نشریه علمی- پژوهشی علوم و فنون نقشه برداری، دوره سوم، شماره ۲، آبان ماه ۱۳۹۲

ارزیابی مدل‌های داده والگوریتم‌ها در پایگاه‌های داده اشیاء متحرک

فرشاد حکیم پور[۱]، فریدالدین چراغی*[۲]، جمشید مالکی[۲]

[۱]استادیار گروه مهندسی نقشه‌برداری- پردیس دانشکده‌های فنی- دانشگاه تهران
fhakimpour@ut.ac.ir

دانش آموخته کارشناسی ارشد سیستم‌های اطلاعات مکانی- گروه مهندسی نقشه‌برداری- پردیس دانشکده‌های فنی - دانشگاه تهران
{j.maleki, fcheraghi}@ut.ac.ir

(تاریخ دریافت فروردین ۱۳۹۱، تاریخ تصویب اردیبهشت ۱۳۹۲)

چکیده

این مقاله با تمرکز روی رویکرد تاریخچه‌ی حرکت در پایگاه‌های داده اشیاء متحرک، به ارزیابی ساختار و مدل داده پرداخته و سپس الگوریتم‌ها به همراه پیچیدگی هر کدام، به طور جداگانه بحث گردیده‌اند. همچنین با معرفی معیارها به عنوان ابزاری مناسب برای مقایسه پایگاه‌های داده اشیاء متحرک، از پرس‌وجوها و داده‌های استفاده شده در معیار BerlinMod، برای ارزیابی و انجام تحلیل‌ها استفاده شده است. نهایتاً به ارائه‌ی روش‌های افزایش عملکرد برای هر پرس‌وجو پرداخته شده است.

واژگان کلیدی: ارزیابی پایگاه‌های داده اشیاء متحرک، مدل داده، پردازش پرس‌وجو، الگوریتم‌های اشیاء متحرک، پالایش افراز.

* نویسنده رابط

۱- مقدمه

پایگاه‌های داده فعلی قادر به ذخیره، بازیابی و مدیریت حجم زیادی داده هستند. علاوه بر داده‌های استاندارد، می‌توان داده‌های پیچیده مانند، داده‌های چند رسانه‌ای (همچون فیلم و تصویر)، داده‌های مکانی و داده‌های مکانی-زمانی را نیز در آن‌ها ذخیره نمود. در دهه‌ی اخیر ردیابی موقعیت اشیاء متحرک، با استفاده از دستگاه‌های GPS که کوچکتر، ارزانتر و دقیق تر شده‌اند، تسهیل گشته است. محبوبیت این ابزار تعیین موقعیت به عاملی قوی برای توسعه‌ی کاربردهای جدید مثل، سرویس‌های مکان مبنا[۱] شده است. هر چند جمع آوری داده های اشیاء متحرک تسهیل یافته است، مدل سازی و پردازش آن‌ها در پایگاه داده یک موضوع در حال تحقیق در حوزه‌های علمی و پژوهشی می‌باشد.

به طور کلی پایگاه‌های داده اشیاء متحرک (MODs[۲]) به دو دسته تقسیم می‌شوند: ۱) برای مدل سازی اشیاء متحرک در زمان حال، مثل سیستم‌های مدیریت ناوگان، که در آن پرس و جوهایی در ارتباط با موقعیت، در زمان حال و آینده‌ی نزدیک قابل پاسخ است. ۲) برای مدل کردن تاریخچه‌ی کامل حرکت، که می‌توان تحلیل‌های پیچیده‌ای را روی حرکت شیء در گذشته و حال انجام داد. در این مقاله تمرکز برورویکرد دوم می‌باشد. پرس‌وجوهای مختلفی که در این مدل قابل پاسخگویی است، شامل موارد زیر است: شی o در لحظه‌ی t در کجا قرار داشت؟ در چه زمانی شیء o یک ویژگی (ویژگی‌خاص) مشخص دارد؟ کدام اشیاء متحرک در نقطه P به هم رسیدند؟ در رویکرد تاریخچه‌ی حرکت، خط‌سیر اشیاء متحرک در دو حالت مدل می گردد. یکی مدل سازی حرکت شیء در شبکه‌ی راه‌ها که به آن حرکت مقید[۳] می‌گویند، دیگری حرکت در فضای اقلیدسی، که در آن حرکت شیء بدون محدودیت مکانی می باشد. به حرکت از نوع دوم، حرکت نامقید یا حرکت آزاد[۴] می‌گویند. هر چند در اینجا تمرکز بر روی داده‌های مکانی-زمانی با حرکت خطی و نامقید[۵]می‌باشد، ولی این مدل داده‌ی عمومی، قابل استفاده برای داده‌های دیگر، مثل داده‌های مکانی-زمانی مقید (مثلا در شبکه راه‌ها)، حرکت‌های غیرخطی (مثلا سهمی) و داده‌های غیرمکانی ولی زمانمند نیز می‌باشد.

۲- کارهای مرتبط

Pelekis و همکاران مروری کامل بر کارهای مشابه [۱] انجام داده‌اند که در آن‌ها تاریخچه‌ی پیشرفت مدل‌های داده مکانی-زمانی مورد بحث قرار گرفته است.

MOST[۶] به همراه FTL[۷] [۸] مدلی برای پاسخ‌گویی به پرس‌وجوهای زمان حال و آینده نزدیک هستند، که عموما به آن مدل ردیابی[۸] [۹–۱۲] می‌گویند. در این مدل حرکت توسط ویژگی‌خاص پویا[۹] مدل می‌شود که بطور پیوسته و تابعی از زمان تغییر می‌کند. هر تابع متشکل از یک نقطه (point) به همراه یک بردارحرکت می‌باشد که نقطه، آخرین موقعیت معلوم شیء و بردار حرکت، آخرین جهت حرکت و سرعت آن را مشخص می‌کند و توانایی انجام پرس‌وجوها در آینده نزدیک را می‌دهد. این مدل به دلیل اینکه تاریخچه‌ی حرکت را ذخیره نمی‌کند، نمی-تواند پرس‌وجوهای مربوط به گذشته‌ی شیء متحرک را پاسخ دهد. این مدل صرفا جهت ارجاع به مدل‌های دیگر برای اشیاء متحرک و تعیین محدوده‌ی تحقیق آورده شده است.

رویکرد اول در ثبت تغییرات زمانمند اشکال هندسی، به کارگیری مدل زمانی برای داده‌های مکانی بود. در [۲],[۳] نویسندگان از snapshots برای نمایش داده‌های مکانی در لحظه‌های مشخص استفاده کرده‌اند. در مدل snapshots کل اطلاعات مکانی در یک لحظه‌ی خاص با یک برچسب زمانی ثبت می‌شود. این روش علاوه بر تکرار داده‌هایی که در زمان تغییر نمی‌کنند، از عمکرد بسیار پایینی برخوردار می‌باشد. به عبارتی این مدل داده، نه تنها قادر به پاسخگویی پرس‌وجوهای پیوسته نمی‌باشد، بلکه

[۱] Location Based Services
[۲] Moving Object Databases
[۳] Constrained Movement
[۴] Free Space Movement

[۵] Unconstrained Linear Time-Sliced Spatio-Temporal Movement
[۶] Moving Objects Spatio-Temporal Model
[۷] Future Temporal Language
[۸] Tracking
[۹] Dynamic Attribute

به دلیل افزونگی داده، منجر به کندی پرس‌وجوهای دیگر می‌شود. در بعدها از time-stamping، برای ذخیره تغییرات گسسته یک شیء مکانی (نقطه، خط، سطح و...) در لحظه‌ی رویدادها و تغییرات یک شیء هندسی مفرد استفاده گردید [۴]-[۷]: در آن، تاریخچه‌ی حرکت بصورت جفت‌های (l,i) ، i لحظه‌ی مشاهده و l مشاهده، که می‌تواند یک شکل هندسی باشد، مدل می‌شود. مشکل عمده این رویکرد گسسته بودن آن است که درونیابی و برونیابی در آن ممکن نیست. این مشکل باعث عدم تعادل در دقت (تعداد زیادی مشاهده) و منابع سیستمی شده که منجر به کندی پردازش پرس‌وجوها می‌شود.

