Developing an Oracle-Based Spatio-Temporal Information Management System

Lei Zhao, Peiquan Jin, Lanlan Zhang, Huaishuai Wang, and Sheng Lin

School of Computer Science and Technology,
University of Science and Technology of China, 230027, Hefei, China
jpq@ustc.edu.cn

Abstract. In this paper, we present an extension of Oracle, named STOC (Spatio-Temporal Object Cartridge), to support spatio-temporal data management in a practical way. The extension is developed as a PL/SQL package and can be integrated into Oracle to offer spatio-temporal data types as well as spatio-temporal operations for various applications. Users are allowed to use standard SQL to access spatio-temporal data and functions. After an overview of the general features of STOC, we discuss the architecture and implementation of STOC. And finally, a case study of STOC is presented, which shows that STOC is effective to represent and query spatio-temporal data on the basis of Oracle.

1 Introduction

Nowadays, many applications show the demand on moving objects management. However, traditional database systems, i.e., the relational DBMSs, can not deal with spatio-temporal data efficiently. Therefore, it has been one of the most important goals in recent research on spatio-temporal databases to design and implement a practical spatio-temporal DBMS.

Previous research on spatio-temporal databases were mainly focused on spatio-temporal semantics[1-2], spatio-temporal data models [3-4], spatio-temporal indexes [5] and spatio-temporal query processing [6-7], whereas little work has been done in the implementation of practical spatio-temporal DBMSs owing to the complexity of spatio-temporal data models and the lack of effective implemental techniques. Although some commercial or open-source DBMSs [8-10] offer support for handling special data types, it is still not feasible to use them to support spatio-temporal data. Recently, the object-relational database technology has been paid a lot of attention in spatio-temporal data model [11] and system implementation, due to its extensibility on new data types and functions.

In this paper, we present an extension of Oracle, named STOC (Spatio-Temporal Object Cartridge), to support spatio-temporal data management in a practical way. The unique features of STOC can be summarized as follows:

(1) It is SQL-compatible and built on a widely-used commercial DBMS (*see Section 2)*, namely Oracle. Thus it can be easily used in real spatio-temporal database applications and provides a practical solution for spatio-temporal data management under current database architecture.

J. Xu et al. (Eds.): DASFAA Workshops 2011, LNCS 6637, pp. 168–176, 2011.

(2) It supports various spatio-temporal data types (*see Section 3.1*), such as *moving number*, *moving bool*, *moving string*, *moving point*, *moving line*, and *moving region*. Combined with the ten types of spatio-temporal operations (*See Section 3.2*) supported by those new data types, users can represent many types of spatio-temporal data involved in different spatio-temporal applications and query different spatio-temporal scenarios.

(3) It can support real spatio-temporal applications (*See Section 4*). A case study concerning moving taxi cars shows that the STOC system is able to store spatio-temporal data captured by GPS and GIS technologies, and also able to represent different types of spatio-temporal queries.

2 Overview of STOC

STOC (Spatio-Temporal Object Cartridge) is a spatio-temporal cartridge based on Oracle and Oracle Spatial. The detailed implemental architecture of STOC is shown in Fig.1. STOC extends spatio-temporal data types and operations using the PL/SQL scripting language. Once installed, STOC becomes an integral part of Oracle, and no external modules are necessary. Users can use standard SQL to gain spatio-temporal support from Oracle. No external work imposes on users.

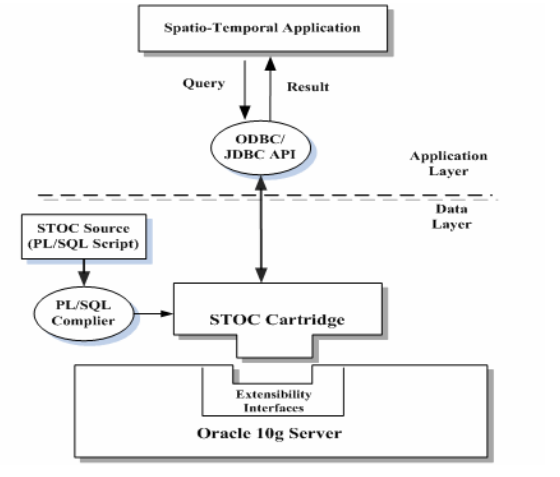

Fig. 1. The Architecture of STOC

We design the spatio-temporal data type model in STOC in order to represent the spatio-temporal objects and complex spatio-temporal changes. STOC extends two categories of new data types into Oracle, namely spatio-temporal data types and temporal data types (as shown in Fig.2). As Oracle has already supported spatial data management from its eighth version, which is known as Oracle Spatial, we build STOC on the basis of Oracle Spatial so as to utilize its mature technologies in spatial data management. The spatio-temporal data types contain moving spatial types and moving base types. The former refers to moving spatial objects in real world, while the latter refers to those numeric, Boolean, or string values changing with time.

