JD Intelligent Cities Research

The Handbook of JUST-Traj

Authors:
Huajun He

May 31, 2021

JUST V 2.1.0
京东城市时空数据引擎

Login

1. We prepare a test account for the reviewers. UserName: icde2020 Password: icde2020!
2. Enter the login page: http://portal-just.urban-computing.cn/login
3. Enter the *User Name* (icde2020) and *Password* (icde2020!), then click the button *login*, as shown in the following picture.

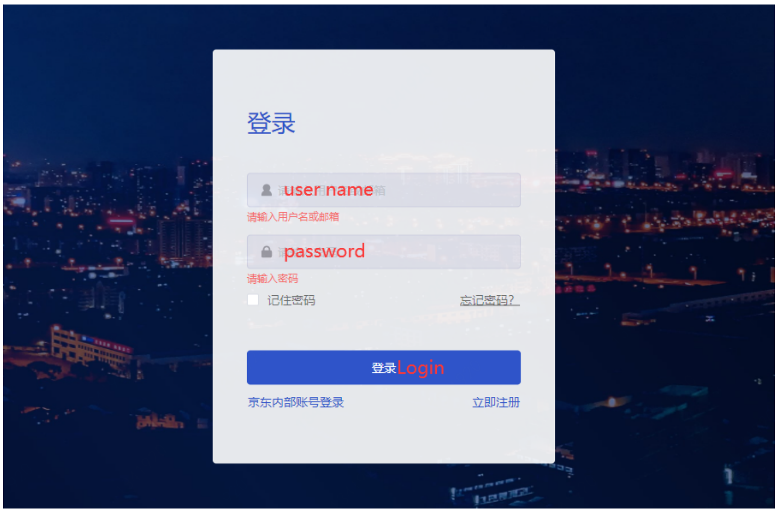
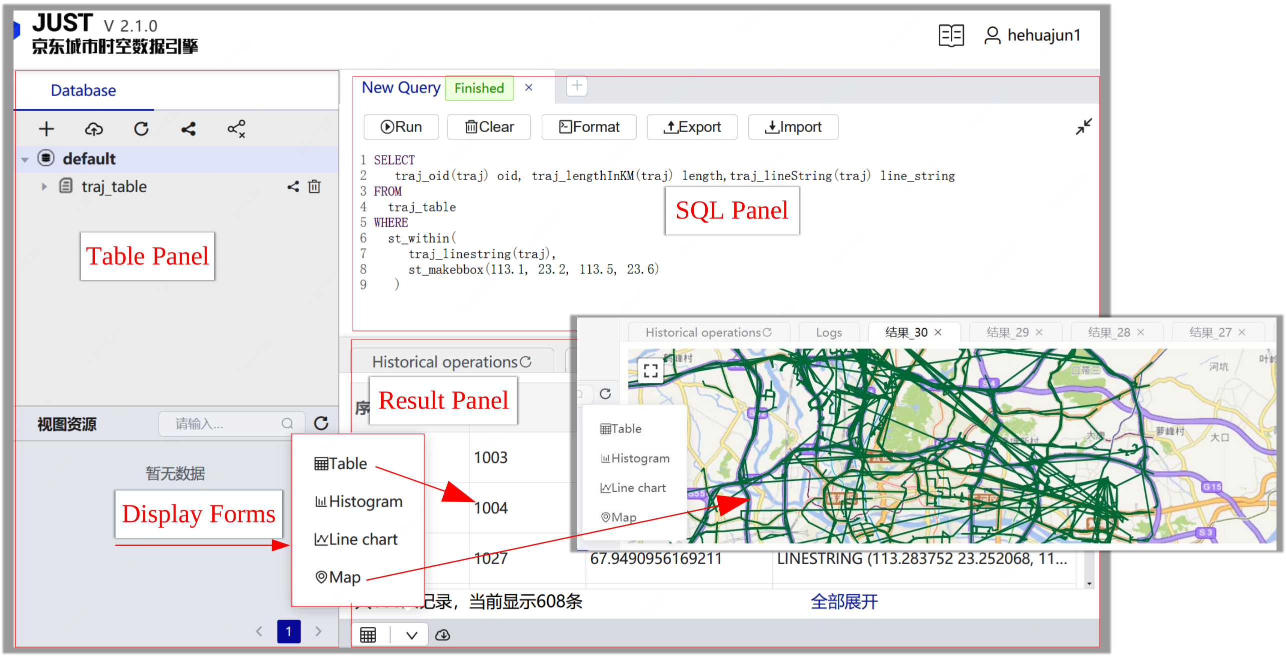


Figure 1: Login.

Web Portal

As shown in Figure 2, the web portal of JUST-Traj has three panels: (1) table panel, which manages the created tables; (2) SQL panel, which provides the SQL editing and execution; and (3) result panel, which visualizes the result by the form of table or map.



```
SELECT
   traj_oid(traj) oid, traj_lengthInKM(traj) length,traj_lineString(traj) line_string
FROM
  traj_table
WHERE
  st_within(
     traj_linestring(traj),
     st_makebbox(113.1, 23.2, 113.5, 23.6)
  )
```

1003
1004
1027 67.9490956169211 LINESTRING (113.283752 23.252068, 11...
记录，当前显示608条 全部展开

Figure 2: The web portal of JUST-Traj.

