Einfaches Tracking und Tracing auf Basis von Geodaten

Karin Patenge, Oracle

Wer Nutzungsrechte für die Oracle Datenbank hat, egal in welcher Edition, hat direkten Zugriff auf umfangreiche Funktionalität inenherhalb der Datenbank rund um das Management, die Prozessierung und Analyse sowie zusätzliche Werkzeuge u.a. für die Visualisierung von Geodaten.
Dieser Artikel stellt mit den Themen ***Contract Tracing*** *und* ***Location Tracking*** *zwei konkrete fachliche Kontexte für Geodaten vor, die weitreichende Anwendungsmöglichkeiten erschließen.*

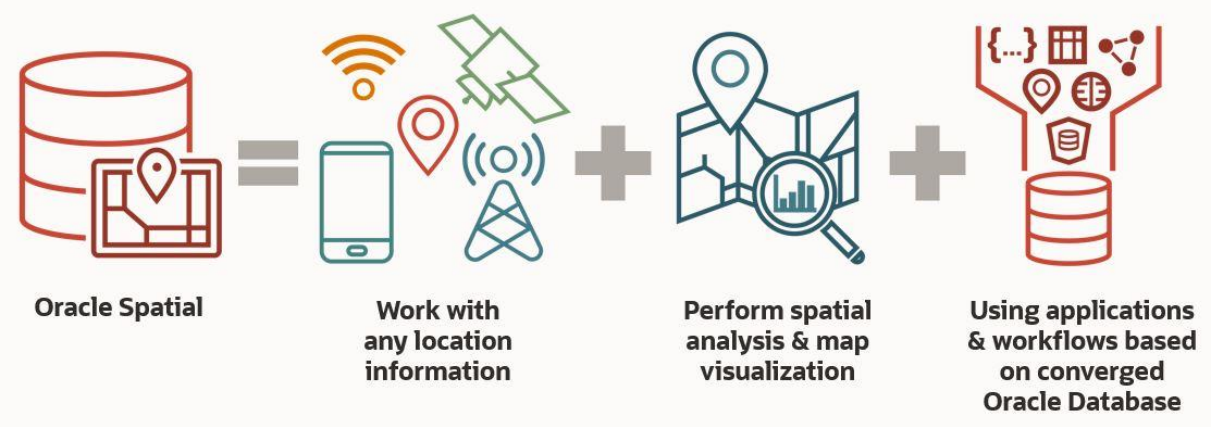

Abbildung 1: Was ist Oracle Spatial?

Contact Tracing – Mittels PL/SQL einfach Kontakte nachverfolgen

Die COVID-19-Pandemie hat alle vor die Herausforderung gestellt, Infektionsketten bestmöglich und zeitnah nachzuverfolgen, das Infektionsrisiko punktgenau einzuschätzen und entsprechende, sinnvolle Maßnahmen daraus abzuleiten. Ein erster Schritt dafür ist, potentielle Risikokontakte zu ermitteln. Wie das geht mit den Mitteln der Oracle Datenbank, möchte ich in diesem Kapitel beschreiben.

Seit ungefähr zwei Jahren gibt es das PL/SQL Paket SDO_OBJ_TRACING, welches als Reaktion auf die Notwendigkeit der Kontaktnachverfolgung von COVID-19 infizierten Personen entstand.Das Paket ist so implementiert, daß es Bewegungsdaten inklusive ihrer raum- und zeitlichen Details – sehr häufig GPS[1]

[1] GPS = Global Positioning System

Daten - hernehmen und analysieren kann. Die Nutzung des Paketes ist sowohl in der MOS Note „*Contact Tracing SQL API Added to Oracle Spatial and Graph (Doc ID 2691413.1)*“ als auch im Blog Post „*Contact Tracing APIs in Oracle Database*“ [1] beschrieben. In der Version 21c ist das Paket standardmäßig verfügbar. Für frühere Versionen gibt es das Paket zum Download über die MOS Note weiter oben im Text. Meine weitere Ausführungen beziehen sich auf Tests mit einer *Always Free Tier*[2] *Autonomous Transactional Processing* Instanz in der Version 21c.

Das *Contact Tracing* Paket kommt gleich mit zwei APIs daher:

1. Die erste API (*Swipe IN/OUT contact tracing*) funktioniert ohne GPS Daten auf der Basis von relational beschriebenen Informationen zu Orten (z.B. Gebäude, Raum, Flur) und Zuständen. Die Daten dazu können zum Beispiel von Sensoren erhoben und geliefert werden, die an den betreffenden Orten installiert sind.
2. Die zweite API (*GPS tracking contact tracing*) verwendet Ortsinformationen aus GPS Tracking Daten.

