects datasets are described in Fig. 3(d). The generator allows to create data sets of varying sizes, thus is a flexible tool for evaluation.

	Berlin	Houston
X Range	[0, 44411]	[0, 133573]
Y Range	[0, 34781]	[0, 163280]
Roads	3,250	4,575
No. Vertices in P	116,516	437,279
Bus Routes	89	92
Metro Routes	10	16
Buildings	4,996	5,992
Size	2.5 G	9.7 G

(a) Infrastructure Data

Type	Berlin	Houston	Rooms NO.
house	3,713	5,000	
officeA	600	530	294
officeB	487	266	214
shopping mall	80	79	360
cinema	6	8	21
hotel	39	40	584
hospital	46	48	89
university	20	20	431
train station	1	1	56

(b) Statistics of Buildings

	Berlin	Houston
No. Bus Trips in One Day (From Mon to Sat)	5,220	5,594
No. Bus Trips On Sun	2,660	2,788
No. Metro Trips in One Day (From Mon to Sun)	1,897	3,046

(c) Mobile Infrastructure Data

Name	Trips No.	Total Units No. (M)	Avg. Units No. Per Trip	Disk Size (G)
Berlin10k	10,068	0.585	58.1	0.086
Berlin20k	20,127	1.17	58.4	0.174
Berlin50k	50,289	2.93	58.3	0.435
Berlin100k	100,635	5.86	58.3	0.871
Berlin200k	201,209	11.75	58.4	1.75
Berlin500k	503,673	29.36	58.3	4.36
Houston500k	515,664	29.73	57.6	4.4

(d) Moving Objects

Figure 3: Classification of Datasets

6.3 Benchmark Performance

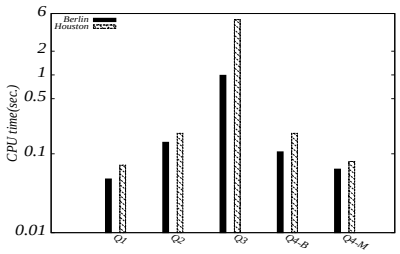
We use the CPU time and I/O accesses as performance metrics where the I/O accesses mean the number of pages that are read into the cache. In the database system, the cache size is set as 64M and one page size is 4k. The results of each query are averaged over five runs. For the constant value of a query such as a particular person, a university or an office building, we manually select an arbitrary value from the possible set.

6.3.1 Queries on Infrastructures

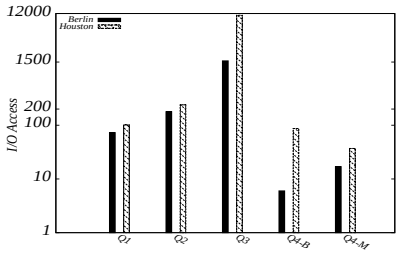
Fig. 4 shows the cost of infrastructure queries where both time and I/O measurements are plotted in logarithmic scale. The results show that the query cost in Houston is higher than that of Berlin. This is because Houston contains more infrastructure objects and occupies a larger area. **Q3** takes more time and I/O accesses than other queries as the procedure involves the costly intersection computation between roads and bus routes.

6.3.2 Scaling Experiments

This part investigates the system performance when the data size increases. We choose the infrastructure data of Berlin and generate different numbers of trips, from Berlin10k to Berlin500k. Fig. 5 depicts the cost where the result is the sum over all queries. When the input size increases, both CPU time and I/O accesses rise proportionally. The detailed value of each query is reported in Fig. 11 in Appendix B.



(a) CPU Time



(b) I/O Access

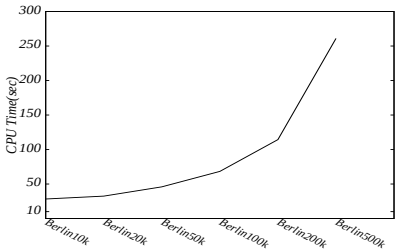
Dataset	Q1	Q2	Q3	Q4-B	Q4-M
Berlin	0.048	0.14	0.99	0.106	0.064
Houston	0.072	0.182	5.02	0.182	0.08

(c) CPU Time (sec)

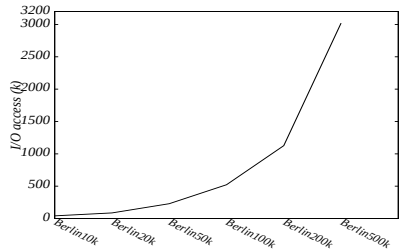
Dataset	Q1	Q2	Q3	Q4-B	Q4-M
Berlin	73	179	1582	6	17
Houston	101	240	11157	87	37

(d) I/O Access

Figure 4: Query Cost



(a) CPU Time



(b) I/O Access

Figure 5: Total Query Cost

6.3.3 Optimization

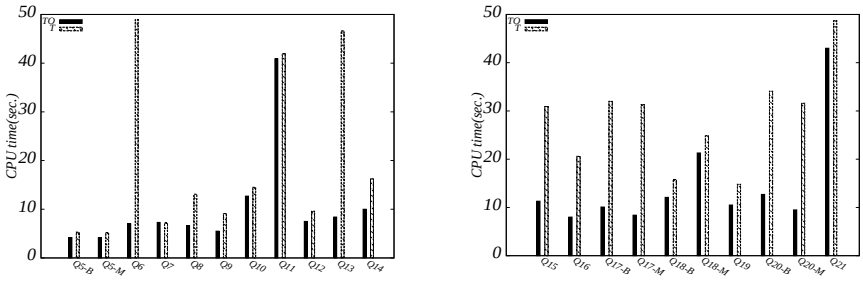
We show in Fig. 6 the extra cost of creating the integers on transportation modes and building the indices on moving objects. The additional storage size is calculated by summarizing the value of all trips. Each trip includes two storage parts: (1) an integer built on transportation modes; (2) an index built on units. Such a formula $\sum_{i=1}^{N}(b+\sum_{j=1}^{|I_j|}3*b)$ is used to compute the total storage cost where N refers to the number of trips in total, $|I_j|$ denotes the number of elements in the index for each trip and b shows the number of bytes for an integer. In the system, we use b = 4 bytes.