Create Table

JUST-Traj uses the following statement to create a trajectory table:

```
CREATE TABLE <table name> (<field name> Trajectory);
```

where <field name> is the field name of the trajectory in the database. For example,

```
 CREATE TABLE traj_table (traj Trajectory);
```

Load Data

We can load data from multiple sources into JUST-Traj using the following statement:

```
LOAD <source type>:<file path> TO JUST:<table name>
CONFIG {<the field mapping relationship>};
```

where *source type* could be HDFS, HIVE, KAFAK, FLINK, and the *CONFIG* provides the field mapping between the source and the JUST-Traj table. For example, we load data from HDFS to JUST-Traj.

```
1  LOAD HDFS:'/trajectories' to JUST:traj_table (
2    oid 0,
3    time to_timestamp(3),
4    point st_makePoint(1, 2)
5 );
```

where '/trajectories' is the path of trajectories, lines from 2 to 4 are the field mappings.

Query

JUST-Traj provides spatial or spatio-temporal queries: *spatial range query, spatio-temporal range query, ID temporal query, similarity query, kNN query.*
(1) spatial query

```
SELECT * FROM traj_table
WHERE <spatial relationship>(traj, st_makeBBox(lng1,lat1, lng2, lat2);
```

where, *spatial relationship* could be *st_within*, *st_intersect*, *st_overlap*. *st_makeBBox* is a spatial range formed by two points (lng_1, lat_1) and (lng_2, lat_2). For example,

```
SELECT * FROM traj_table
WHERE st_within(traj_linestring(traj),
      st_makeBBox(113.0, 23.0, 113.5, 23.6));
```

(2) spatio-temporal query

```
SELECT * FROM traj_table
WHERE st_within(traj_linestring(traj),
      st_makeBBox(113.0, 23.0, 113.5, 23.6))
  and traj_startTime(traj) >= '2014-03-13 07:04:51'
  and traj_endTime(traj) <= '2014-03-16 08:04:51';
```

(3) ID-temporal query

```
SELECT * FROM traj_table
WHERE
  traj_oid(traj) = '1003')
  and traj_startTime(traj) >= '2013-07-03 14:33:27'
  and traj_endTime(traj) <= '2018-08-03 14:33:27';
```

(4) Similarity query

```
SELECT * FROM traj_table
WHERE
  traj_oid(traj) = '1003')
  and traj_startTime(traj) >= '2013-07-03 14:33:27'
  and traj_endTime(traj) <= '2018-08-03 14:33:27';
```

(5) kNN query

```
SELECT * FROM traj_table
WHERE
   traj_oid(traj) = '1003')
   and traj_startTime(traj) >= '2013-07-03 14:33:27'
   and traj_endTime(traj) <= '2018-08-03 14:33:27';
```

Analytics

JUST-Traj supports many out-of-the-box analyses on trajectories, which facilitates the development of applications. Five common analyses provided by JUST-Traj, i.e.,
Processing. Although we can pre-process before storing, the parameter of each algorithm could be adjusted when analyzing. Thus, JUST-Traj also supports the processing in the analytics stage;
Aggregation. JUST-Traj provides many aggregation operations, e.g., *max*(), *min*(), *count*();
Stay Point Detection. Moving objects tend to stay due to certain events, such as vehicles staying for refueling, couriers staying for delivery. By analyzing the place that a mobile object stays, we can infer to some places of interesting, e.g., the delivery address;
Clustering. It is one of the basic methods to explore the movement patterns of groups;
Close-contacts tracking. It finds people who had close contact with an abnormal person. It is vital for many applications, e.g., epidemic prevention, companion detection. The SQL statement of DAL is as follows:

```
SELECT <analysis operation>(traj, {<parameters>})
FROM <table name>;
```

where, *analysis operation* is the name of analysis on trajectories, *parameters* set the corresponding parameters. For example,

```
SELECT st_trajStayPoint(traj,
   '{ "maxStayDistInMeter": 10,
      "minStayTimeInSecond": 60}')
FROM
   traj_table;
```

A Holistic Solution

Stay Point Detection

In this scenario, we detect stay points from the spatio-temporal query result. We define the location where the lorry stays over a given time threshold (*minStayTimeInSecond*) as a stay point, where the location is a spatial region whose maximum distance not greater than a distance threshold (*maxStayDisInMeter*). The underlying locations of stay points could be the delivery addresses. The SQL is as follows:

```
1  SELECT st_trajStayPoint(traj,
2     '{ "maxStayDistInMeter": 10,
3        "minStayTimeInSecond": 60}')
4  FROM
5     traj_table
6  WHERE
7     st_within(traj_linestring(traj),
8          st_makeBBox(113.0, 23.0, 113.5, 23.6))
9     and traj_startTime(traj) >= '2014-03-13 07:04:51'
10    and traj_endTime(traj) <= '2014-03-16 08:04:51';
```

Lines from 7 to 10 take a spatio-temporal range to query trajectories from the database. Lines from 1 to 3 execute the *Stay Point Detection* operation on the extracted trajectories, where lines from 2 to 3 are parameters of *Stay Point Detection*.

Noise Filtering

In this scenario, we define the point whose speed exceeds the maximum limited speed (*maxSpeedMeterPerSecond*) as the noise point, and we filter that point. The SQL is as follows:

```
1 SELECT st_trajNoiseFilter(traj,
2   '{ "maxSpeedMeterPerSecond": 20.0}')
3 FROM
4   traj_table
5 WHERE
6   traj_oid(traj) = '1197404443')
7   and traj_startTime(traj) >= '2018-07-03 14:33:27'
8   and traj_endTime(traj) <= '2018-08-03 14:33:27';
```

Lines from 7 to 9 take a *ID Temporal* query to extract trajectories from the database. Lines from 1 to 3 execute the *Noise Filtering* operation on the extracted trajectories.