Beide APIs benötigen zusätzlich sowohl einen Zeitstempel als auch ein die jeweilige Person oder das Objekt identifizierendes Merkmal (z.B. USER_ID).
Listing 1 beschreibt die sehr einfach gehaltenen Datenmodelle für beide APIs mit Hilfe von CREATE TABLE Statements.

```
create table swipe_table (
  user_id      number,
  building_id  varchar2(100),
  floor_id     varchar2(100),
  room_id      varchar2(100),
  swipe_in_out varchar2(5),    -- Zustand (IN oder OUT)
  time DATE);

create table osm_tracks_new (
  user_id                 number,
  geom                    sdo_geometry,
```

[2] OCI Cloud Free Tier (https://www.oracle.com/cloud/free/)

```
  time_adjusted           date,
  time_adjusted_as_number number);
```

Listing 1: CREATE Table Statements für die Datenmodelle der Contact Tracing APIs

Das Paket enthält neben anderen Hilfsfunktionen die zwei folgenden Funktionen:

- GET_ALL_SWIPE_IO_DURATIONS für das *Swipe IN/OUT contact tracing*
- GET_ALL_DURATIONS für das *GPS tracking contact tracing*

Wie werden diese Funktionen genutzt, welche Analysen werden unterstützt? Zunächst einmal kann die grundlegende Frage beantwortet werden, welche Personen oder Objekte zur selben Zeit für welche Dauer am gleichen Ort waren waren. Das Attribut SEGMENT_OR_ALL mit den möglichen Werten SEGMENT oder ALL dient dazu, das Ergebnis auf jede einzelne Begegnung herunterzubrechen (SEGMENT – siehe *Listing 2* und *Abbildung 2*) bzw. in einer weiteren Anfrage alle Kontaktzeiten von zwei Personen (ALL – siehe *Listing 3* und *Abbildung 3*) zu aggregieren.

```
select a.in_user_id,
  a.out_user_id,
  a.building_id,
  a.room_id,
  to_char(a.start_time,'DD-MM-YYYY HH24:MI:SS') as start_time,
  to_char(a.end_time,'DD-MM-YYYY HH24:MI:SS') as end_time,
  round(a.duration/60, 2) as duration_in_min
from table (
  sdo_obj_tracing.get_all_swipe_io_durations (
    user_id                 => 1,
    start_time              => to_date('28-10-2022
08:00:00','DD-MM-YYYY HH24:MI:SS'),
    end_time                => to_date('28-10-2022
20:00:00','DD-MM-YYYY HH24:MI:SS'),
    track_table_name        => 'swipe_table',
    user_id_column_name     => 'user_id',
    building_id_column_name => 'building_id',
    floor_id_column_name    => 'floor_id',     -- optional
```

```
    room_id_column_name      => 'room_id',       -- optional
    swipe_io_column_name     => 'swipe_in_out',
    time_column_name         => 'time')) a       -- optional
where segment_or_all = 'SEGMENT'
order by in_user_id, out_user_id, start_time;
```

Listing 2: Jeder Kontakt von Person mit USER_ID = 1 mit anderen Personen

IN_USER_ID	OUT_USER_ID	BUILDING_ID	ROOM_ID	START_TIME	END_TIME	DURATION_IN_MIN
1	2	BUILDING_1	ROOM_1	28-10-2022 13:00:00	28-10-2022 13:30:00	30
1	2	BUILDING_1	ROOM_2	28-10-2022 16:20:00	28-10-2022 16:40:00	20
1	3	BUILDING_1	ROOM_1	28-10-2022 13:00:00	28-10-2022 14:00:00	60

Abbildung 2: Ergebnis der Abfrage aus Listing 2

```
select a.in_user_id,
  a.out_user_id,
  to_char(a.start_time,'DD-MM-YYYY HH24:MI:SS') as start_time,
  to_char(a.end_time,'DD-MM-YYYY HH24:MI:SS') as end_time,
  round(a.duration/60, 2) as duration_in_min,
  a.num_contact_times
from table (
  sdo_obj_tracing.get_all_swipe_io_durations (
    user_id                  => 1,
    start_time               => to_date('28-10-2022
08:00:00','DD-MM-YYYY HH24:MI:SS'),
    end_time                 => to_date('28-10-2022
20:00:00','DD-MM-YYYY HH24:MI:SS'),
    track_table_name         => 'swipe_table',
    user_id_column_name      => 'user_id',
    building_id_column_name  => 'building_id',
    floor_id_column_name     => 'floor_id',      -- optional
    room_id_column_name      => 'room_id',       -- optional
    swipe_io_column_name     => 'swipe_in_out',
    time_column_name         => 'time')) a       -- optional
where segment_or_all = 'ALL'
order by in_user_id, out_user_id, start_time;
```