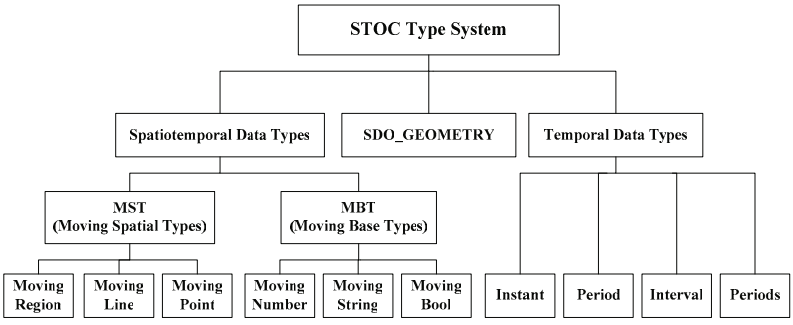

Fig. 2. The Type System of STOC

3 Implementation of STOC

3.1 Moving Data Types in STOC

3.1.1 Moving Base Types (MBT)

MBT represents the base data types changing with time, including *Moving Bool*, *Moving String* and *Moving Number*, which are the essentials for describing spatio-temporal changes.

Moving Bool and *Moving String* represent the discretely-changing Boolean values and String values respectively. It consists of a set of *period snapshots* ordered by time. A period snapshot is a <*value*, [*from*, *to*)> pair. The value of object is constant at the period [*from*, *to*). We use the VARRAY data type to organize the set of period snapshots, which is the same as follows.

Moving Number represents the changing numerical values with time. Moving number is a set of *moving number period snapshots* ordered by time. Moving number period snapshot is a quadruple <*t_type*, *num*, *chR*, [*from*, *to*)>. The *t_type* is the change type of moving number snapshot, when it equals to *0* meaning discrete change, or equals to *1* meaning continuous change. The *num* is the value at the beginning of this period. The [*from*, *to*) is the valid period about this snapshot. The *chR* represent the change rate of value in this period. If the change of moving number is discrete, *chR* is always *0*.

3.1.2 Moving Spatial Types (MST)

MST represent the spatial data types changing with time, including *Moving Point*, *Moving Line* and *Moving Region*, which are the cores of moving objects. MST object is a three-dimension object with spatial space (two-dimension) and temporal space (one-dimension).

Figure 3 shows the definition for moving point using PL/SQL. The *t_type* is the change type of moving point, when it equals to *0* meaning discrete change, or equals to *1* meaning continuous change. The *srid* is the coordinate system id for moving point. STOC supports the coordinate systems for Oracle Spatial, such as *NULL* (user-defined coordinate system), *8307* (WGS-84 coordinate system). Those coordinate systems are described in MDSYS.CS_SRS system table. *M_point_units* is a set of

```
CREATE OR REPLACE TYPE M_POINT_U AS OBJECT
 (point mdsys.SDO_GEOMETRY,
  cRX NUMBER,--changeRateofX
  cRY NUMBER,--changeRateofY
  period T_period) ;
CREATE OR REPLACE TYPE M_POINT_U_ARRAY IS VARRAY (102400) OF M_POINT_U;
CREATE OR REPLACE TYPE M_POINT AS OBJECT
   (t_type NUMBER,
    srid NUMBER,
  m point units M POINT U ARRAY);
```

Fig. 3. Defining Moving Point using PL/SQL

moving point period units ordered by time. *Moving point period unit* is a quadruple <*point*, *cRX*, *cRY*, [*from*, *to*)>. The *point* is the start position at the beginning of the period for the moving point. The *cRX* and *cRY* represent the change rates for x-axis position and y-axis position respectively. The [*from*, *to*) is the valid period for this moving point unit.

Since moving line only supports the discrete change, similar to moving bool and moving string, moving line is a set of *moving line period units* ordered by time. Each moving line period unit indicates the position for moving line and its valid period.