اولین مدل انتزاعی[۱] که تاریخچه‌ی حرکت یک شیء متحرک را بصورت پیوسته و تابعی از زمان مدل نمود در [۱۳] ارائه شده است. در [۱۴]، مدل گسسته‌ای[۲] که قابل پیاده‌سازی درپایگاه داده می‌باشد، برای مدل انتزاعی تاریخچه‌ی حرکت ارائه گردید. در آن از نمایش قطعه‌ای[۳] استفاده گردید و تغییرات پیوسته شیء با استفاده از توابع ساده (مثلا خطی) مدل شده‌اند. نهایتاً الگوریتم‌ها برای عملگرهای مکانی-زمانی در [۱۵] معرفی شدند و در [۱۶] با معرفی summary-fields کامل شدند. این مدل جامع، که مدل مورد ارزیابی در طول این مقاله می‌باشد، علاوه بر درون‌یابی در نقاط بین مشاهدات، از عملکرد بالاتری نسبت به مدل‌های داده‌ی ماقبل خود برخوردار است.

از سیستم‌هایی که در آن‌ها مدل گسسته‌ی تاریخچه‌ی حرکت پیاده شده است،SECONDO[۱۷–۱۹]می‌باشد. SECONDOیک پایگاه داده آزمایشگاهی رایگان و متن باز می‌باشد که قابلیت اضافه کردن جبرهای مختلف در آن وجود دارد. در حال حاضر نزدیک به بیست جبر مختلف، به همراه جبر اشیاء متحرک در آن پیاده سازی شده است. علاوه بر آن، در [۲۰–۲۲] نویسندگان به معرفی HERMES می‌پردازند که مدل تاریخچه‌ی حرکت در آن پیاده‌سازی شده است. HERMES از نمایش قطعه‌ای استفاده می‌کند و علاوه بر توابع خطی و توابع ثابت از توابع چندجمله‌ای مرتبه دوم و توابع رادیکالی چندجمله‌ای مرتبه دوم پشتیبانی می‌کند. درHERMES، که در پایگاه داده اوراکل طراحی و پیاده سازی شده است، هر نقطه‌ی متحرک در یک جدول، اشاره‌گری به یک جدول دیگر[۴]می‌باشد که از تعدادی unit_moving_point تشکیل شده است. هر unit_moving_point نیز از یک بازه‌ی زمانی و یک unit_function[۵] تشکیل شده است. ساختار و معماری HERMES، علاوه بر اوراکل، در پایگاه داده PostgresSQL پیاده سازی شده است [۲۳]. در [۲۴] نویسندگان از SECODNO برای طراحی و پیاده-سازی یک معیار[۶]، به منظور ارزیابی مدل داده تاریخچه‌ی حرکت، استفاده نمودند. در این مقاله از پرس‌وجوها و داده‌های تولیدشده در این معیار استفاده شده است.

در غالب سیستم‌ها، درون‌یابی استفاده شده خطی می‌باشد. دلیل این امر، کم بودن پیچیدگی الگوریتم‌ها و بار محاسباتی توابع و عملگرهای مربوطه می‌باشد. Becker و همکاران در [۲۵]، رویکردی برای درون‌یابی غیر خطی (B-Spline)، ارائه نمودند. آزمایشات مختلف نشان داد که استفاده از این تابع برای درون‌یابی (در مقایسه با توابع خطی) هزینه‌ی محاسبات را ۵۰ تا ۳۰ درصد افزایش می‌دهد. لذا در پیاده‌سازی‌های انجام شده در این مقاله، از درون‌یابی خطی استفاده شده است. روش‌های مشابهی به نمایش دقیق تر خط سیر[۷] و کاهش حجم داده و در نتیجه افزایش عملکرد در [۲۶] معرفی شدند.

۳- ساختار داده برای مدل تاریخچه حرکت

در طراحی و پیاده سازی مدل داده حدالامکان سعی شده است از نوع داده‌های پایه‌ی پایگاه داده استفاده شود. سه دسته کلی نوع داده پایه، زمانمند و مکانی در پایگاه‌های داده موجود می‌باشد. نوع داده‌های پایه شامل integer، string و مانند آنها هستند؛ نوع داده زمانمند شامل انواعی مانند: date و timestamp هستند و از انواع مکانی در اوراکل می‌توان sdo_geometry و sdo_tin را نام برد. نوع داده‌های جدید مربوط به اشیاء متحرک که از نوع

---

[۱] Abstract Model
[۲] Discrete Model
[۳] Sliced Representation

[۴] در اوراکل جداولی که از یک جدول به آنها ارجاع داده شده است، Nested Table نام دارند.

[۵] از دو یا سه مختصات ابتدا و انتها و پارامتری برای تعیین روش درون‌یابی، تشکیل شده است.

[۶] Benchmark
[۷] Trajectory

داده‌های اصلی[1] تشکیل شده‌اند به انواع داده‌ها در پایگاه‌داده اضافه شده‌اند. در جدول ۱ لیست تمامی نوع داده‌های جدید ملاحظه می‌شود.

بطور کلی چهار نوع بازه‌ی زمانی وجود دارد، یعنی بازه‌ی باز-باز ( )، بازه‌ی باز-بسته ( ] ، بازه‌ی بسته-باز [ ) و بازه‌ی بسته-بسته [ ] که بازه‌ی استفاده شده در time_period از نوع سوم، یعنی بسته-باز می‌باشد. با این نوع بازه، با توجه به گسسته بودن زمان، می‌توان به بازه‌های دیگر نیز دست یافت.