Listing 3: Summierte Kontaktdauern für Person mit USER_ID = 1 mit anderen Personen

IN_USER_ID	OUT_USER_ID	START_TIME	END_TIME	DURATION_IN_MIN	NUM_CONTACT_TIMES
1	2	28-10-2022 13:00:00	28-10-2022 16:40:00	50	2
1	3	28-10-2022 13:00:00	28-10-2022 14:00:00	60	1

Abbildung 3: Ergebnis der Abfrage aus Listing 3

Durch die flächendeckende Verbreitung und Nutzung von mobilen Geräten mit GPS Funktion, bietet sich ein immenses Potential, Bewegungsdaten zu nutzen, welche sowohl die räumlichen Informationen in Form von GPS Koordinaten als auch die Zeitstempel direkt enthalten. Auf dieser Basis können ebenfalls sehr einfach relevante Kontakte ermittelt werden. In *Abbildung 4* sind beispielhaft Bewegungsdaten auf einer Karte zu sehen. Der dargestellte Zeitausschnitt (engl.: *Timeslider*) wurde dabei so gewählt, daß er mit dem Ergebnis der SQL Anfrage auf ein fiktives Bewegungsdatenset für die Stadt Boston/USA in *Listing 4* zusammenpaßt. In der Abfrage wird die Funktion GET_ALL_DURATIONS der *Contact Tracing API* verwendet. Diese Funktion unterscheidet sich in einigen Parametern von der zuvor verwendeten Funktion GET_ALL_SWIPE_IO_DURATIONS. Eine Gegenüberstellung findet Ihr deshalb in *Tabelle 1*.

Der Parameter TIME_TOLERANCE_IN_SEC wird in Abhängigkeit vom Zeitintervall für die Erfassung der Datenpunkte gewählt. Werden die Datenpunkte sekündlich erfaßt, dann ist der Standardwert für diesen Parameter 2 (das Doppelte). Der Parameter CHAINING_TOLERANCE_IN_SEC sollte mindestens gleich oder größer dem Wert von TIME_TOLERANCE_IN_SEC sein.