	Additional Storage Size (M)	Creation Time (sec)
Berlin500k	93.62	86.39
Houston500k	89.38	88.28

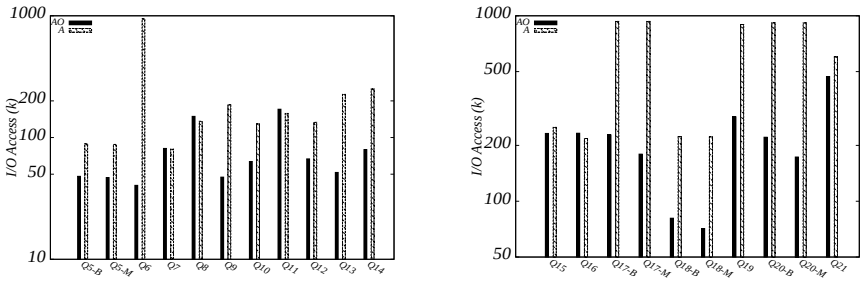
Figure 6: Indices Cost

We use the symbols TO and T to denote the time cost for a query with and without optimization, respectively, and symbols AO and A refer to the number of I/O accesses for a query with and without optimization. Fig. 7 and Fig. 8 depict the query cost of Berlin and Houston, respectively. The result shows that the CPU time is reduced by optimization techniques for almost all queries, and some queries are significantly improved, e.g., **Q6**, **Q15**. There is one query (**Q7**) that does not need optimization techniques so that the results are nearly the same for two cases. In a few occasions, the number of I/O accesses by AO is a little more than A (e.g., **Q8**). But in most cases, optimization techniques still show the effect and advantage, demonstrated in both figures.

In both datasets, by optimization the CPU time is around 10 seconds for most queries. **Q21** takes the most among all queries as a majority of time is spent on mapping bus routes into road segments, up to 80%. We put the accurate cost value for each query in Fig. 12 and Fig. 13 in Appendix-B.

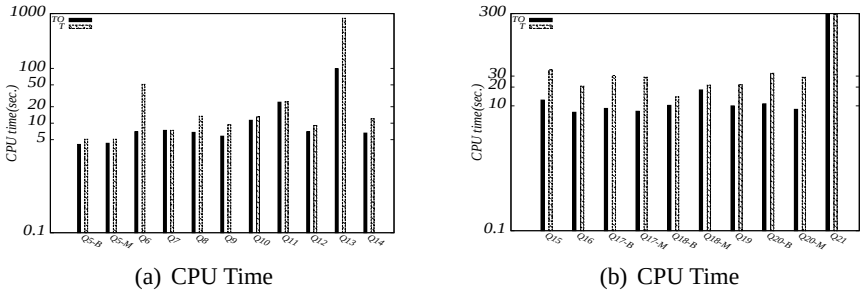


(a) CPU Time (b) CPU Time

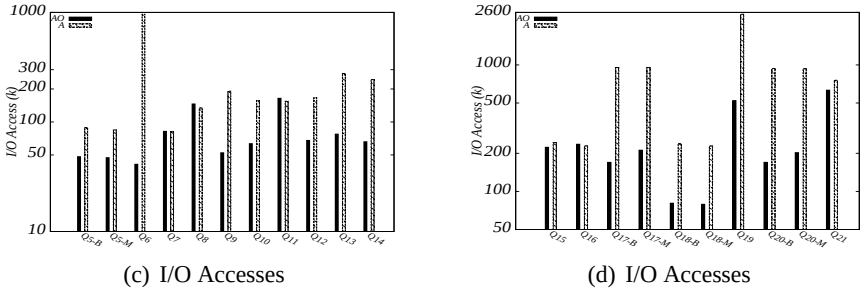


(c) I/O Accesses (d) I/O Accesses

Figure 7: Berlin500K



(a) CPU Time (b) CPU Time



(c) I/O Accesses (d) I/O Accesses

Figure 8: Houston500K

7 Conclusions

In this paper, we propose a benchmark GenMOBench to evaluate the performance of a database system managing moving objects that travel through different environments. We develop a method to create the benchmark data in a realistic scenario with the aim of reflecting the distribution and characteristics of human movement in practice. A group of queries on infrastructure data and moving objects is set as the benchmark workload. Optimization techniques are proposed to reduce the query cost. The efficiency and effectiveness of the proposed techniques are studied through comprehensive experiments and the detailed experimental results are reported.

References

[1] http://www.bbbike.de/cgi-bin/bbbike.cgi (2008).

[2] http://www.census.gov/geo/www/tiger/tgrshp2010/tgrshp2010.html (2011).

[3] http://www.edenresort.com/home (2011).

[4] http://www.greenhosp.org/floor_plans.asp (2011).

[5] http://www.modulargenius.com/default.aspx (2011).

[6] V. Bauer, J. Gamper, R. Loperfido, S. Profanter, S. Putzer, and I. Timko. Computing isochrones in multi-modal, schedule-based transport networks. In *ACM GIS, Demo*, 2008.