جدول ۱- انواع داده‌های پیاده شده درپایگاه داده

نوع داده	توضیحات
Coords	از دو مختصات x,y تشکیل شده است.
time_point	نوع داده timestamp در پایگاه داده را توسعه می‌دهد و خواص دانه بندی (دقت) را به آن اضافه می‌کند.
time_period	از دو time_point تشکیل شده است که یک بازه زمانی مطلق را نمایش می‌دهد.
temp_element	از تعدادی time_period تشکیل شده است که هر دو بازه‌ی زمانی مجاور نباید با هم همپوشانی داشته باشند (disgoint باشند). (شکل ۱)
unit_function	برای مدل کردن نوع حرکت در یک بازه زمانی به کار می‌رود. مثلا اگر حرکت خطی باشد از چهار مختصات،یعنی x,yی ابتدا و انتها و یک flag که تعیین می‌کند حرکت خطی است تشکیل شده است. مقادیر قابل قبول برای flag می‌تواند تابع ثابت و تابع درجه دوم باشد. (شکل ۲)
umbool	این نوع داده از یک bool و یک بازه‌ی زمانی تشکیل شده است.
umreal	این نوع داده از چهار پارامتر a,b,c,r و یک بازه‌ی زمانی تشکیل شده، که سه پارامتر اول ضرایب چندجمله‌ای مرتبه اول و r یک bool می‌باشد که اگر ۰ باشد، تابع چندجمله‌ای مرتبه اول و اگر ۱ باشد ریشه آن می‌باشد.
umpoint	از یک unit_function و یک time_period تشکیل شده است.
mbool	از یک جدول داخلی از umbool و تعدادی عملگر تشکیل شده است.
mreal	از یک جدول داخلی از umreal و تعدادی عملگر تشکیل شده است.
mpoint	از یک جدول داخلی از umpoint و تعدادی عملگر تشکیل شده است.

مثلاً بازه‌ی [t1,t2] تبدیل به [t1,t2+1)می‌شود (دانه-بندی در مدل زمان ثانیه است).

نوع داده‌ی temp_element نیز از تعدادی بازه‌ی زمانی تشکیل می‌شود که نباید با هم، هم پوشانی داشته باشند. از این نوع داده در تابع پراستفاده‌ی پالایش افرازاستفاده شده است. نوع داده unit_function نیز با انواع درون‌یابی‌ها در (شکل ۲) قابل ملاحظه می‌باشد.

۴- الگوریتم‌های پیاده شده در پایگاه داده

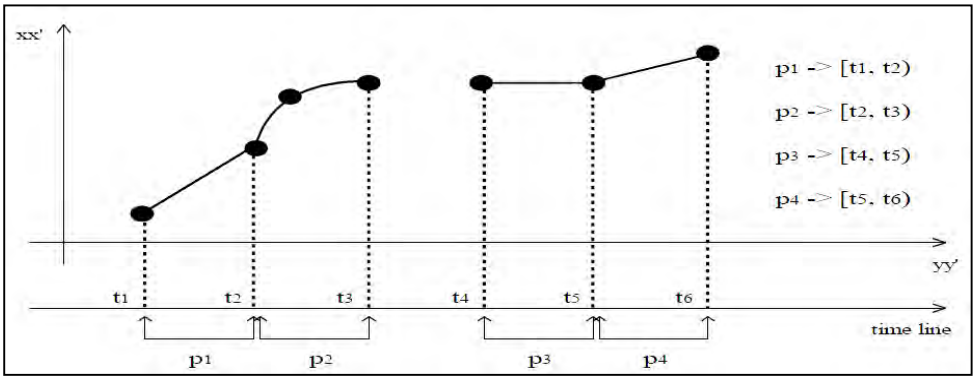
برای آزمایش و ارزیابی عوامل موثر بر روی عملکرد سیستم تعدادی از عملگرها و توابع در پایگاه داده (اوراکل) پیاده‌سازی شدند. توابع انتخاب شده براساس پرس‌وجوهایی می‌باشند که در معیار BerlinMod از آن‌ها استفاده شده است. در (جدول ۲) لیست این توابع و عملگرها قابل مشاهده است. برای ارائه‌ی عملکرد تئوری الگوریتم‌ها از تابع O بزرگ[2] استفاده می‌شود [۲۷]. در علوم کامپیوتر، O(n) برای بیان کارایی یک الگوریتم از نظر زمان پردازش و یا حافظه‌ی مورد نیاز با توجه به اندازه‌ی مسئله n (مثلا تعداد موارد)، می‌باشد. این تابع در واقع، بار محاسباتی الگوریتم را بر اساس سرعت رشد داده‌ی ورودی اندازه‌گیری می‌کند. برای مثال برای مسئله‌ی جمع n عدد طبیعی، پیچیدگی n2 می‌باشد. زیرا:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

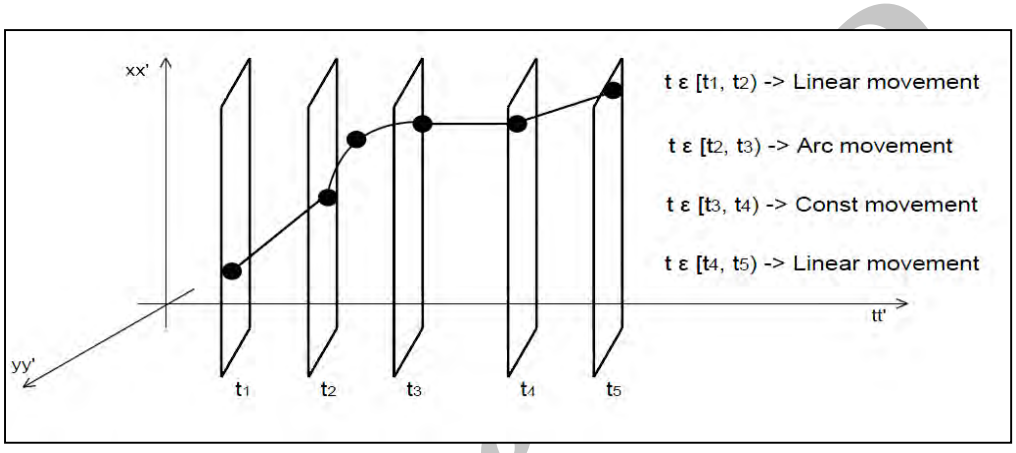
بالاترین درجه در این سری، n2 می باشد. لذا O(n2)، پیچیدگی آن می باشد. عملکرد توابع (جدول۲) در ادامه این بخش تشریح و مورد بحث قرار می‌گیرند.

[1] در این مقاله، برای سادگی، نوع داده‌های پایه، زمانمند و مکانی، نوع داده‌های اصلی نامیده شده‌اند.

[2] Big O Notation



شکل ۱- یک عنصرزمانی (Temporal Element)



شکل ۲- انواع درونیابی در Unit_functionها

۴-۱- پالایش افراز[۱]

در ادامه به بررسی الگوریتم‌های استفاده شده در عملگرها و محاسبه‌ی پیچیدگی برای هر کدام از آنها می‌پردازیم. در بسیاری از توابعی که ورودی آنها دو شیء متحرک است از عملگری به نام partition_refinement استفاده می‌شود. این عملگر، اشتراک دامنه‌ی زمانی دو شیء متحرک را محاسبه می‌کند. یک افراز پالایش شده از خرد کردن یک unit به unitهای کوچکتر ایجاد می‌گردد (شکل ۳).