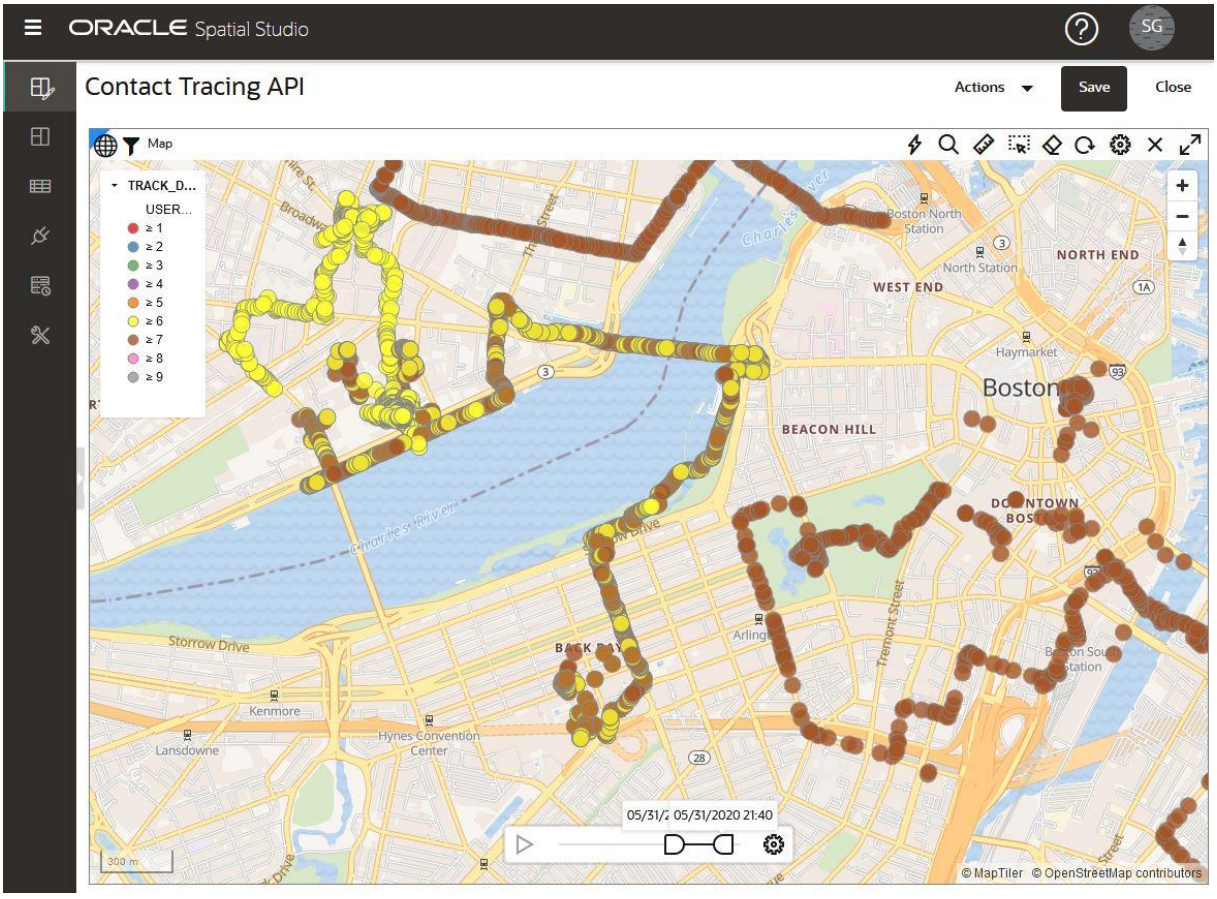

Abbildung 4: Raum- und zeitliche Darstellung von GPS-basierten Bewegungsdaten in Oracle Spatial Studio

```
select a.in_user_id
  ,a.out_user_id
  ,round(a.duration/60,2) duration_in_minutes
  ,to_char(a.start_time,'MM/DD/YY HH24:MI:SS') start_time
  ,to_char(a.end_time, 'MM/DD/YY HH24:MI:SS') end_time
  ,a.num_contact_times
from table(sdo_obj_tracing.get_all_durations(
  user_id                    => 7,
  start_time                 => to_date('31-MAY-2020
00.00.03','DD-MON-YYYY HH24.MI.SS'),
  end_time                   => to_date('31-MAY-2020
23.59.54','DD-MON-YYYY HH24.MI.SS'),
  distance                   => 15,
  time_tolerance_in_sec      => 5,
  chaining_tolerance_in_sec  => 20,
```

```
  track_table_name           => 'osm_tracks_new',
  geom_column_name           => 'geom',
  user_id_column_name        => 'user_id',
  time_column_name           => 'time_adjusted',
  date_as_number_column_name => 'time_adjusted_as_number')) a
where a.segment_or_all = 'ALL'
order by a.in_user_id, a.out_user_id, a.start_time;
```

Listing 4: Summierte Kontaktdauern für Person mit USER_ID = 7 mit anderen Personen

IN_USER_ID	OUT_USER_ID	DURATION_IN_MINUTES	START_TIME	END_TIME	NUM_CONTACT_TIMES
7	5	0.2	05/31/20 18:53:24	05/31/20 18:53:36	1
7	6	86.08	05/31/20 15:26:34	05/31/20 21:42:57	34
7	8	0.07	05/31/20 14:37:02	05/31/20 14:37:06	1

Abbildung 5: Ergebnis der Abfrage aus Listing 4

GET_ALL_SWIPE_IO_DURATIONS	GET_ALL_DURATIONS
USER_ID (NUMBER) START_TIME (DATE) END_TIME (DATE TRACK_TABLE_NAME (VARCHAR2) USER_ID_COLUMN_NAME (VARCHAR2) BUILDING_ID_COLUMN_NAME (VARCHAR2) FLOOR_ID_COLUMN_NAME (VARCHAR2) ROOM_ID_COLUMN_NAME (VARCHAR2) SWIPE_IO_COLUMN_NAME (VARCHAR2) TIME_COLUMN_ NAME (VARCHAR2) MUST_MATCH_COLUMNS (SDO_STRING_ARRAY)	USER_ID (NUMBER) START_TIME (DATE) END_TIME (DATE DISTANCE (NUMBER) TIME_TOLERANCE_IN_SEC (NUMBER) CHAINING_TOLERANCE_IN_SEC (NUMBER) TRACK_TABLE_NAME (VARCHAR2) GEOM_COLUMN_NAME (VARCHAR2) USER_ID_COLUMN_NAME (VARCHAR2) TIME_COLUMN_NAME (VARCHAR2) DATE_AS_NUMBER_COLUMN_NAME (VARCHAR2)