[7] D. Bitton, D. J. DeWitt, and C. Turbyfill. Benchmarking database systems a systematic approach. In *VLDB*, pages 8–19, 1983.

[8] J. Booth, P. Sistla, O. Wolfson, and I.F. Cruz. A data model for trip planning in multimodal transportation systems. In *EDBT*, pages 994–1005, 2009.

[9] T. Brinkhoff. Generating network-based moving objects. In *SSDBM*, pages 253–255, 2000.

[10] T. Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.

[11] M. J. Carey, D. J. DeWitt, and J. F. Naughton. The oo7 benchmark. In *SIGMOD Conference*, pages 12–21, 1993.

[12] S. Chen, C. S. Jensen, and D. Lin. A benchmark for evaluating moving object indexes. *PVLDB*, 1(2):1574–1585, 2008.

[13] S. Duan, A. Kementsietsidis, K. Srinivas, and O. Udrea. Apples and oranges: a comparison of rdf benchmarks and real rdf datasets. In *SIGMOD Conference*, pages 145–156, 2011.

[14] C. Düntgen, T. Behr, and R. H. Güting. Berlinmod: A benchmark for moving object databases. *VLDB Journal*, 18(6):1335–1368, 2009.

[15] G. Gidófalvi and T. B. Pedersen. St-acts: a spatio-temporal activity simulator. In *GIS*, pages 155–162, 2006.

[16] M. C. González, C. A. Hidalgo R., and A. Barabási. Understanding individual human mobility patterns. *Nature*, 453:779–282, 2008.

[17] R. H. Güting, V. Almedia, D. Ansorge, T. Behr, Z. Ding, T. Höse, F. Hoffmann, and M. Spiekermann. Secondo:an extensible dbms platform for research prototyping and teaching. In *ICDE, Demo Paper*, pages 1115–1116, 2005.

[18] H. Hu and D. L. Lee. Gamma: A framework for moving object simulation. In *SSTD*, pages 37–54, 2005.

[19] C. S. Jensen, H. Lu, and B. Yang. Indexing the trajectories of moving objects in symbolic indoor space. In *SSTD*, pages 208–227, 2009.

[20] C. S. Jensen, D. Tiesyte, and N. Tradisauskas. The cost benchmark-comparison and evaluation of spatio-temporal indexes. In *DASFAA*, pages 125–140, 2006.

[21] S. Mascetti, D. Freni, C. Bettini, X. S. Wang, and S. Jajodia. On the impact of user movement simulations in the evaluation of lbs privacy- preserving techniques. In *PiLBA*, 2008.

[22] J. Myllymaki and J. H. Kaufman. Dynamark: A benchmark for dynamic spatial indexing. In *Mobile Data Management*, pages 92–105, 2003.

[23] A. D. Nguyen, P. Sénac, V. Ramiro, and M. Diaz. Steps - an approach for human mobility modeling. In *Networking (1)*, pages 254–265, 2011.

[24] D. Pfoser and Y. Theodoridis. Generating semantics-based trajectories of moving objects. *Computers, Environment and Urban Systems*, 27(3):243–263, 2003.

[25] S. Ray, B. Simion, and A. D. Brown. Jackpine: A benchmark to evaluate spatial database performance. In *ICDE*, pages 1139–1150, 2011.

[26] S. Reddy, M. Mun, J. Burke, D. Estrin, M. H. Hansen, and M. B. Srivastava. Using mobile phones to determine transportation modes. *TOSN*, 6(2), 2010.

[27] J. M. Saglio and J. Moreira. Oporto: A realistic scenario generator for moving objects. *GeoInformatica*, 5(1):71–93, 2001.

[28] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the positions of continuously moving objects. In *SIGMOD Conference*, pages 331–342, 2000.

[29] M. Schmidt, T. Hornung, G. Lausen, and C. Pinkel. Sp2bench: A sparql performance benchmark. In *ICDE*, pages 222–233, 2009.

[30] L. Stenneth, O. Wolfson, P. Yu, and B. Xu. Transportation mode detection using mobile devices and gis information. In *ACM SIGSPATIAL*, pages 54–63, 2011.

[31] M. Stonebraker, J. Frew, K. Gardels, and J. Meredith. The sequoia 2000 benchmark. In *SIGMOD Conference*, pages 2–11, 1993.

[32] Y. Theodoridis. Ten benchmark database queries for location-based services. *Comput. J.*, 46(6):713–725, 2003.

[33] Y. Theodoridis, J. R. O. Silva, and M. A. Nascimento. On the generation of spatiotemporal datasets. In *SSD*, pages 147–164, 1999.

[34] T. Tzouramanis, M. Vassilakopoulos, and Y. Manolopoulos. On the generation of time-evolving regional data. *GeoInformatica*, 6(3), 2002.

[35] P. F. Werstein. A performance benchmark for spatiotemporal databases. In *10th Annual Colloquium of the Spatial Information Research Centre*, pages 365–373, 1998.

[36] J. Xu and R. H. Güting. A generic data model for moving objects. Informatik-report 360, Fernuniversität Hagen, 2011.

[37] J. Xu and R. H. Güting. Infrastructures for research on multimodal moving objects. In *MDM, Demo Paper*, pages 329–332, 2011.

[38] J. Xu and R. H. Güting. Mwgen:a mini world generator. In *MDM, To Appear*, 2012.

[39] B. Yang, H. Lu, and C. S. Jensen. Probabilistic threshold k nearest neighbor queries over moving objects in symbolic indoor space. In *EDBT*, pages 335–346, 2010.