نهایتاً تعداد unitهای دو شیء متحرک p خواهد بود. این عملیات با پویش موازی unitهای دو شیء متحرک انجام می‌شود. پیچیدگی این عملیات O(n+m) خواهد بود. توابعی که از این عملگر استفاده می‌کنند، خروجی شان یک شیء متحرک خواهد بود

[۱] Partition Refinement

[۲] Partition Refinement

جدول ۲- عملگرها و توابع استفاده شده در پرس‌وجوها

نام تابع/عملگر	امضاء تابع	توضیحات
at_instant	mpoint × instant → point	نقطه‌ی متحرک را در لحظه‌ی داده شده محاسبه می‌کند
present	mpoint × instant → bool	آیا لحظه‌ی داده شده در بازه تعریف نقطه متحرک می‌باشد
passes	mpoint × sdo_geometry (point or polygon) → bool	آیا نقطه‌ی متحرک از نقطه یا سطح عبور می‌کند
trajectory	mpoint → sdo_geometry (line or multi_line)	خط سیر نقطه‌ی متحرک را برمی‌گرداند
initial	mpoint → point	نقطه‌ی ابتدایی نقطه‌ی متحرک در لحظه‌ی شروع را برمی‌گرداند
intersection	mpoint × sdo_geometry (point or polygon) → mpoint	نقطه‌ی متحرک را به نقطه یا سطح داده شده محدود می‌کند
intersection	mpoint × mpoint → mpoint	تقاطع دو نقطه‌ی متحرک در بازه‌های زمانی مشترک آن دو را می‌دهد
f_distance	mpoint × mpoint → mreal	فاصله‌ی بین دو شیء متحرک در بازه‌های زمانی مشترک آنها می‌دهد
f_length	mpoint → real	مسافت طی شده‌ی نقطه‌ی متحرک را می‌دهد
at_temp_element	mpoint × temp_element → mpoint	نقطه‌ی متحرک را به بازه‌های زمانی داده شده محدود می‌کند
temp_element	mpoint → temp_element	عنصر زمانی نقطه‌ی متحرک را می‌دهد
sometimes	mbool → bool	اگر moving_bool حتی یکبار TRUE شود این تابع TRUE برمی‌گرداند
isLessThan_orEqual	mreal × real → mbool	خروجی یک moving_bool با بازه‌های زمانی یکسان با moving_real است. در بازه‌هایی که از real کوچکتر یا مساوی mreal است، مقدار TRUE و بقیه‌ی جاها FALSE.
concat	mpoint × mpoint → mpoint	اتصال تاریخچه‌ی حرکت دو شیء متحرک (دامنه‌ی زمانی دو شیء نباید اشتراک داشته باشد)

۴-۲- رویکرد فیلتری

در اکثر الگوریتم‌ها یک مرحله‌ی ابتدایی فیلتر وجود دارد که با استفاده از summery-fields[۱] گزینه‌های جستجو را کاهش می دهد. برای مثال در پرس‌وجوهای اتصال ابتدا به مقایسه‌ی MBRها پرداخته شده و سپس به جستجوی دقیق‌تر برای رسیدن به جواب نهایی پرداخته می‌شود. مرحله‌ی اول را فیلترینگ و مرحله‌ی دوم را پالایش می‌نامند. در ادامه نحوه عملکرد به همراه پیچیدگی هر الگوریتم آمده‌است.

۴-۳- الگوریتم‌های بکار رفته در توابع

تابع at_instant ابتدا یک time_point را در unitهای mpoint جستجوی دودویی می‌کند و در صورتی که یک unit آن را شامل شود، مقادیر آن در لحظه‌ی time_point محاسبه می‌شود. پیچیدگی این عملیات O(log(m)) می‌باشد.

تابع present مانند تابع فوق با همان پیچیدگی عمل می‌کند، با این تفاوت که نیازی به محاسبه‌ی تابع در لحظه‌ی time_point نمی‌باشد. پیچیدگی در بدترین حالت O(log(m)) می‌باشد. تابع initial نیز تابع at_instant را در لحظه‌ی شروع دامنه‌ی زمانی شیء متحرک می‌دهد.

عملگر temp_element (شکل ۱) عنصر زمانی یک شیء متحرک را استخراج می‌کند. به این صورت که با بررسی هر unit، بازه‌ی زمانی آن را در آرایه‌ای از بازه‌های زمانی قرار می‌دهد. پیچیدگی آن O(m) است.

عملگر trajectory خط سیر یک نقطه‌ی متحرک را برمی‌گرداند. تمامی unitها به ترتیب بررسی می‌شود، اگر مختصات انتهایی یک unit با مختصات ابتدایی unit بعدی یکی شود، یک خط (polyline) از اتصال این دو unit ایجاد شده و در غیر این صورت خط تبدیل به یک multiline می‌شود. پیچیدگی این عملگر O(m) می‌باشد.

[۱] مثلا برای یک moving_real مقادیر مینیمم و ماکزیمم هر unit و کل moving_real محسوب می‌شود.

عملگر length برای محاسبه‌ی مسافت طی شده توسط شیء متحرک به کار گرفته می‌شود و برای محاسبه آن لازم است ابتدا خط سیر یک متحرک توسط تابع trajectory محاسبه شده و سپس از توابع پایه پایگاه داده (مثلا در اوراکل sdo_geom.sdo_length) مسافت طی شده را بدست آورد. پیچیدگی مرحله‌ی اول O(m) و تابع طول که از جمع طول‌های هر قطعه خط بدست می‌آید (دو مختصات پی در پی) برابر O(m) می‌باشد. در نهایت بعد از ترکیب دو تابع trajectory و lengthپیچیدگی O(m) را خواهیم داشت. زیرا : O(m) + O(m) = O(m)

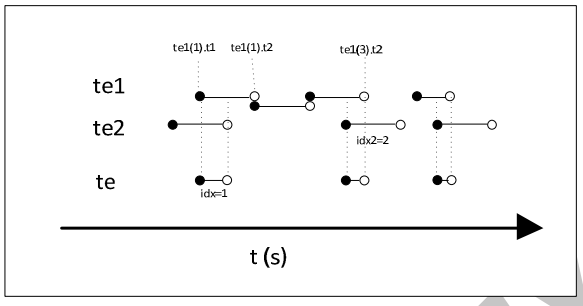
تابع intersection با امضاء:

mpoint × sdo_geometry → mpoint

یک نقطه‌ی متحرک را با یک نقطه یا سطح تقاطع می‌دهد و قسمتی از نقطه‌ی متحرک که با نقطه یا سطح تقاطع دارد را بازمی‌گرداند. رویکرد عمومی برای آن، تقاطع ترتیبی unitهای نقطه‌ی متحرک با شکل هندسی و در صورتی که تقاطع داشتند، ذخیره در نقطه متحرک خروجی می‌باشد. در نهایت پیچیدگی این عملیات O(m)می‌باشد.

تابع passes، یک نقطه‌ی متحرک و یک نقطه یا سطح به عنوان ورودی می‌پذیرد، در صورتی که نقطه‌ی متحرک از نقطه یا سطح عبور کند، TRUE برمی‌گرداند. در این تابع، از روشی که در تابع intersection استفاده شده باید استفاده نمود. یعنی با تقاطع ترتیبی unitها با شکل هندسی، در مواردی که اولین unit با شکل هندسی تقاطع داشت، TRUE و در غیر این صورت FALSE برمی‌گرداند. در بدترین حالت پیچیدگی آن O(m) می‌باشد.

