Université libre de Bruxelles
Faculté des Sciences Appliquées

January 23, 2024

INFO-H-415 – Advanced databases

First session examination

A shipping company put a database in place to keep track of their boats. An inventory of all the past and currently used shipping lanes is kept (i.e., routes taken by boat from one port to another). When moving, each boat has its real position updated every hour. The company also keeps track of the crew present on its boat. Each crew member has a role on the boat that can change throughout her career.

Next is an excerpt of the relational schema used.

- **Boat** (boatId, fromDate, toDate, name, capacity)
  - fromDate is the commissioning date of the boat
  - toDate is the decommissioning date (it is NULL if the boat is still in service)
- **ShippingLane** (laneId, geom)
  - geom is a MULTILINE geometry
- **Position** (boatId, time, laneId, geom)
  - boatId references Boat.boatId
  - laneId references ShippingLane.laneId
  - time is a timestamptz datatype
  - geom is a POINT geometry
- **Crew** (crewId, firstName, lastName, aStreet, aCity, aZip, aCountry, telephone)
- **Managed** (crewId, fromDate, toDate, manager)
  - crewId references Crew.crewId
  - manager references Crew.crewId
  - fromDate is the date upon which the crew member was put under her manager
  - toDate is the date upon which he stopped being under her manager
- **Role** (crewId, fromDate, toDate, role)
  - crewId references Crew.crewId
  - fromDate marks the start of the role of a crew member
  - toDate marks the end of the role
- **PartOf** (crewId, fromDate, toDate, boatId)
  - crewId references Crew.crewId
  - boatId references Boat.boatId
  - fromDate is the date the crew member has arrived on the boat
  - toDate is the date when the crew member left

1 Spatial Databases

For the following questions, suppose you are using a PostgreSQL database. We also suppose that the geometries are in the SRID 4326 (WGS84) spatial reference system.

a. List the name of boats that have deviated from their shipping lane by more than a kilometre as well as the most recent time when they did.
b. For each boat, find how far it has gotten on its latest shipping lane. This progress will be presented as a percentage of the total length of the shipping lane.
c. For each boat, count how many other boats it has crossed so far (we suppose two boats have crossed each other if they have been ever less than 500 meters from each other).
d. Find the shipping lanes that cross the shipping lane where the oldest boat (that is still running) has last been.

2 Temporal Databases

a. Give the role and boat history of the crew members.
b. Give the name and last role of crew members that are not on any boat at this time (suppose that if a crew member is currently on a boat, the toDate of its entry in the PartOf table is NULL).
c. Give the history of the boat with the highest number of crew members (do not coalesce the results).
d. Coalesce the result of your previous answer.

3 Active Databases

For the following questions, suppose you are using a SQLServer database.

a. A shipping lane cannot be fully contained in another shipping lane.
b. Only active ships (i.e., ships that are not decommissioned) can perform trips and send their position.
c. At each point in time a crew member has a single role.
d. There cannot be cycles in the manager relationship represented in table Managed.

4 Mobility Databases

For this section we will use another database implemented using MobilityDB. This database contains the trips various cars have been doing within Brussels. The database has information about the communes in Brussels as well as various points of interest within those communes. It is supposed that the geometries are in the SRID 3812 (ETRS89 / Belgian Lambert 2008).

- **Vehicles** (vehId, licence, type)
- **Trips** (tripId, vehId, day, trip)
  - tripId is the identifier of the trip
  - vehId references Vehicle.vehId
  - day is the date of the trip
  - trip is a MobilityDB type `tgeompoint`
- **Communes** (communeId, name, population, geom)
  - geom is the geometry of the commune
- **Points** (pointId, geom)
  - geom is a geometry POINT representing the location of the point of interest

a. Give the first time when each vehicle visited a point of interest in table Points.
b. Give the minimal distance between each pair of vehicles whenever their time intervals overlap.
c. Give for each vehicle the duration of the trip of maximum length over all of its trips. The result should show three columns: vehId, duration of the trip, trajectory of the trip. If a vehicle has multiple trips with the same maximum length, it must appear multiple times in the result set (one for each duration time).
d. For every pair of adjacent communes, give the trips that cross each of them. You should only show the trips restricted by these boundaries and the time when the intersection occurred.

You can use the following PostGIS functions:

- **ST_Area(geometry)**: Return the area of the geometry if it is a Polygone or Multipolygone.
- **ST_Centroid(geometry)**: Return the geometric center of a geometry
- **ST_Contains(geometry,geometry)**: Returns true if the first geometry contains the second one
- **ST_Distance(geomA, geomB)**: Return the 2D Cartesian distance between two geometries
- **ST_DumpPoints(geometry)**: Return a set of all points that make up a geometry
- **ST_Intersection(geomA, geomB)**: Return a geometry that represents the shared portion of geomA and geomB.
- **ST_Intersects(geomA, geomB)**: Return TRUE if the Geometries share any portion of space and FALSE if they do not.
- **ST_Length(geometry)**: Return the length of the geometry if it is a Line or MultiLine.
- **ST_LineLocatePoint(geomA, geomB)**: Return a float between 0 and 1 representing the location of the closest point on a line geomA to the given Point geomB, as a fraction of 2d line length.
- **ST_LineInterpolatePoint(geomA, fraction)**: Return a point interpolated along a line geomA at a fractional location.
- **ST_Segmentize(geometry)**: Return a modified geometry having no segment longer than the given distance
- **ST_Touches(geometry, geometry)**: Return true if two geometries have at least one point in common, but their interiors do not intersect
- **ST_Union(geometry)**: Aggregating function, return a geometry that represents the point set union of the geometries.
- **ST_Value(geometry, raster)**: Return the value of a given band of a raster at a given geometry point.

You can use the following MobilityDB functions:

- Return the lower or upper bound
  - lower(spans) → base
  - upper(spans) → base
- Return the value or time span ignoring the potential gaps
  - valueSpan(tnumber) → numspan
  - timeSpan(ttype) → tstzspan
- Return the trajectory
  - trajectory(tpoint) → geo
- Return the start, end, or n-th timestamp
  - startTimestamp(ttype) → timestamptz
  - endTimestamp(ttype) → timestamptz
  - timestampN(ttype,integer) → timestamptz
- Restrict to (the complement of) a set of values
  - atValues(ttype,values) → ttype
  - minusValues(ttype,values) → ttype
- Return the duration
  - duration({datespan,tstzspan}) → interval
  - duration({datespanset,tstzspanset},boundspan bool=false) → interval
- Return the smallest distance ever
  - {geo,tpoint} |=| {geo,tpoint} → float
- Return the temporal distance
  - {point,tpoint} <-> {point,tpoint} → tfloat
- Restrict to (the complement of) a geometry and a Z span
  - atGeometry(tgeompoint,geometry,zspan=NULL) → tgeompoint
  - minusGeometry(tgeompoint,geometry,zspan=NULL) → tgeompoint