[40] Y. Zheng, Y. Chen, X. Xie, and W. Y. Ma. Understanding transportation mode based on gps data for web application. *ACM Transaction on the Web*, 4(1):1–36, 2010.

[41] Y. Zheng, L. Liu, L. Wang, and X. Xie. Learning transportation mode from raw gps data for geographic applications on the web. In *WWW*, pages 247–256, 2008.

Appendix A - Formulate Benchmark Queries

To formulate queries, we provide a relational interface to exchange information. Several relations are provided to manage infrastructure data and moving objects, summarized in Table 4. $\underline{route}$ is a data type that we propose to represent bus (metro) routes. A bus (metro) route is represented by a sequence of segments each of which defines the locations of start and end bus stops as well as the connection described by a line. For the indoor environment, we design a data type called $\underline{groom}$ used in Rel_Room to represent all rooms of a building, e.g., office rooms, corridors, staircases. Since in some cases one room can have several floors such as an amphitheater and a chamber, a $\underline{groom}$ object consists of a set of elements each of which defines the 2D area of a floor and the height above the ground level.

Road Network	Rel_Road	(R_id:int, GeoData: line, Name: string)
Region-based Outdoor	Rel_Rbo	(Reg_id:int, Reg: region, Name: string)
Bus Network	Rel_BStop	(Br_id:int, Stop_id: int, GeoData: point, Name: string)
	Rel_BRoute	(Br_id:int, GeoData: route, Name: string)
	Rel_Bus	(Bus_id:int, Br_id:int, Traj: genmo, Name: string)
Metro Network	Rel_MStop	(Mr_id:int, Stop_id: int, GeoData: point, Name: string)
	Rel_MRoute	(Mr_id:int, GeoData: route, Name: string)
	Rel_Metro	(Metro_id:int, Mr_id:int, Traj: genmo, Name: string)
Indoor	Rel_Building	(B_id: int, GeoData: region, Name: string)
	Rel_Room	(B_id: int, Room_id: int, GeoData: groom, Name: string)
Generic Moving Objects	MOGendon	(Mo_id:int, Traj: genmo, Name:string)

Table 4: Data Relation Schemas

To access an infrastructure, we assign a symbol to each relation (summarized in Fig.9(a)) and obtain the data from the operator **get_infra** described in Fig. 9(b). Fig. 10 lists the operators that are used to access the data and formulate the queries. To represent the trajectories of moving objects (i.e., the projection into space), we propose a data type $\underline{genrange}$ which defines a set of elements. Each element records the path according to a referenced object as well as the transportation mode.

ROAD	RBO	BUSSTOP	BUSROUTE	BUS
METROSTOP	METROROUTE	METRO	BUILDING	ROOM

(a) Infrastructure Symbols

Name	Signature
get_infra	$\underline{space} \times \underline{int} \rightarrow$ rel

(b) Access Relations

Figure 9: Access Infrastructures

We use qt to denote the query time parameter. For the queries (**Q4**, **Q5**, **Q17**, **Q18**, **Q20**) on both bus and metro, it is sufficient to show the query expression for a bus network as the procedure is similar for metros.

- **Q1**. Find out all metros passing through the city center.

  Let Reg_C be a region denoting the city center area.

```
SELECT mr.Mr_id
FROM get_infra(SpaceGendon, METROROUTE) as mr
WHERE mr.GeoData intersects Reg_C
```

Name	Signature
intersects	$\underline{route} \times \underline{region} \rightarrow \underline{bool}$
	$\underline{route} \times \underline{line} \rightarrow \underline{bool}$
	$\underline{periods} \times \underline{periods} \rightarrow \underline{bool}$
inside	$\underline{genloc} \times \underline{line} \rightarrow \underline{bool}$
	$\underline{genloc} \times \underline{region} \rightarrow \underline{bool}$
distance	$\underline{region} \times \underline{point} \rightarrow \underline{real}$
theloc	$\underline{int} \times \underline{real} \times \underline{real} \rightarrow \underline{genloc}$
length	$\underline{genrange} \rightarrow \underline{real}$
duration	$\underline{periods} \rightarrow \underline{real}$
deftime	$\underline{genmo} \rightarrow \underline{periods}$
val	$\underline{intime(genloc)} \rightarrow \underline{genloc}$
initial	$\underline{genmo} \rightarrow \underline{intime(genloc)}$
final	$\underline{genmo} \rightarrow \underline{intime(genloc)}$
ref_id	$\underline{genloc} \rightarrow \underline{int}$

(a) Basic

Name	Signature
atinstant	$\underline{genmo} \times \underline{instant} \rightarrow \underline{intime(genloc)}$
atperiods	$\underline{genmo} \times \underline{periods} \rightarrow \underline{genmo}$
contains	$\underline{genmo} \times \underline{tm} \rightarrow \underline{bool}$
	$\underline{genmo} \times \underline{int} \rightarrow \underline{bool}$
at	$\underline{genmo} \times \underline{tm} \rightarrow \underline{genmo}$
	$\underline{genmo} \times \underline{genloc} \rightarrow \underline{genmo}$
	$\underline{genmo} \times \underline{point} \rightarrow \underline{genmo}$
freespace	$\underline{genmo} \rightarrow \underline{mpoint}$
	$\underline{genloc} \rightarrow \underline{point}$
passes	$\underline{genmo} \times \underline{region} \rightarrow \underline{bool}$
	$\underline{genmo} \times \underline{groom} \rightarrow \underline{bool}$
trajectory	$\underline{genmo} \rightarrow \underline{genrange}$