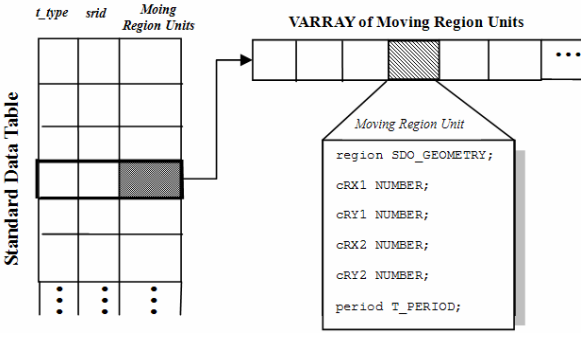

Fig. 4. Data Structure of Moving Region

The data structure of moving region is shown in Fig.4. In this data structure, the *t_type* is the change type of moving region; the *srid* is the coordinate system id; the *m_region_units* is a set of *moving region period units* ordered by time. The moving region period unit is a six-tuple. The *period* is the valid period. On the one hand, if moving region changes discretely, the *region* stores the location information of moving region in this period. On the other hand, if the change is continuous, the *region* stores the *MBR* of moving region at the beginning of this period. The *cRX1*, *cRY1*, *cRX2*, *cRY2* represent the bottom-left and top-right points' change rates for x-axis position and y-axis position respectively.

3.2 Spatio-Temporal Operations in STOC

STOC provides the following types of spatio-temporal operations to support various spatio-temporal queries. All the operations are implemented by PL/SQL and as member functions of spatio-temporal data types.

Object data management operations: for moving object, add or delete the specific period unit to manage the data of moving object.

Object attribute operations: get the attributes of moving object. For example, get the max/min value of moving number; get the change type, coordinate system and speed for moving point.

Temporal dimension project operations: get the temporal information of moving object, such as valid periods or start time about moving object.

Value dimension project operations: project the moving object to value dimension to get the value range of moving object. For moving bool/ moving string/moving number, return the set of its values. For moving point, if it changes discretely, return the set of points, if it changes continuously, return the segments of trajectory. For moving line, return the set of lines. For moving region, if it changes discretely, return the set of regions, or return the region where the moving MBR passed.

Temporal selection operations: return the value of moving object at specific *instant*, *period* or *periods*.

Quantification operations: return whether or not the moving bool/moving string always/sometimes equals to the specific bool/string in valid time.

Moving Boolean operations: return the logical relation between two moving bool object. It is used to support the complex logical query. For *not* operation, the result is false when the moving bool is true in one period and vice versa. For *and* operation, the result is false when at least one moving bool is false in one period, or the result is true when both moving bool objects are true in the same period. For *or* operation, the result is true when at least one moving bool is true in one period, or the result is false when both moving bool objects are false in the same period.

Temporal relation operations: return the temporal relations between each period unit of moving object and specific *instant*, *period*. The temporal relations are defined in [12].

Object relation operations: return the relations between moving object with moving object or non-moving object which may change with time. The data type of return value is moving string. For moving bool/moving string, return equal or not equal. For moving number, return greater than, less than or equal. For moving spatial type, return the spatial relations defined in [13].

Distance operations: return the distance between moving object with moving object or non-moving object which may change with time. The data type of return value is moving number.

4 Case Study: A Traffic Information System

To demonstrate the functions of STOC, we use a traffic dataset to study the effectiveness of STOC. The dataset was also used in BerlinMOD [14].

4.1 Create BerlinMOD Database

The schema for BerlinMOD database is shown in Fig.5. The information about moving cars is stored in table *dataScar.* Table QueryPoints /QueryRegions /QueryInstants /QueryPeriods / QueryLicences record the query conditions for position of point

```
dataScar (Licence: Varchar2,
        Model: Varchar2,Type:Varchar2, Trip: M_Point);
QueryPoints    (Id: NUMBER, Pos: SDO_GEOMETRY);
QueryRegions   (Id: NUMBER, Region: SDO_GEOMETRY);
QueryInstants  (Id: NUMBER, Instant: T_Instant);
QueryPeriods   (Id: NUMBER, Period: T_Period);
QueryLicences  (Id: NUMBER, Licence: Varchar2);
```

Fig. 5. Database Schema of BerlinMOD

(SDO_GEOMETRY data type), position of region (SDO_GEOMETRY data type), instant (T_Instant data type), period (T_Period data type), and the license of car (Varchar2 data type). The *id* is the prime key for each query table.

4.2 Spatio-Temporal Queries

STOC supports the various kinds of spatio-temporal queries referred by [11, 14]. According to the query conditions and types, those can be divided into six types of spatio-temporal, which are (1) temporal range query, (2) spatial range query, (3) spatio-temporal range query, (4) spatio-temporal distance query, (5) spatio-temporal topology query and (6) spatio-temporal aggregate query.