در سه تابع at_temp_element، f_intersection و distance از مرحله‌ی مقدماتی پالایش افراز، استفاده می‌شود. تابع at_temp_element لحظه‌ی ابتدای اولین unit از te1 یعنی te1(1).t1 را در te2[1] جستجوی دودویی می‌کند. اگر بازه‌ای در te2 یافت شد و t1 در آن قرار داشت، اشتراک دو unit از te1 و te2 در خروجی کپی می‌شوند. در غیر این صورت ابتدا لحظه‌ی ابتدایی اولین unit از te2 یعنی te2(1).t1را در te1 جستجو می‌شود، اگر بازه‌ای در te1 یافت شد، اشتراک آنها برمی‌گردد، در غیراین صورت لحظه‌ی ابتدای unit اول دو عنصرزمانی را با یکدیگر مقایسه می‌شوند، اگر برای te1 کوچک‌تر بود، idx1 یعنی شاخص مربوط به te1 یکی افزایش می‌یابد، در غیر اینصورت، شاخص مربوط به te2 یکی افزایش می‌یابد (شکل ۱) اندیس‌های دو آرایه به همین صورت به طور موازی افزایش می‌یابد تا زمانی که به انتهای‌اندیس یکی از آرایه‌ها می‌رسد (الگوریتم ۱). این الگوریتم به دلیل پیچیدگی خاص و همچنین به دلیل کاربرد آن در ۳ تابع مختلف، به صورت pseudo-code آورده شده است. در این الگوریتم از تابع binary_search استفاده شده است که یک لحظه از یک عنصرزمانی(te(idx).t)را در یک عنصرزمانی دیگر جستجوی دودویی کرده و در صورت یافت شدن، شاخص آن را برمی‌گرداند.



شکل ۳- پالایش افزار برای دو عنصرزمانی

در تابع f_intersection بعد از یافتن اشتراک بازه‌ی زمانی دو نقطه‌ی متحرک، دو نقطه متحرک جدید بدست می‌آید که اگر مقادیر unit_functionها در بازه‌های زمان‌ی مشترک در این دو نقطه متحرک یکسان بود، نتیجه در خروجی تابع ذخیره می‌شود. در تابع distance نیز با همین روش، ابتدا اشتراک بازه‌های زمانی محاسبه شده، سپس فاصله‌ی دو نقطه‌ی متحرک جدید برای هر unit با استفاده از رابطه‌ی فاصله دو نقطه محاسبه می‌شود.

تابع فاصله بعد از ساده کردن، به یک تابع رادیکالی از تابع چندجمله‌ای درجه دوم می‌رسد که با یک unit_moving_real[2] قابل نمایش می‌باشد.

[1] در اینجا te1,te2 هر کدام یک عنصرزمانی هستند.

[2] از سه ضریب (a,b,c) چندجمله‌ای و یک پارامتر برای تعیین رادیکالی بودن تابع و یک بازه‌ی زمانی، تشکیل شده است.

```
Algorithm 1 Parition_Refinement
Input : two temporal element(te1,te2)
Output:  a temporal element (te)
idx = 1, index for temporal element te;
idx1 = 1, index for te1;
idx2 = 1, index for te2;
find = 1,
while idx1 <= te1.count and idx2 <= te2.count  Loop
if idx1 is changed or idx1==1 then
find = binary_search(te1(idx1).t1,te2);
        if find exists then
          te(idx).t1 = te1(idx1).t1;
          idx2 = find;
                    if te1(idx1).t2 <= te2(idx2).t2 ;
                    te(idx).t2 = te1(idx1).t2;
                    idx1++;
                    else
                    te(idx).t2 = te2(idx2).t2;
                    idx2++;
                    end if;
idx++;
end if;
elseif idx2 is changed or idx2 =1 then
find = binary_search(te2(idx2).t1,te1);
        if find exists then
          te(idx).t1 = te2(idx2).t1;
          idx1= find;
                    if te1(idx1).t2 <= te2(idx2).t2 ;
                    te(idx).t2 = te1(idx1).t2;
                    idx1++;
                    else
                    te(idx).t2 = te2(idx2).t2;
                    idx2++;
                    end if;
idx++;
end if;
else
          if (te1(idx1).t1 <= te2(idx2).t1 then
          idx1++;
          else
          idx2++;
          end if;
end Loop;
```

۵- عوامل موثر در عملکرد سیستم

برای ارزیابی عملکرد مدل داده‌ای پیاده شده در پایگاه داده، معیار ها [۱] ابزار مناسبی می‌باشند[۲۴]. معیارها از تعدادی پرس‌وجو و داده‌هایی مشخص و در دسترس عموم که حجم داده‌های آن قابل کنترل می‌باشد، تشکیل شده‌اند. در پایگاه داده، معیارها از مجموعه‌ای از پرس‌وجوی‌های SQL تشکیل شده‌اند. هر چند ساختار داده‌ها، شاخص‌ها و پیاده سازی عملگرها می‌توانند بطور مجزا از دیگر اجزاء سیستم، بررسی و مقایسه شوند، ولی تأثیر آنها در پایگاه داده هنگامی مشاهده می‌شود که همه‌ی آنها در کنارهم و در یک سیستم واقعی آزمایش شوند. معیارها، مدل‌های داده و پیاده‌سازی‌های مختلف که توسط گروه‌های تحقیقاتی مختلف انجام شده‌اند را قابل مقایسه می‌کند. در ده سال اخیر مقالات فراوانی در حوزه‌ی اشیاء متحرک ارائه شده که تمرکز بسیاری از آنها روی شاخص‌گذاری‌های خاص و الگوریتم‌های بهینه برای عملگرها بوده است. این درحالی است که ممکن است عملکرد ضعیف سیستم دلیل دیگری داشته باشد و اصلا نیازی به بهینه کردن پردازش پرس‌وجوها و الگوریتم‌ها نباشد. به عنوان مثال، ممکن است با تغییر اندکی در ساختارداده، عملکرد را افزایش داد.

معیار معرفی شده در [۲۴] (BerlinMod) که حرکت وسایل نقلیه در شهر برلین را مدل می‌کند، ابزار مناسبی برای ارزیابی پایگاه‌های داده اشیاء متحرک می‌باشد.BerlinMod، از دو رویکرد برای ذخیره داده‌های اشیاء متحرک استفاده می‌کند. یکی OBA[۲] که در آن کل تاریخچه‌ی حرکت یک شیء متحرک در یک ستون ذخیره می‌شود و دیگری TBA[۳] که تاریخچه‌ی حرکت شی را به سفرهای کوتاه تر تقسیم می‌کند. لذا در رویکرد دوم برای یک شیء مشخص چند سطر در پایگاه داده خواهیم داشت. این تغییر در نحوه‌ی ذخیره‌سازی داده‌ها، باعث افزایش و یا کاهش عملکرد در بعضی از پرس‌وجوها گردیده است. در BerlinMod از مدل حرکت نامقید دو بعدی استفاده شده است، ولی می‌توان آن را برای حرکت مقید، حرکت‌های دوره‌ای[۴][۲۸] و خط سیر مفهومی[۵][۲۹] استفاده نمود. در این مقاله نیز از داده‌های تولیده شده در معیار معرفی شده استفاده شده است. برای تولید داده‌های شبیه سازی شده از پایگاه داده Secondo استفاده شده است و داده‌هایی با ضریب مقیاس‌های مختلف ایجاد شده است. هم چنین برای سنجش عملکرد سیستم از ۱۷ پرس‌وجو که عموما پرس‌وجوهای بازه‌ای می‌باشند، استفاده شده است. البته نویسندگان در [۳۰] معیار را با تعدادی پرس‌وجوی نزدیک ترین k همسایگی[۶] گسترش دادند.