(b) Advanced

Figure 10: Operators used in Queries

- **Q2**. Given a building named by X, find all bus stops within 300 meters.

```
SELECT bs
FROM get_infra(SpaceGendon, BUILDING) as b,
     get_infra(SpaceGendon, BUSSTOP) as bs
WHERE b.Name = X and distance(b.GeoData, bs.GeoData) < 300
```

- **Q3**. Which streets does bus No. 12 pass by?

```
SELECT r.Name
FROM get_infra(SpaceGendon, BUSROUTE) as br
     get_infra(SpaceGendon, ROAD) as r
WHERE br.Br_id = 12 and br.GeoData intersects r.GeoData
```

- **Q4**. Where can I switch between bus route No. 16 and No. 38?

```
SELECT bs1, bs2
FROM  get_infra(SpaceGendon, BUSSTOP) as bs1,
      get_infra(SpaceGendon, BUSSTOP) as bs2,
WHERE bs1.Br_id = 16 AND bs2.Br_id = 38 AND
      bs1.GeoData = bs2.GeoData
```

- **Q5**. At 8am on Monday, who sits in the bus No. 32?

```
SELECT mo.Name
FROM  get_infra(SpaceGendon, BUS) as bus,
      MOGendon as mo
WHERE ref_id(val(mo.Traj atinstant qt)) = bus.Bus_id and
      bus.Br_id = 32
```

- **Q6**. What is the percentage of people traveling by public transportation vehicles?

```
Let no_mo = SELECT COUNT(*) FROM MOGendon

Let no_submo = SELECT COUNT(*)
                FROM MOGendon as mo
                WHERE mo.Traj contains BUS or
                      mo.Traj contains METRO or
                      mo.Traj contains TAXI

SELECT DIV(no_submo, no_mo)
```

- **Q7**. Where and how long does *Bobby* walk during his trip?

```
SELECT mo.Traj at Walk
FROM MOGendon as mo
WHERE mo.Name = "Bobby"

SELECT duration(deftime(mo.Traj at Walk))
FROM MOGendon as mo
WHERE mo.Name = "Bobby"
```

- **Q8**. Find out all people passing room 312 at the office building between 9:00am and 11:00am on Monday.

```
SELECT mo.Name
FROM MOGendon as mo,
     get_infra(SpaceGendon, BUILDING) as b,
     get_infra(SpaceGendon, ROOM) as rm
WHERE b.Name = "Office-X" and b.B_id = rm.B_id and
      and rm.Room_id = 312 and
      ((mo.Traj at Indoor) atperiods qt) contains rm.Room_id
```

- **Q9**. Who arrived by taxi at the university on Friday?

  To arrive by taxi at the university means that the final location of the passenger within the taxi belongs to some driveway area close to the university. Such a part of the road network is also managed in the database as an object UniDriveway of type $\underline{line}$.

```
SELECT mo.Name
FROM MOGendon AS mo
WHERE  val(final((mo.Traj atperiods qt) at Taxi)))
       inside UniDriveway
```

- **Q10**. Who entered bus No. 3 at bus stop University on Tuesday afternoon?

```
SELECT mo.Name
FROM MOGendon AS mo,
     get_infra(SpaceGendon, BUSSTOP) AS bs,
     get_infra(SpaceGendon, BUS) AS bus
WHERE bs.Br_id = 3 AND bs.Name = "University" AND bus.Br_id = 3 AND
      bs.GeoData = freespace(val(initial((mo.Traj atperiods qt) at
                           theloc(bus.Bus_id, undef, undef)))))
```

- **Q11**. Find out all people walking through zone A and zone B on Saturday between 10am and 3pm.

```
SELECT mo.Name
FROM MOGendon AS mo,
     get_infra(SpaceGendon, RBO) AS R1,
     get_infra(SpaceGendon, RBO) AS R2
WHERE R1.Name = "Zone-A" AND R2.Name = "Zone-B" AND
      ((mo.Traj atperiods qt) at Walk) passes R1.Reg AND
      ((mo.Traj atperiods qt) at Walk) passes R2.Reg
```

- **Q12**. Did anyone who was on floor H-5 of the office building between 2pm and 5pm take a bus to the train station on Friday?

  To be on floor H-5 means to be in any of the rooms of floor H-5. We create a table associating rooms with their floors:

  Uni_Rel(B_id: int, Floor: string, Room_id: int)

```
SELECT mo.Name
FROM MOGendon AS mo,
     Uni_Rel AS u,
     get_infra(SpaceGendon, ROOM) AS r,
     get_infra(SpaceGendon, BUSSTOP) AS bs1,
     get_infra(SpaceGendon, BUSSTOP) AS bs2
WHERE u.Floor = "H-2" AND u.B_id = r.B_id AND u.Room_id = r.Room_id AND
      (mo.Traj atperiods qt) passes r.GeoData AND
      bs1.Name = "University" AND
      bs2.Name = "Main station" AND
      freespace(val(initial((mo.Traj atperiods qt) at Bus))) =
      bs1.GeoData AND
      freespace(val(final((mo.Traj atperiods qt) at Bus))) =
      bs2.GeoData
```

- **Q13**. Did bus No. 35 pass any bicycle traveler by on Monday?