Q1 (Temporal Range Query): Where have the vehicles whose licenses are from QueryLicences been at each instant in QueryInstants?

```
  SELECT LL.Licence AS Licence, II.Instant AS Instant, C.Trip.at_instant
(II.Instant) AS Pos
  FROM dataScar C, QueryLicences LL, QueryInstants  II
  WHERE C.Licence = LL.Licence AND C.Trip.at_instant (II.Instant) IS NOT
NULL;
```

Q2 (Spatial Range Query): Which license numbers belong to vehicles that have passed the points from QueryPoints?

```
    SELECT PP. Pos AS Pos, C.Licence AS Licence
    FROM dataScar C, QueryPoints PP
    WHERE     sdo_geom.relate(C.Trip.get_trajectory),     'ANYINTERACT'     ,
PP.Pos,0.005)='TRUE';
```

Q3 (Spatio-Temporal Range Query): Which vehicles traveled within one of the regions from QueryRegions during the periods from QueryPeriods.

```
    SELECT RR. Region AS Region, PP. Period AS Period ,C.Licence AS Licence
    FROM dataScar C, QueryRegions RR, QueryPeriods PP
    WHERE     C.Trip.at_period(PP.Period)  IS  NOT  NULL  AND  sdo_geom.relate
(C.Trip.at_period(PP.Period).get_trajectory()
    ,'ANYINTERACT',RR.region,0.005)='TRUE';
```

Q4 (Spatio-Temporal Distance Query): What are the pairs of licence numbers of "trucks", that have ever been as close as 10m or less to each other?

```
    SELECT V1.Licence AS Licence1, V2.Licence AS Licence2
    FROM dataScar V1, dataScar V2
    WHERE  V1.Licence  <  V2.Licence  AND  V1.Type  =  'truck'  AND  V2.Type  =
'truck' AND V1.Trip.distance_m_Point(V2.Trip,0.005).get_min_number()<=10;
```

Q5 (Spatio-Temporal Topology Query): What are the pairs of licence numbers of "trucks", that have meet each other?

```
SELECT V1.Licence AS Licence1, V2.Licence AS Licence2
FROM dataScar V1, dataScar V2
WHERE V1.Licence < V2.Licence AND V1.Type = 'truck' AND V2.Type =
'truck' AND V1.Trip.relation_m_Point (V2.Trip,0.005).sometimes
('EQUAL')='TRUE';
```

Q6 (Spatio-Temporal Aggregate Query): Which points from QueryPoints have been visited by a maximum number of different vehicles?

```
CREATE VIEW visited_car(pp_id ,licence) AS
    SELECT PP.ID AS Pos , c.licence
    FROM QueryPoints PP, dataScar C
    WHERE sdo_geom.relate(C.Trip. get_trajectory(), 'ANYINTERACT',
     PP.Pos,0.05)='TRUE';
CREATE VIEW PosCount AS
    SELECT vc.pp_id AS Pos_id , COUNT(*) AS Hits
    FROM visited_car vc
    GROUP BY vc.pp_id ;
SELECT N.Pos_id, N.Hits
FROM PosCount  N
WHERE N.Hits = (SELECT MAX(Hits) FROM PosCount );
```

5 Related Work

Early work in implementing spatio-temporal database systems employed a layered approach [16, 17], which aimed at implementing an additional spatio-temporal layer on top of standard relational DBMS. Since the relational DBMS has little built-in spatio-temporal support, generated query may become very complex and potentially difficult to optimize for the underlying representation of spatio-temporal data. The object-relational approach, which refers to extending an extensible DBMS with spatio-temporal plug-ins, has been the most focused approach in recent years, since this makes it feasible to implement a spatio-temporal DBMS completely. Now major database vendors have provided techniques for this extensibility, including Oracle and IBM. And some extensions have been developed for spatial/temporal/spatio-temporal data management. The typical examples are Informix Timeseries Datablade [18] and Informix Geodetic Datablade [19]. The Timeseries Datablade is designed for capturing single attribute that changes over time, in which the temporal data types are very restricted. The Geodetic DataBlade is a spatio-temporal extension provided by Informix, but it is only designed for specific GIS applications.

There are also some research prototypes developed by other researchers. SECONDO [20] realizes the abstract moving object data types and operators defined in [11] and supports spatio-temporal queries and analysis by employing Berkeley DB as data storage. Nonetheless, SECONDO is unable to combine with existing DBMSs and is not compatible with SQL. SpADE [21] is built on MySQL, with the extension of a moving object module and an index module. However it only supports a limited set of spatio-temporal queries, namely the range query and the nearest neighbors query. Hermes [22] is based on the trajectory database of Oracle in support of range query, nearest neighbors query and similar trajectory query of moving point. But it does

not support moving regions and spatio-temporal analysis query. TrajStore [23] only focused on spatio-temporal range query on massive datasets, thus can not suit for different spatio-temporal applications.