[۱] Benchmarks

[۲] Object Based Approach
[۳] Trip Based Approach
[۴] Periodic Movement
[۵] Semantics Trajectory
[۶] K-Nearest Neighbors Query

تمامی‌پرس‌وجوهای معیار مورد نظر، به قالب مورد استفاده در پایگاه داده اوراکل تبدیل شده‌اند. در جدول ۴ و ۵، تمامی‌این پرس‌وجوها آورده شده است. جداول استفاده شده در پرس‌وجوها نیز از قرار زیر می‌باشند:

Journey_Mod: **Relation**{moid varchar2, licence varchar2,type varchar2,model varchar2,mpointmoving_point};

QueryLicences: **Relation**{id number, licence varchar2 };-100 rows

QueryInstances: **Relation**{id number, instants time_point }; -100 rows

QueryPeriods: **Relation**{id number, periods time_period}; -100 rows

QueryPoints: **Relation**{id number, geometry sdo_geometry}; -100 rows

QueryRegions: **Relation**{id number, geometry sdo_geometry}; -100 rows

بعد از پیاده‌سازی و آزمایش سیستم، برای افزایش عملکرد هر پرس‌وجو نتایج جدول ۳ اخذ گردید. در غالب پرس‌وجوها با ایجاد شاخص (مکانی، زمانی و مکانی-زمانی)، تا حد بسیار زیادی می‌توان عملکرد سیستم را افزایش داد. در موارد دیگر برای کاهش زمان اجرای پرس-وجو، تعداد داده‌های جداول مورد پرس‌وجو از ۱۰۰ به ۱۰ تقلیل یافته است.

جدول ۳- نتایج حاصل ازپرس‌وجوها

شماره پرس‌وجو	ارزیابی
۱	یک شاخص روی Licence در هر دو جدول می‌تواند مفید باشد.
۲	—
۳	یک شاخص زمانی روی mpoint می‌تواند مفید باشد.
۴	یک شاخص مکانی روی mpoint می‌تواند مفید باشد.
۵	شاخص روی Licence تاثیر بالایی روی عملکرد این پرس‌وجو دارد.
۶	یک شاخص مکانی(مکانی-زمانی) روی mpoint تاثیر بالایی در این پرس‌وجو دارد. از این پرس‌وجومی‌توان برای مقایسه شاخص‌های تخصصی مکانی-زمانی استفاده نمود.
۷	—
۸	—
۹	—
۱۰	از آنجایی که این پرس‌وجو پیچیده می‌باشد (فاصله‌ی زمانمند بین همه‌ی ماشین‌ها باید محاسبه شود)، تعداد Licenceها به ۱۰ مورد محدود شده است. هیچ شاخصی به عملکرد آن کمکی نمی‌کند. الگوریتم پیاده‌شده برای تابع at_temp_element در سرعت این پرس‌وجو موثر است.
۱۱	در اینجا شاخص صرفا مکانی، از شاخص مکانی-زمانی بهتر عمل می‌کند. شاخص زمانی هم تاثیری در عملکرد ندارد.
۱۲	به دلیل اتصال (Join) شاخص مکانی-زمانی روی mpoint تاثیر بالایی در عملکرد این پرس وجو دارد.
۱۳	این یک پرس‌وجو با پنجره‌ی سه بعدی (مکانی-زمانی) می‌باشد. عملکرد آن تا حد زیادی بستگی به شاخص مکانی-زمانی مورد استفاده دارد.
۱۴	این پرس وجو احتیاج به یک شاخص زمانمند روی Mpoint و یک شاخص مکانی روی Region دارد.
۱۵	یک شاخص تک بعدی زمانی روی mpoint می‌تواند موثر باشد.
۱۶	این پرس‌وجو عملکرد عملگرهای مکانی-زمانی (تقاطع) را می‌سنجد.
۱۷	—

در اینجا معادل پرس‌وجوهای استفاده شده در BerlinMOD به همراه ارزیابی‌ها و همچنین پرس‌وجوهای پیاده شده در پایگاه داده ملاحظه می‌شود.

جدول ۴- معادل فارسی پرس‌وجوهای معیار

شماره	پرس‌وجو
۱	مدل ماشین‌هایی با پلاک از QueryLicences کدام می‌باشد؟
۲	چه تعداد ماشین از نوع 'Passenger' وجود دارد؟
۳	ماشین‌هایی با پلاک از QueryLicences1 در لحظه‌هایی از QueryInstants1 کجا قرار داشته‌اند؟
۴	چه ماشین‌هایی (پلاک) از نقاط موجود در QueryPoints عبور کرده‌اند؟
۵	کم ترین فاصله‌ی بین مکان‌هایی که یک ماشین با پلاک از QueryLicences1 و ماشین دیگر با پلاک از QueryLicences2 از هم داشته‌اند؟
۶	جفت پلاک ماشین‌هایی از نوع 'Truck' که فاصله آنها از هم از ۱۰ متر کمتر شده است؟
۷	پلاک ماشین‌هایی از نوع 'Passenger' که زودتر از ماشین‌های دیگر از نوع 'Passenger' به نقاط QueryPoints رسیده‌اند؟
۸	مسافت طی شده توسط ماشین‌ها با پلاک از QueryLicences و در بازه‌هایی از QueryPeriods چقدر است؟
۹	طولانی ترین مسیری که توسط ماشین‌های مختلف در بازه‌های زمانی از QueryPeriods طی شده است؟
۱۰	ماشین‌هایی با پلاک از QueryLicences در کجا و چه زمانی فاصله‌ی آنها از دیگر ماشین‌ها کمتر از ۳ متر شده است؟
۱۱	چه ماشین‌هایی از نقاطی از QueryPoints1 در لحظه‌ای از QueryInstants1 عبور کرده‌اند؟
۱۲	چه ماشین‌هایی در یک نقطه از QueryPoints1 و در یک لحظه از QueryInstants1 همدیگر را ملاقات کرده‌اند؟
۱۳	چه ماشین‌هایی در یک ناحیه از QueryRegions1 و یک بازه زمانی از QueryPeriods1 عبور کرده‌اند؟
۱۴	چه ماشین‌هایی در یک ناحیه از QueryRegions1 و یک لحظه از QueryInstants1 عبور کرده‌اند؟
۱۵	کدام ماشین‌ها از یک نقطه از QueryPoints1 و در بازه‌ی زمانی از QueryPeriods1می‌گذرند؟
۱۶	جفت ماشین‌هایی که‌یکی از QueryLicences1 و دیگری از QueryLicences2 باشد و هر دو در یک ناحیه از QueryRegions1 و در یک بازه از QueryPeriods1 قرار داشته باشند ولی همدیگر را ملاقات نمی‌کنند؟
۱۷	چه نقطه‌ای از QueryPoints، بیشترین ماشین‌های متمایز از آن عبور کرده‌اند؟

جدول ۵- پرس‌وجوهای معیار BerlinMOD

شماره	پرس‌وجو
۱	SELECTC.Model, C.licence FROM journey_mod C,QueryLicences ql WHERE C.licence = ql.licence;
۲	SELECT COUNT(Licence) FROM journey_mod WHERE Type = 'passenger' ;0.099
۳	SELECT LL.Licence ASLicence , II . Instant AS Instant , C.mpoint.at_instant(II.Instantss) AS Pos FROM journey_mod C, QueryLicences LL, QueryInstants II WHERE C. Licence = LL. Licence AND C.mpoint.at_instant(II.Instantss) is not NULL ;
۴	SELECT PP.geometry AS Pos, C. Licence AS Licence FROM journey_mod C, QueryPoints PP WHERE to_number(C.moid)=10 AND Passes(C.mpoint,PP.geometry,.005) = 'TRUE';
۵	SELECT LL1.Licence AS Licence1 , LL2.Licence AS Licence2 , sdo_geom.sdo_distance(f_trajectory (V1.mpoint) , f_trajectory (V2.mpoint),.005) AS Dist