  We define *pass* to mean that there exists a time instant that the distance between the two moving objects is less than 3 meters.

```
SELECT mo.Name
FROM  get_infra(SpaceGendon, BUS) AS bus,
      MOGendon AS mo
WHERE bus.Br_id = 35 AND
      sometimes(distance(freespace(bus.Traj atperiods qt),
                         freespace((mo.Traj atperiods qt) at Bicycle)) < 3.0)
```

- **Q14**. Did someone spend more than 15 minutes on waiting for the bus at the bus stop Cinema on Saturday?

```
SELECT mo.Name
FROM MOGendon AS mo,
     get_infra(SpaceGendon, BUSSTOP) AS bs
WHERE bs.Name = "Cinema" AND
      duration(deftime((mo.Traj atperiods qt) at bs.GeoData)) > 15
```

- **Q15**. How many people visit the cinema on Saturday?

```
SELECT COUNT(*)
FROM MOGendon AS mo,
     get_infra(SpaceGendon, BUILDING) AS b
WHERE b.Name = "Cinema" AND
      (mo.Traj atperiods qt) contains b.B_id
```

- **Q16**. Find out all people staying at room 154 in the university for more than one hour on Thursday.

```
SELECT mo.Name
FROM MOGendon AS mo,
     get_infra(SpaceGendon, BUILDING) AS b,
     get_infra(SpaceGendon, ROOM) AS r
WHERE b.Name = "University" AND b.B_id = r.B_id AND
      r.Room_id = 154 AND
      duration(deftime(mo.Traj atperiods qt) at
                       theloc(r.Room_id,undef,undef))) > 60
```

- **Q17**. Did someone spend more than one hour traveling by bus?

```
SELECT mo.Name
FROM MOGendon AS mo
WHERE duration(deftime(mo.Traj at Bus)) > 60
```

- **Q18**. Find out all people changing from bus No. A to bus No. B at stop X.

```
SELECT mo.Name
FROM MOGendon AS mo
     get_infra(SpaceGendon, BUSSTOP) as bs,
     get_infra(SpaceGendon, BUS) as bus1,
     get_infra(SpaceGendon, BUS) as bus2
WHERE  bus1.Br_id = A AND bus2.Br_id = B AND
       bs.Name = X AND
       freespace(val(final(mo.Traj at theloc(bus1.Bus_id)))) =
       freespace(val(initial(mo.Traj at theloc(bus2.Bus_id)))) AND
       freespace(val(final(mo.Traj at theloc(bus1.Bus_id)))) =
       bs.GeoData
```

- **Q19**. Find out all trips starting from zone A and ending in zone B by public transportation vehicles with the length of the walking path being less than 300 meters.

```
SELECT mo.Name
FROM MOGendon AS mo
    get_infra(SpaceGendon, RBO) AS R1,
    get_infra(SpaceGendon, RBO) AS R2
WHERE R1.Name = "ZoneA" AND R2.Name = "ZoneB" AND
      val(initial(mo.Traj)) inside R1.GeoData AND
      val(final(mo.Traj)) inside R2.GeoData AND
      ((mo.Traj contains Bus) OR (mo.Traj contains Metro) OR
      (mo.Traj contains Taxi)) AND
      length(trajectory(mo.Traj at Walk)) < 300
```

- **Q20.** Find the top k bus (metro) routes with high passenger flow for all workdays.

```
SELECT TOP k *
FROM
  SELECT bus.Br_id, COUNT(*) AS NO
  FROM MOGendon AS mo
       get_infra(SpaceGendon, BUS) AS bus
```

```
  WHERE mo.Traj contains Bus AND
      ((mo.Traj atperiods qt) at Bus)) contains bus.Bus_id
  GROUP BY bus.Br_id
  ORDER BY NO DESC
```

- **Q21.** Find the top k road segments with high traffic during the rush hour for all workdays.

  To answer such a query, we create a relation storing all road segments with the schema

  Rel_RoadSeg: (S_id: int, GeoData: line, R_id: int)

  where S_id is the unique segment id, GeoData stores the geometrical property and R_id indicates the id for the road that the segment belongs to. For each road, we decompose it into a set of pieces at the junction positions and each piece is stored as a tuple in Rel_RoadSeg.

  First, we get the traffic of each road segment by aggregrating the number of buses passing by.

```
LET Rel_BRCOUNT =
    SELECT br.Br_id, br.GeoData, COUNT(*) AS NO
    FROM get_infra(SpaceGendon, BUSROUTE) as br,
         get_infra(SpaceGendon, BUS) as bus
    WHERE deftime(bus.Traj) intersects qt AND bus.Br_id = br.Br_id
    GROUP BY br.Br_id
```

  Each tuple in Rel_BRCOUNT indicates the number of trips for each bus route.

```
LET Rel_RES1 =
    SELECT s.S_id, SUM(br.NO) AS FLOW
    FROM Rel_BRCOUNT as br,
         Rel_RoadSeg as s
    WHERE br.GeoData intersects s.GeoData
    GROUP BY s.S_id
```

  We perform the join on bus routes and road segments to get the segments that each bus route maps to. Then, we group the tuple by segment id and call the aggregation function to get the total number of buses passing a segment from different routes.