Tabelle 1: Funktionen im PL/SQL Paket SDO_OBJ_TRACING für die beiden Contact Tracing APIs

Die *Contact Tracing APIs* im Paket SDO_OBJ_TRACING sind generisch gehalten. Sie können daher auch für das Nachverfolgen von Begegnungen z.B. sich bewegender Objekte genutzt werden, um zum Beispiel Fragen zu beantworten wie: Welche Objekte wurden zur gleichen Zeit und auf den gleichen Wegabschnitten an einen anderen Ort verbracht?

Location Tracking - Wo fahren Sie denn?

Eine etwas andere, aber ähnliche Form der Betrachung raum- und zeitlicher Verläufe von Daten, ist mit dem Thema *Location Tracking* verbunden. Hierbei liegt der Fokus darauf, zu jedem Zeitpunkt in der Lage zu sein, den Ort für ein Objekt (oder auch eine Person) zu kennen und in Bezug zu ortsabhängigen Aktionen zu setzen. Das ist z.B. für Speditionsunternehmen wichtig, um zu bestimmen, ob sich deren Fahrzeuge entlang der definierten Routen bewegen oder diese verlassen. Die Verarbeitung der dafür notwendigen Positionsdaten basiert in der Regel auf einem *Producer-Consumer*-Prinzip, welches auch *Advanced Queuing* benutzt. Verläßt also ein Fahrzeug die vorgeschriebene Route oder passiert das Fahrzeug eine speziell definierte Region (z.B. Grenzüberfahrt), kann eine Nachricht ausgelöst und an die Zentrale verschickt werden.

Das Hineinbewegen in eine solche Region oder auch das Herausbewegen ist eng mit dem englischen Begriff *Geofencing* verknüpft. *Geo* und *fence* zusammen ergeben einen Zaun für ein geografisch definiertes Gebiet.

Das *Location Tracking* ist ebenfalls direkt in der Datenbank möglich und über das PL/SQL Paket SDO_TRKR inplementiert. Es kann mit jeder Edition der Oracle DB ab Version 12.2 verwendet werden ab. Das gilt auch für die Autonomous Datenbanken ATP und ADW in den Versionen 19c und 21c. Für die Nutzung des Paketes werden einige zusätzliche Rechte für den DB User benötigt, die mit *Advanced Queuing* in Zusammenhang stehen. Diese sind in *Listing 5* aufgeführt.

```
-- Local DB or DBCS: Ausführen als SYS oder SYSTEM
-- Autonomous DB: Ausführen als ADMIN
grant aq_administrator_role, create job, manage scheduler to
testuser;
```

```
grant execute on dbms_aq to testuser;
grant execute on dbms_aqadm to testuser;
grant execute on dbms_lock to testuser;
grant execute on dbms_aqin to testuser;
grant execute on dbms_aqjms to testuser;
grant select_catalog_role to testuser;
```

Listing 5: Notwendige Rechte für den DB User

Wie funktioniert das *Location Tracking* in der Oracle Datenbank?
Die einzelnen Schritte beschreibe ich nachfolgend ausführlicher mit Hilfe der jeweiligen Skriptausschnitte. Das gesamte Skript mit allen Schritten und weiteren zuätzlichen Abfragen kann von GitHub [2] heruntergeladen werden.

Ausgangsbasis für das *Location Tracking* ist ein sogenanntes *Tracking Set*. Dieses erzeugt die notwendigen DB Objekte im Anwenderschema, u.a. Tabellen für die Verwaltung von Nachrichten oder eine Tabelle zum Speichern der Geofences als SDO_GEOMETRY Objekte. *Listing 6* zeigt das Anlegen eines solchen *Tracking Sets*. Alle für ein *Tracking Set* (in meinem Beispiel TS1) angelegten Tabellen sind in *Abbildung 6* zu sehen.

```
begin
  sdo_trkr.create_tracking_set(
    'ts1'
    , 1     -- Anzahl der Queues für die Tracking messages
    , 1);   -- Anzahl der Queues für die Location messages
end;
/
```