شماره	پرس‌وجو
	FROM journey_mod V1, journey_mod V2, QueryLicences1 LL1, QueryLicences2 LL2 WHERE V1.Licence = LL1.Licence AND V2.Licence = LL2.Licence AND V1.Licence <> V2.Licence;
۶	SELECT V1.Licence ASLicence1, V2.Licence AS Licence2 FROM journey_mod V1, journey_mod V2 WHERE V1.Licence < V2. Licence AND V1.Type = 'truck' AND V2.Type = 'truck' AND F_distance(V1.mpoint ,V2.mpoint,.05).Less_Than_Or_Equal(10.0).Sometimes() ='TRUE';
۷	SELECT a.Pos_id , a.Pos, a.Licence, Min(a.time) FROM (SELECT PP.id Pos_id, PP.geometry AS Pos, V1.Licence AS Licence ,f_initial(V1.mpoint.f_intersection(PP.geometry, .005)).t time FROM journey_mod V1, QueryPoints PP WHERE Passes(V1.mpoint,PP.geometry,.005) = 'TRUE' AND V1.Type = 'passenger') a;
۸	SELECT V1.Licence AS Licence, PP.Periods AS Period, f_length(V1.mpoint.at_temp_element(PP.Periods)) AS Dist FROM journey_mod V1, QueryPeriods1 PP, QueryLicences1 LL WHERE V1.Licence = LL.Licence AND present(V1.mpoint, PP.Periods) = 'TRUE' ;
۹	SELECT PP.Periods AS Period , MAX(f_length(V1.mpoint.at_temp_element(PP.Periods))) AS Dist FROM journey_mod V1, QueryPeriods PP WHERE present(V1.mpoint, PP.Periods) = 'TRUE' GROUP BY PP.Periods ;
۱۰	SELECT V1.Licence AS QueryLicence, V2.Licence AS OtherLicence , V1.mpoint.at_temp_element(f_distance(V1.mpoint , V2.mpoint,0.005).less_than_or_equal(3.0).f_intersection('TRUE').f_temp_element()) AS Pos FROM journey_mod V1, journey_mod V2, QueryLicences1 LL WHERE V1.Licence = LL. Licence AND V2.Licence <> V1.Licence AND f_distance(V1.mpoint ,V2.mpoint,0.005).less_than_or_equal(3.0).sometimes = 'TRUE';
۱۱	SELECT C. Licence AS Licence, PP.geometry AS Pos, II.Instants AS Instant FROM journey_mod C, QueryPoints1 PP, QueryInstants1 II WHERE SDO_EQUAL(PP.geometry,C.mpoint.at_instant(II.Instants)) = 'TRUE';
۱۲	SELECT PP.geometry AS Pos, II.Instants AS Instant , C1.Licence AS Licence1 , C2.Licence AS Licence2 FROM journey_mod C1, journey_mod C2, QueryPoints1 PP, QueryInstants1 II WHERE SDO_EQUAL(PP.geometry,C1.mpoint.at_instant(II.Instants)) = 'TRUE' AND SDO_EQUAL(PP.geometry,C2.mpoint.at_instant(II.Instants)) = 'TRUE';
۱۳	SELECT RR.Geometry AS Region, PP.Periods AS Period , C.Licence AS Licence FROM journey_mod C, QueryRegions1 RR, QueryPeriods1 PP WHERE C.mpoint.at_temp_element(PP.Periods).f_intersection(RR.Geometry,.005) is not NULL;
۱۴	SELECT RR.Geometry AS Region, II.Instants AS Instant , C. Licence AS Licence FROM journey_mod C, QueryRegions1 RR, QueryInstants1 II WHERE SDO_INSIDE(RR.Geometry,C.mpoint.at_instant(II.Instants)) = 'TRUE';
۱۵	SELECT PO.geometry AS Pos, PR.Periods AS Period ,

شماره	پرس‌وجو
۱۶	C. Licence AS Licence FROM journey_mod C, QueryPoints1 PO, QueryPeriods1 PR WHERE C.mpoint.at_temp_element(PR.Periods).f_intersection(PO.Geometry,.005) is not NULL ; SELECT PP.Periods AS Period , RR.Geometry AS Region, C1.Licence AS Licence1 , C2.Licence AS Licence2 FROM journey_mod C1, journey_mod C2, QueryRegions1 RR, QueryPeriods1 PP, QueryLicences1 LL1, QueryLicences2 LL2 WHERE C1.Licence = LL1.Licence AND C2.Licence = LL2.Licence AND LL1.Licence < LL2.Licence AND Passes(C1.mpoint.at_temp_element(PP.Periods),RR.geometry,.005) = 'TRUE' AND Passes(C2.mpoint.at_temp_element(PP.Periods),RR.geometry,.005) = 'TRUE' AND f_Intersection(C1.mpoint , C2.mpoint).at_temp_element(PP.Periods).f_intersection(RR.Geometry,.005) is NULL ;
۱۷	CREATE VIEW PosCount AS SELECT PP.id geom_id, COUNT(C.Licence) AS Hits FROM QueryPoints PP, journey_mod C WHERE Passes(C.mpoint,PP.geometry,.005) = 'TRUE' GROUP BY PP.id; SELECT geom_id FROM PosCount N,(SELECT MAX(Hits) max_hit FROM PosCount) M WHERE N.Hits = M.max_hit ;

۶- جمع‌بندی و افق آینده

در این مقاله بعد از مروری کامل بر کارهای انجام شده در گذشته ، به توضیح مختصری درباره‌ی مدل و ساختار داده در پایگاه‌های داده اشیاء متحرک با رویکرد تاریخچه‌ی حرکت و جنبه‌ها و چالش‌های پیاده سازی آن در پایگاه‌های داده پرداخته شد. سپس زیرمجموعه‌ای از توابع و عملگرهای استفاده شده در پرس‌وجوها، انتخاب و الگوریتم‌ها به همراه پیچیدگی هر کدام بررسی شدند. با آزمایشات انجام شده، در مواردی که چندین الگوریتم برای یک تابع وجود داشته، الگوریتمی که بالاترین عملکرد را برای داده‌های مختلف را دارد، آورده شده‌است.