  Second, we get the traffic from moving objects that contain one of the modes $\{Car, Taxi, Bike\}$. At first, we collect the paths of these trips.

```
LET Rel_Traj_C =
    SELECT trajectory(mo.Traj at CAR) AS Path
    FROM MOGendon AS mo
    WHERE deftime(mo.Traj) intersects qt

LET Rel_Traj_T =
    SELECT trajectory(mo.Traj at TAXI) AS Path
    FROM MOGendon AS mo
    WHERE deftime(mo.Traj) intersects qt

LET Rel_Traj_B =
    SELECT trajectory(mo.Traj at BIKE) AS Path
    FROM MOGendon AS mo
    WHERE deftime(mo.Traj) intersects qt

LET Rel_Traj = Rel_Traj_C union Rel_Traj_T union Rel_Traj_B
```

Then, we do the join on the paths and road segments to get the traffic value.

```
LET Rel_RES2 =
    SELECT s.S_id, COUNT(*) AS FLOW
    FROM Rel_Traj AS p,
         Rel_RoadSeg AS s
    WHERE p.Path intersects s.GeoData
    GROUP BY s.S_id
```

Third, we merge the two traffic relations Res_RES1 and Res_RES2 to get the final result.

```
SELECT TOP k *
FROM
  SELECT r1.S_id, r1.FLOW + r2.FLOW as C
  FROM Rel_RES1 as r1,
       Rel_RES2 as r2
  WHERE r1.S_id = r2.S_id
  ORDER BY C DESC
```

Unique IFOB id and Reference id for an indoor location. In order to have a unique id for each IFOB, we define a set of disjoint ranges each of which is used for one infrastructure, e.g., [1, 5000] for I_{rn}, [6000, 7000] for I_{bn}. Given a generic location $gl \in D_{\underline{genloc}}$, the meaning of $gl.oid$ is clear if the IFOB belongs to an outdoor infrastructure. However, for the indoor environment different buildings can have the same room id (e.g., ROOM NO 12, ROOM NO 35), only using the building id cannot uniquely identify an indoor location. To solve the problem, $gl.oid$ is set by combining a building id and a room id for an indoor location. We introduce how to achieve the goal in the following. For the sake of clear presentation, we use two strings B_Id and R_Id to denote a building id and a room id, respectively. Suppose that gl is located in the room R_Id = "123" of a building B_Id = "167654", then $gl.oid$ is assigned by the concatenation of B_Id and R_Id, that is "167654123". Let $len(B\_Id)$ return the length of a string for B_Id. The range for all building ids is carefully chosen in order to fulfill the condition $len(B\_Id_{min})$ = $len(B\_Id_{max})$, i.e., the number of digits is the same for each id. Otherwise, an ambiguous case can occur. Afterwards, $len(B\_Id)$ is a fixed value (new building construction is not considered) and we can extract B_Id and R_Id from an indoor location. Using the example before, we can get B_Id = "167654" and R_Id = "123" from $gl.oid$ = "167654123".

Appendix B - Experimental Statistics

Query	Berlin10k	Berlin20k	Berlin50k	Berlin100k	Berlin200k	Berlin500k
Q5-B	0.2	0.3	0.5	0.9	1.8	4.2
Q5-M	0.1	0.2	0.5	0.9	1.7	4.2
Q6	0.1	0.3	0.7	1.4	2.7	7.0
Q7	0.2	0.4	0.8	1.5	2.9	7.3
Q8	0.2	0.3	0.7	1.3	2.5	6.6
Q9	0.7	0.8	1.1	1.6	2.6	5.5
Q10	0.8	1.0	1.7	3.0	5.4	12.7
Q11	0.6	1.2	3.2	6.5	14.3	40.9
Q12	0.7	0.9	1.3	2.0	3.4	7.5
Q13	1.4	1.6	2.0	2.7	4.2	8.4
Q14	0.4	0.6	1.2	2.1	4.1	10.0
Q15	0.3	0.5	1.0	2.1	4.3	11.3
Q16	0.2	0.4	0.8	1.5	3.0	8.0
Q17-B	0.2	0.4	0.9	1.9	3.8	10.1
Q17-M	0.2	0.3	0.8	1.6	3.2	8.4
Q18-B	0.7	0.9	1.6	2.8	5.0	12.1
Q18-M	0.7	1.2	2.4	4.4	8.7	21.3
Q19	0.1	0.3	0.8	2.3	4.4	10.5
Q20-B	0.4	0.7	1.4	2.6	5.0	12.7
Q20-M	0.3	0.5	1.0	1.9	3.8	9.5
Q21	19.7	20.0	21.5	23.4	27.8	43.0
Total	28.3	32.5	45.9	68.3	114.4	261.0

(a) CPU Time (sec)

Query	Berlin10k	Berlin20k	Berlin50k	Berlin100k	Berlin200k	Berlin500k
Q5-B	2.0	3.0	5.8	8.4	19.9	48.0
Q5-M	0.4	0.5	0.9	1.5	7.4	46.9
Q6	0.001	0.001	0.001	0.001	0.5	40.5
Q7	0.03	0.03	0.3	0.3	0.08	81.4
Q8	2.1	4.0	10.0	20.4	54.6	149.3
Q9	1.6	1.7	2.1	8.4	19.4	47.3
Q10	1.5	2.0	3.5	8.6	23.8	63.6
Q11	2.6	5.3	13.3	27.9	64.0	170.8
Q12	1.0	1.6	5.2	14.0	27.8	66.8
Q13	1.4	1.6	2.7	6.4	18.8	51.7
Q14	0.3	0.6	1.5	9.1	24.2	79.6
Q15	1.5	3.2	10.1	35.4	90.4	232.0
Q16	3.9	7.8	21.5	46.8	93.0	232.6
Q17-B	1.8	5.3	17.7	45.3	90.7	227.8
Q17-M	2.1	4.7	15.4	35.9	72.1	179.4
Q18-B	1.0	1.9	5.1	16.1	32.9	81.0
Q18-M	1.8	2.0	2.1	9.8	28.7	71.2
Q19	4.9	9.9	24.9	51.8	109.6	285.1
Q20-B	0.1	6.1	22.0	45.0	88.8	221.4
Q20-M	0.03	4.4	15.2	34.8	69.7	173.2
Q21	1.2	21.2	50.2	97.2	190.6	470.5
Total	41.8	86.8	229.1	522.5	1126.7	3020.2