Listing 6: Anlegen eines Tracking Sets

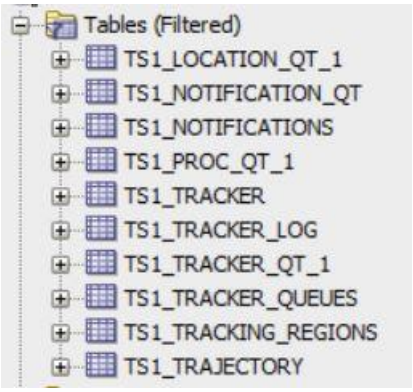

Abbildung 6: Tabellen des Location Tracker Servers für ein Tracking Set

Ist das *Tracking Set* definiert, muß es noch aktiviert werden (*Listing 7*).

```
exec sdo_trkr.start_tracking_set('ts1');

-- Ergebnis überprüfen
-- Queues, die vom Tracking Set verwendet werden
select name
from user_queues
where name like 'TS1%'
order by name;
-- Scheduler Jobs, die vom Tracking Set verwendet werden
select job_name, state
from user_scheduler_jobs
where job_name like'TS%'
order by job_name;
```

Listing 7: Starten des Tracking Sets und Überprüfen der Queues

Danach werden die Regionen als SDO_GEOMETRY Objekte erzeugt. Im *Listing 8* ist eine solche Region innerhalb von Deutschland als einfaches Rechteck mit fünf Stützpunkten definiert.

Hinweis:

Das komplette Skript auf GitHub nimmt eine kleine Änderung zu den Skriptauszügen hier in diesem Artikel vor. Anstelle der Oracle-eigenen Koordinatensystem ID 8307 für das von GPS verwendete Koordinatensystem, WGS 84, soll die von der EPSG[3]

[3] European Petroleum Survey Group

vergebene Koordinatensystem ID 4326 [3] verwendet werden. Das hat zur Folge, daß sowohl die *Oracle Spatial* spezifischen Metadaten (USER_SDO_GEOM_METADATA) anzupassen als auch der räumliche Index neu zu erzeugen sind.

```
insert into ts1_tracking_regions values (
  1,
  mdsys.sdo_geometry(
    2003,
    8307,
    null,
    sdo_elem_info_array(1, 1003, 1),
    sdo_ordinate_array(
       12,51,     -- 1. Stützpunkt
       14,51,
       14,53,
       12,53,
       12,51)));  -- Letzter Stützpunkt identisch mit 1.
commit;
```

Listing 8: Registrieren einer Region

Im nächsten Schritt wird festgelegt, wann für welche Objekte eine Nachricht ausgelöst und in eine Queue geschrieben wird. Im Beispiel in *Listing 9* sollen Nachrichten immer dann generiert und in Queue 1 geschrieben werden, wenn:

- Objekt 1 sich in Region 1 hineinbewegt („I“)
- Objekt 2 sich aus Region 1 herausbewegt („O“)

```
exec sdo_trkr.send_tracking_msg('TS1', mdsys.tracker_msg(1, 1,
'I'));
exec sdo_trkr.send_tracking_msg('TS1', mdsys.tracker_msg(2, 1,
'O'));

-- Überprüfen
select object_id, region_id, alert_when
from ts1_tracker;
```

Listing 9: Definition, wann Nachrichten verschickt werden

Nunmehr sind alle Voraussetzungen erfüllt, um Positionsdaten zu empfangen und auf diese zu reagieren, wenn die Bedingungen hinreichend erfüllt sind. *Listing 10* zeigt acht eingehende Nachrichten zu Positionen der zwei Objekte mit den IDs 1 und 2. Die Parameter drei und vier von MDSYS.LOCATION_MSG übergeben jeweils die Position als X (Längengrad) und Y (Breitengrad) Koordinate.

```
begin
  sdo_trkr.send_location_msgs(
    'TS1',
    mdsys.location_msg_arr(
      mdsys.location_msg(1,CURRENT_TIMESTAMP(),11.5,50.5),
      mdsys.location_msg(1,CURRENT_TIMESTAMP()+1,12.5,51.5),
      mdsys.location_msg(1,CURRENT_TIMESTAMP()+2,13.5,52.5),
      mdsys.location_msg(1,CURRENT_TIMESTAMP()+3,14.5,53.5),
      mdsys.location_msg(2,CURRENT_TIMESTAMP(),13,50.5),
      mdsys.location_msg(2,CURRENT_TIMESTAMP()+1,13,51.5),
      mdsys.location_msg(2,CURRENT_TIMESTAMP()+2,13,52.5),
      mdsys.location_msg(2,CURRENT_TIMESTAMP()+3,13,53.5)));
end;
/
```