6 Conclusions

In this paper, we present a spatiotemporal extension of Oracle called STOC, aiming at providing practical support of spatiotemporal data management for various commercial applications. STOC is developed using the cartridge technology provided by Oracle, which enables us to add new data types as well as functions and indexing methods into the kernel of Oracle. The most valued feature of STOC is that it is SQL-compatible and allows users to develop spatio-temporal applications upon Oracle.

Acknowledgements

This work is supported by the National High Technology Research and Development Program ("863" Program) of China (No. 2009AA12Z204), the National Science Foundation of China (no. 60776801), the Open Projects Program of National Laboratory of Pattern Recognition (20090029), the Key Laboratory of Advanced Information Science and Network Technology of Beijing (xdxx1005), and the USTC Youth Innovation Foundation.

References

1. Sistla, P., Wolfson, O., et al.: Modeling and Querying Moving Objects. In: ICDE, Birmingham, UK (1997)
2. Bohlen, M., Jensen, C., Skjellaug, B.: Spatio-Temporal Database Support for Legacy. In: 1998 ACM Symposium on Applied Computing, Georgia (1998)
3. Erwig, M., Güting, R.H., et al.: Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases. GeoInformatica 3(3), 265–291 (1999)
4. Güting, R.H., Bohlen, M.H., et al.: A foundation for representing and quering moving objects. ACM Transactions Database Systems 25(1), 1–42 (2000)
5. Pfoser, D., Jensen, C.S., Theodoridis, Y.: Novel Approaches to the Indexing of Moving Object Trajectories. In: Proceedings of VLDB (2000)
6. Benetis, R., Jensen, C.S., et al.: Nearest neighbor and reverse nearest neighbor queries for moving objects. In: IDEAS, pp. 44–53 (2002)
7. Lema, J.A.C., Forlizzi, L., et al.: Algorithms for moving objects databases. The Computer Journal 46(6), 680–712 (2003)
8. Oracle Corp. Oracle Spatial User's Guide and Reference, `http:// www.oracle.com /technology/products/spatial/spatial_doc_index.html`
9. SQL Server Spatial Data, `http://www.microsoft.com/sqlserver/2008/en/us/spatial-data.aspx`
10. PostGIS, `http://postgis.refractions.net/`
11. Forlizzi, L., Güting, R.H., et al.: A Data Model and Data Structures for Moving Objects Databases. In: SIGMOD, pp. 319–330 (2000)

12. Allen, J.F.: Maintaining knowledge about temporal intervals. Communication of ACM 26, 832–843 (1983)
13. Egenhofer, M., Franzosa, R.: Point-Set Topological Spatial Relations. International Journal of Geographical Information Systems 5(2), 161–174 (1991)
14. Düntgen, C., Behr, T., Güting, R.H.: BerlinMOD: a benchmark for moving object databases. VLDB J. 18(6), 1335–1368 (2009)
15. Wolfson, O., Sistla, P., et al.: DOMINO: Databases for Moving Objects Tracking. In: SIGMOD, Philadelphia, PA, pp. 547–549 (1999)
16. Torp, K., Jensen, C.S., Bohlen, M.H.: Layered Implementation of Temporal DBMSs-Concepts and Techniques. In: DASFAA, Melbourne, Australia, pp. 371–380 (1997)
17. Torp, K., Jensen, C.S., Snodgrass, R.T.: Stratum Approaches to Temporal DBMS Implementation. In: IDEAS, Cardiff, Wales, pp. 4–13 (1998)
18. Informix Corp. Informix Timeseries DataBlade Module User's Guide, Version 3.1 (1997)
19. Informix Corp. Informix Geodetic DataBlade Module User's Guide, Version 2.1 (1997)
20. Güting, R.H., de Almeida, et al.: Secondo: Anextensible DBMS platform for research prototyping and teaching. In: ICDE, pp. 1115–1116 (2005)
21. Ooi, B.C., Huang, Z., et al.: Adapting Relational Database Engine to Accommodate Moving Objects in SpADE. In: ICDE, Istanbul, pp. 1505–1506 (2007)
22. Pelekis, N., Frentzos, E., Giatrakos, N., Theodoridis, Y.: HERMES:Aggregative LBS via a trajectory DB engine. In: ISIGMOD, pp. 1255–1258 (2008)
23. Cudre-Mauroux, P., Wu, E., Madden, S.: TrajStore: An Adaptive Storage System for Very Large Trajectory Data Sets. In: ICDE (2010)