(b) I/O Accesses(k)

Figure 11: Scaling Evaluation Results

Query	TO	T
Q5-B	4.2	5.3
Q5-M	4.2	5.2
Q6	7.0	48.9
Q7	7.3	7.2
Q8	6.6	13.1
Q9	5.5	9.1
Q10	12.7	14.5
Q11	40.9	41.9
Q12	7.5	9.6
Q13	8.4	46.6
Q14	10.0	16.2

(a) CPU Time (sec)

Query	TO	T
Q15	11.3	30.9
Q16	8.0	20.6
Q17-B	10.1	32.0
Q17-M	8.4	31.3
Q18-B	12.1	15.8
Q18-M	21.3	24.8
Q19	10.5	14.8
Q20-B	12.7	34.1
Q20-M	9.5	31.6
Q21	43.0	48.7

(b) CPU Time (sec)

Query	AO	A
Q5-B	48.0	88.6
Q5-M	46.9	87.8
Q6	40.5	940.9
Q7	81.4	80.4
Q8	149.2	136.1
Q9	47.3	185.2
Q10	63.6	129.1
Q11	170.8	157.8
Q12	66.8	133.3
Q13	51.7	226.1
Q14	79.6	250.3

(c) I/O Accesses (k)

Query	AO	A
Q15	232.0	250.2
Q16	232.6	217.7
Q17-B	227.8	933.0
Q17-M	179.4	933.0
Q18-B	81.0	223.0
Q18-M	71.2	222.1
Q19	285.1	897.2
Q20-B	221.4	915.2
Q20-M	173.2	914.3
Q21	470.5	601.5

(d) I/O Accesses (k)

Figure 12: Results of Berlin500K

Query	TO	T
Q5-B	4.1	5.1
Q5-M	4.3	5.1
Q6	7.0	51.2
Q7	7.4	7.4
Q8	6.8	13.4
Q9	5.8	9.3
Q10	11.3	13.0
Q11	24.2	24.9
Q12	7.0	9.1
Q13	98.8	822.8
Q14	6.6	12.1

(a) CPU Time (sec)

Query	TO	T
Q15	12.4	37.7
Q16	7.9	20.7
Q17-B	9.1	30.5
Q17-M	8.2	28.8
Q18-B	10.2	13.9
Q18-M	18.0	21.5
Q19	10.0	21.8
Q20-B	10.8	33.2
Q20-M	8.8	28.7
Q21	293.3	295.2

(b) CPU Time (sec)

Query	AO	A
Q5-B	48.2	88.7
Q5-M	47.2	84.5
Q6	41.1	961.7
Q7	82.1	81.5
Q8	146.0	134.1
Q9	52.3	190.9
Q10	63.4	156.7
Q11	164.4	154.5
Q12	67.9	166.2
Q13	77.5	277.1
Q14	66.0	243.2

(c) I/O Accesses (k)

Query	AO	A
Q15	223.3	243.1
Q16	236.1	228.6
Q17-B	169.8	953.3
Q17-M	211.8	953.3
Q18-B	80.7	238.2
Q18-M	79.1	228.4
Q19	522.2	2514.0
Q20-B	169.8	936.6
Q20-M	202.5	932.4
Q21	631.3	756.6

(d) I/O Accesses (k)

Figure 13: Results of Houston500K

Verzeichnis der zuletzt erschienenen Informatik-Bericht

[345] vor der Brück, T.; Helbig, H.; Leveling, J.:
The Readability Checker Delite Technical Report

[346] vor der Brück, T.:
Application of Machine Learning Algorithms for Automatic Knowledge Acquisition and Readability Analysis Technical Report

[347] Fechner, B.:
Dynamische Fehlererkennungs- und –behebungsmechanismen für verlässliche Mikroprozessoren

[348] Brattka, V., Dillhage, R., Grubba, T., Klutsch, A.:
CCA 2008 - Fifth International Conference on Computability and Complexity in Analysis

[349] Osterloh, A.:
A Lower Bound for Oblivious Dimensional Routing

[350] Osterloh, A., Keller, J.:
Das GCA-Modell im Vergleich zum PRAM-Modell

[351] Fechner, B.:
GPUs for Dependability

[352] Güting, R. H., Behr, T., Xu, J.:
Efficient *k*-Nearest Neighbor Search on Moving Object Trajectories

[353] Bauer, A., Dillhage, R., Hertling, P., Ko K.I., Rettinger, R.:
CCA 2009 Sixth International Conference on Computability and Complexity in Analysis

[354] Beierle, C., Kern-Isberner, G.
Relational Approaches to Knowledge Representation and Learning

[355] Sakr, M.A., Güting, R.H.
Spatiotemporal Pattern Queries

[356] Güting, R. H., Behr, T., Düntgen, C.:
SECONDO: A Platform for Moving Objects Database Research and for Publishing and Integrating Research Implementations

[357] Düntgen, C., Behr, T., Güting, R.H.:
Assessing Representations for Moving Object Histories

[358] Sakr, M.A., Güting, R.H.
Group Spatiotemporal Pattern Queries

[359] Hartrumpf, S., Helbig, H., vor der Brück, T. , Eichhorn, C.
SemDupl: Semantic Based Duplicate Identification

[360] Xu, J., Güting, R.H.
A Generic Data Model for Moving Objects

[361] Beierle, C., Kern-Isberner, G.
Evolving Knowledge in Theory and Application: 3rd Workshop on Dynamics of Knowledge and Belief, DKB 2011