Listing 10: Verschicken der Positionsdaten

Im nächsten Schritt werden die Positionsdaten ausgewertet und Nachrichten erzeugt, wenn Objekt 1 sich in die Region hin- oder Objekt 2 sich aus der Region heraus bewegt hat. Dieser Vorgang wird auch *Dequeuing* genannt (*Listing 11*).
Die Nachrichten sind werden in die Tabelle TS1_NOTIFICATIONS geschrieben (*Listing 12*).

```
declare
  message mdsys.notification_msg;
begin
  loop
    sdo_trkr.get_notification_msg(
```

```
      tracking_set_name => 'TS1',
      message => message,
      deq_wait =>2);     -- Min. 2 Sekunden warten
    if (message is null) then
      exit;
    end if;
    insert into ts1_notifications (
      object_id, region_id, time, x, y, state)
      (select
        message.object_id,
        message.region_id,
        message.time,
        message.x,
        message.y,
        message.state
      from sys.dual);
  end loop;
end;
/
```

Listing 11: Dequeuing der Positionsdaten

```
select object_id, region_id, x, y, state
from ts1_notifications
order by object_id, time;

-- Ergebnis:
OBJECT_ID  REGION_ID          X          Y STATE
---------- ---------- ---------- ---------- --------
         1          1       12.5       51.5 INSIDE
         1          1       13.5       52.5 INSIDE
         2          1         13       50.5 OUTSIDE
         2          1         13       53.5 OUTSIDE
```

Listing 12: Notifications Tabelle mit den erzeugten Nachrichten

Visuell kann das Ergebnis nun leicht anhand der *Abbildung 7* überprüft werden. Die Kartendarstellung wurde ebenfalls mit Hilfe von *Spatial Studio* erzeugt. Sie zeigt den *Geofence* (grün) und genau die Positionen der beiden Objekte, welche Benachrichtigungen auslösen. Für Objekt 1 (rot) sind es die Positionen innerhalb, für Objekt 2 (blau) die Positionen außerhalb vom *Geofence*.