به عبارتی تعدادی از الگوریتم‌ها برای نوع خاصی از داده‌ها سریع‌تر پاسخ می‌دهند، ولی اگر داده، به این شکل خاص نباشند، سرعت به شدت کاهش می‌یابد. همچنین تعدادی از توابع معرفی شده در [۱] احتیاج به الگوریتم مجزایی ندارد و با استفاده از عملگرهای موجود در پایگاه داده، می‌توان این توابع را پیاده نمود. در این حالت‌ها حتی سرعت پردازش پرس‌وجوها بسیار بالاتر از الگوریتم-های ارائه شده در [۱] می‌باشد. نهایتا با انتخاب معیار BerlinMod به عنوان معیار مناسب برای ارزیابی MODs، به ارائه‌ی عوامل موثر در عملکرد سیستم و روش‌هایی برای افزایش عملکرد سیستم پرداخته شد. در حال حاضرتلاش نویسندگان بر انجام پرس‌وجوها با استفاده از شاخص‌های تخصصی‌تر، مثلا TB-Tree [۳۱] ، و مقایسه عددی با استفاده از زمان پاسخ پرس‌وجو می‌باشد. در آینده با تغییر در ساختار داده و همچنین استفاده از شاخص گذاری‌های تخصصی تر برای MODs مثل TB-Tree [۳۱] به مقایسه‌ی سیستم‌هایی که از مدل داده تاریخچه‌ی حرکت استفاده می‌کنند، خواهیم پرداخت. همچنین از پرس‌وجوهای نزدیک‌ترین همسایگی نیز در مقایسات استفاده خواهد شد.

مراجع

[1] N. Pelekis, B. Theodoulidis, I. Kopanakis, and Y. Theodoridis, “Theodoridis. Literature review of spatiotemporal database models,” *THE KNOWLEDGE ENGINEERING REVIEW*, vol. 19, p. 235--274, 2005.

[2] A. Prasad Sistla, O. Wolfson, S. Chamberlain, and S. Dao, “Modeling and querying moving objects,” in *Data Engineering, 1997. Proceedings. 13th International Conference on*, 1997, pp. 422–432.

[3] A. Prasad Sistla, O. Wolfson, S. Chamberlain, and S. Dao, “Querying the uncertain position of moving objects,” *Temporal Databases: Research and Practice*, pp. 310–337, 1998.

[4] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez, “Cost and imprecision in modeling the position of moving objects,” in *Data Engineering, 1998. Proceedings., 14th International Conference on*, 1998, pp. 588–596.

[5] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang, “Moving objects databases: Issues and solutions,” in *Scientific and Statistical Database Management, 1998. Proceedings. Tenth International Conference on*, 1998, pp. 111–122.

[6] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha, “Updating and querying databases that track mobile units,” *Distributed and parallel databases*, vol. 7, no. 3, pp. 257–387, 1999.

[7] G. Langran and N. R. Chrisman, “A framework for temporal geographic information,” *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 25, no. 3, pp. 1–14, 1988.

[8] G. Langran, *Time in geographic information systems*. CRC, 1992.

[9] G. J. Hunter and I. P. Williamson, “The development of a historical digital cadastral database†,” *International Journal of Geographical Information System*, vol. 4, no. 2, pp. 169–179, 1990.

[10] “D. Peuquet and E. Wentz, ‘An approach for time-based spatial analysis of spatio-temporal data,’ in Advances in GIS Research, Proceedings 1, 1994, pp. 489–504.

[11] D. J. Peuquet and N. Duan, “An event-based spatiotemporal data model (ESTDM) for temporal analysis of geographical data,” *International Journal of Geographical Information Systems*, vol. 9, no. 1, pp. 7–24, 1995.

[12] C. Claramunt and M. Thériault, “Managing time in GIS: an event-oriented approach,” in *Proceedings of the International Workshop on Temporal Databases: Recent Advances in Temporal Databases*, 1995, pp. 23–42.

[13] M. Erwig, R. H. Guting, M. Schneider, and M. Vazirgiannis, “Spatio-temporal data types: An approach to modeling and querying moving objects in databases,” *GeoInformatica*, vol. 3, no. 3, pp. 269–296, 1999.

[14] L. Forlizzi, R. H. Güting, E. Nardelli, and M. Schneider, *A data model and data structures for moving objects databases*, vol. 29. ACM, 2000.

[15] R. H. Güting et al., “A foundation for representing and querying moving objects,” *ACM Transactions on Database Systems (TODS)*, vol. 25, no. 1, pp. 1–42, 2000.

[16] C. L. JA, L. Forlizzi, R. H. Guting, E. Nardelli, and M. Schneider, “Algorithms for Moving Objects Databases,” *Computer Journal*, vol. 46, no. 6, pp. 680–712, 2003.

[17] S. Dieker and R. H. Guting, “Plug and play with query algebras: SECONDO, a Generic DBMS development environment,” in *Database Engineering and Applications Symposium, 2000 International*, 2000, pp. 380–390.

[18] R. H. Guting et al., “Secondo: An extensible DBMS platform for research prototyping and teaching,” in *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, 2005, pp. 1115–1116.

[19] R. H. Güting, T. Behr, and C. D\üntgen, “Secondo: A platform for moving objects database research and for publishing and integrating research implementations,” *Data Engineering*, p. 56, 2010.

[20] N. Pelekis and Y. Theodoridis, “Boosting location-based services with a moving object database engine,” in *Proceedings of the 5th ACM international workshop on Data engineering for wireless and mobile access*, 2006, pp. 3–10.

[21] N. Pelekis, Y. Theodoridis, S. Vosinakis, and T. Panayiotopoulos, “Hermes-a framework for location-based data management,” *Advances in Database Technology-EDBT 2006*, pp. 1130–1134, 2006.

[22] N. Pelekis and Y. Theodoridis, “An oracle data cartridge for moving objects,” 2005.

[23] “S. Boulahya, ‘Repr´esentation et interrogation de donn´ees spatio-temporelles: Cas d’´etude sur PostgreSQL/PostGIS,’ Master’s thesis, Universit´e Libre de Bruxelles, Facult´e des Sciences, D’epartement d’Informatique, 2009..

[24] C. Düntgen, T. Behr, and R. H. Güting, “BerlinMOD: a benchmark for moving object databases,” *The VLDB Journal—The International Journal on Very Large Data Bases*, vol. 18, no. 6, pp. 1335–1368, 2009.

[25] L. Becker, H. Blunck, K. Hinrichs, and J. Vahrenhold, “A framework for representing moving objects,” in *Database and Expert Systems Applications*, 2004, pp. 854–863.

[26] B. Yu, S. H. Kim, T. Bailey, and R. Gamboa, “Curve-based representation of moving object trajectories,” in *Database Engineering and Applications Symposium, 2004. IDEAS’04. Proceedings. International*, 2004, pp. 419–425.

[27] D. E. Knuth, “Big omicron and big omega and big theta,” *ACM Sigact News*, vol. 8, no. 2, pp. 18–24, 1976.

[28] T. Behr, V. T. de Almeida, and R. H. G\üting, “Representation of periodic moving objects in databases,” in *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, 2006, pp. 43–50.

[29] A. Frihida, D. Zheni, H. Ben Ghezala, and C. Claramunt, “Modeling trajectories: A spatio-temporal data type approach,” in *Database and Expert Systems Application, 2009. DEXA’09. 20th International Workshop on*, 2009, pp. 447–451.

[30] R. H. Güting, T. Behr, and J. Xu, “Efficient k-nearest neighbor search on moving object trajectories,” *The VLDB Journal*, vol. 19, no. 5, pp. 687–714, Oct. 2010.

[31] D. Pfoser, C. S. Jensen, and Y. Theodoridis, “Novel approaches to the indexing of moving object trajectories,” in *Proceedings of VLDB*, 2000, pp. 395–406.

[32] R. H. Güting, T. Behr, and J. Xu, “Efficient k-nearest neighbor search on moving object trajectories,” *The VLDB Journal*, vol. 19, no. 5, pp. 687–714, Oct. 2010.