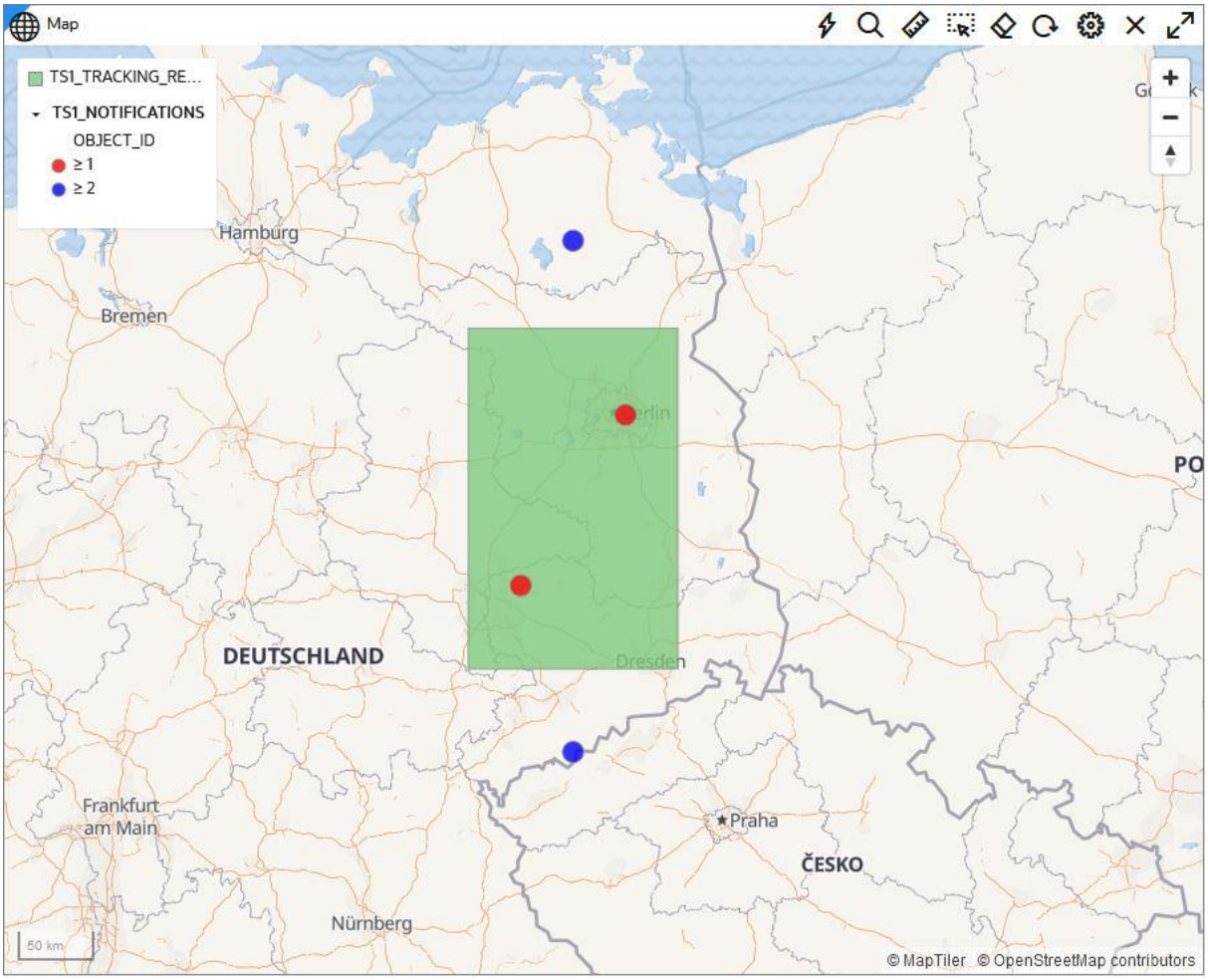

Abbildung 7: Kartendarstellung in Spatial Studio für die Notifications mit Bezug zum Geofence

Nach Bedarf können jederzeit weitere *Geofences* (Regionen) registriert, die Regeln für das Auslösen von Nachrichten geändert oder weitere Queues hinzugefügt werden. Sollen zum Beispiel für das Objekt 1 in Bezug auf die Region 1 keine Benachrichtigungen mehr ausgelöst werden, dann kann diese Zuordung sehr einfach *disabled* werden (*Listing 13*).

```
exec sdo_trkr.send_tracking_msg(
```

```
  'TS1',
  mdsys.tracker_msg(1, 1, 'D'));
```

Listing 13: Notifications Tabelle mit den erzeugten Nachrichten

Im Sinne der Vollständigkeit möchte ich noch auf die beiden Prozeduren SDO_TRKR.STOP_TRACKING_SET sowie SDO_TRKR.DROP_TRACKING_SET hinweisen. Anhand der Namensgebung läßt sich leicht erkennen, wofür diese verwendet werden, nämlich entweder zum Deaktivieren oder zum Löchen eines *Tracking Sets*.

Fazit

Im Artikel habe ich einen Teilbereich der Arbeit mit Geodaten genauer beleuchtet, welcher die Entwicklung von Applikationen für spannende und zugleich gängige Anwendungsfälle unterstützt. Die vollständige Integration der beschriebenen Funktionalität in die Oracle Datenbank vereinfacht sowohl den Betrieb, die Administration als auch die Absicherung der Anwendungen mit den gängigen Werkzeugen bzw. Sicherungsmechanismen der Datenbank. Die Nutzung der beschriebenen Funktionen erfordert weder zusätzliche Lizenzgebühren noch *Cloud Credits*, sondern bietet die Chance, fachliche Anforderungen einfach umzusetzen, ohne „das Rad noch mal neu zu erfinden“.

Quellen

[1] *Contact Tracing* APIs in Oracle Database, https://blogs.oracle.com/oraclespatial/post/contact-tracing-apis-in-oracle-database
[2] GitHub Repository für *Location Tracking*, https://github.com/karinpatenge/location_tracker
[3] WGS 84, https://epsg.io/4326

Dokumentation

- Oracle Database 19c Dokumentation – Spatial, https://docs.oracle.com/en/database/oracle/oracle-database/19/spatial.html

- Oracle Database 21c Dokumentation - Spatial, https://docs.oracle.com/en/database/oracle/oracle-database/21/spatial.html

Über die Autorin

Karin Patenge ist als Produktmanagerin verantwortlich für die Themenbereiche Graph und Spatial Technologien im Bereich der Oracle Datenbank Produktentwicklung. Sie arbeitet seit 2007 für Oracle in Deutschland.
Karin hat sowohl Informatik als auch Geoinformatik an Universitäten in Rumänien, Deutschland und Österreich studiert. Sie hat einen Abschluß als Dipl.-Informatikerin und ist seit über 30 Jahren in unterschiedlichen IT-Rollen für verschiedene Unternehmen in Deutschland tätig.

Kontakt:

Karin Patenge

Oracle Global Services GmbH

Invalidenstr. 65

10557 Berlin

karin.patenge@oracle.com