Advanced DB Exam 2020

January 23, 2023

1 Temporal and Spatial DB

1) Give the period during which user with UID 1 was living in the commune called Ixelles but did not have any subscription valid for this commune.

```
-- Case 4
SELECT UA.StartDate AS StartDate, UA.EndDate AS EndDate
FROM UserAddress UA, Commune C
WHERE UA.UID = 1 AND ST_Intersects(UA.Point, C.Geom) AND C.CommuneName = 'Ixelles'
  AND NOT EXISTS(
    SELECT *
    FROM Subscription S, CommuneSubscription CS
    WHERE S.UID = 1 AND CS.CommuneName = 'Ixelles' AND CS.SID = S.SID
      AND ((S.StartDate >= UA.StartDate AND S.StartDate < UA.EndDate)
        OR
        (S.EndDate > UA.StartDate AND S.EndDate <= UA.EndDate))

UNION
-- Case 1
SELECT UA.StartDate AS StartDate, S.StartDate AS EndDate
FROM UserAddress UA, Commune C, Subscription S, CommuneSubscription CS
WHERE UA.UID = 1 AND ST_Intersects(UA.Point, C.Geom) AND C.CommuneName = 'Ixelles'
  AND S.UID = 1 AND CS.CommuneName = 'Ixelles' AND CS.SID = S.SID
  AND UA.StartDate < S.StartDate AND S.StartDate < UA.EndDate
  AND NOT EXISTS (
    SELECT *
    FROM Subscription S2, CommuneSubscription CS2
    WHERE S2.UID = 1 AND CS.CommuneName = 'Ixelles' AND CS.SIS = S.SID
      AND UA.StartDate < S2.EndDate AND S2.FromDate < S.EndDate)

UNION
-- Case 2
SELECT S.EndDate AS StartDate, UA.EndDate AS EndDate
FROM UserAddress UA, Commune C, Subscription S, CommuneSubscription CS
WHERE UA.UID = 1 AND ST_Intersects(UA.Point, C.Geom) AND C.CommuneName = 'Ixelles'
  AND S.UID = 1 AND CS.CommuneName = 'Ixelles' AND CS.SID = S.SID
  AND UA.StartDate < S.EndDate AND S.EndDate < UA.EndDate
  AND NOT EXISTS (
    SELECT *
    FROM Subscription S2, CommuneSubscription CS2
    WHERE S2.UID = 1 AND CS.CommuneName = 'Ixelles' AND CS.SID = S.SID
      AND S.EndDate < S2.EndDate AND S2.FromDate < UA.EndDate)

UNION
--Case 3
SELECT S1.EndDate AS StartDate, S2.StartDate AS EndDate
FROM UserAddress UA, Commune C, Subscription S1, Subscription S2, Commune Subscription CS1,
    Commune Subscription CS2
WHERE UA.UID = 1 AND ST_Intersects(UA.Point, C.Geom) AND C.CommuneName = 'Ixelles'
  AND S1.UID = 1 AND CS1.CommuneName = 'Ixelles' AND CS1.SID = S1.SID
  AND S2.UID = 1 AND CS2.CommuneName = 'Ixelles' AND CS2.SID = S2.SID
  AND UA.StartDate < S1.EndDate AND S1.EndDate < UA.EndDate
  AND UA.StartDate < S2.StartDate AND S2.StartDate < UA.EndDate
  AND NOT EXISTS(
    SELECT *
    FROM Subscription S3, CommuneSubscription CS3
```

```
    WHERE S3.UID = 1 AND CS3.CommuneName = 'Ixelles' AND CS3.SID = S.SID
      AND S1.ToDate < S3.ToDate AND S3.FromDate < S2.FromDate)
```

2) Give the history of the communes for which there was the most subscriptions. Do not coalesce the results.

```
CREATE VIEW SubsChanges(Day) AS
  SELECT DISTINCT StartDate
  FROM Subscription

  UNION

  SELECT DISTINCT EndDate
  FROM Subscription

CREATE VIEW SubsPeriods(StartDate, EndDate) AS
  SELECT C1.Day, C2.Day
  FROM SubsChanges C1, SubsChanges C2
  WHERE C1.Day < C2.Day
    AND NOT EXISTS(
      SELECT *
      FROM SubsChanges C3
      WHERE C1.Day < C3.Day AND C3.Day < C2.Day)

CREATE VIEW TempTotal(CS.CommuneName, TotalSubs, StartDate, EndDate) AS
  SELECT CS.CommuneName, COUNT(DISTINCT UID), P.StartDate, P.EndDate
  FROM Subscription S, CommuneSubscription CS, SubsPeriods P
  WHERE S.SID = CS.SID
    AND S.StartDate <= P.StartDate AND P.EndDate <= S.EndDate
  GROUP BY CS.CommuneName, P.StartDate, P.EndDate

CREATE VIEW TempMax(MaxSubs, StartDate, EndDate) AS
  SELECT MAX(TotalSubs), StartDate, EndDate
  FROM TempTotal
  GROUP BY StartDate, EndDate

CREATE VIEW TempResult(CommuneName, MaxSubs, StartDate, EndDate) AS
  SELECT TT.CommuneName, TM.MaxSubs, StartDate, EndDate
  FROM TempTotal TT, TempMax TM
  WHERE TT.TotalSubs = TM.MaxSubs AND TT.StartDate = TM.StartDate AND TT.EndDate = TM.EndDate
```

3) For each user, provide for each of the commune the coalesced history for which the user had an active subscription in this commune.

```
CREATE VIEW TempJoin(UID, CommuneName, StartDate, EndDate) AS
  SELECT UA.UID, CS.CommuneName, UA.StartDate, UA.EndDate
  FROM UserAddress UA, Subscription S, CommuneSubscription CS
  WHERE S.UID = UA.UID AND S.SID = CS.SID
  AND S.StartDate < UA.StartDate AND UA.EndDate <= S.EndDate

  UNION ALL

  SELECT UA.UID, CS.CommuneName, UA.StartDate, S.EndDate
  FROM UserAddress UA, Subscription S, CommuneSubscription CS
  WHERE S.UID = UA.UID AND S.SID = CS.SID
    AND S.StartDate <= UA.StartDate AND UA.StartDate < S.EndDate AND UA.EndDate > S.EndDate

  UNION ALL

  SELECT UA.UID, CS.CommuneName, S.StartDate, UA.EndDate
  FROM UserAddress UA, Subscription S, CommuneSubscription CS
  WHERE S.UID = UA.UID AND S.SID = CS.SID
    AND UA.StartDate <= S.StartDate AND S.StartDate < UA.ToDate AND S.EndDate > UA.EndDate

  UNION ALL

  SELECT UA.UID, CS.CommuneName, S.StartDate, S.EndDate
  FROM UserAddress UA, Subscription S, CommuneSubscription CS
  WHERE S.UID = UA.UID AND S.SID = CS.SID
    AND UA.StartDate < S.StartDate AND S.EndDate <= UA.EndDate

SELECT DISTINCT F.UID, F.CommuneName, F.StartDate, L.EndDate
```

```
FROM TempJoin F, TempJoin L
WHERE F.StartDate < L.EndDate AND F.UID = L.UID AND F.CommuneName = L.CommuneName
  AND NOT EXISTS(
    SELECT *
    FROM TempJoin T
    WHERE T.UID = F.UID AND T.CommuneName = F.CommuneName
      AND F.StartDate < T.StartDate AND T.StartDate < L.EndDate
      AND NOT EXISTS(
        SELECT * FROM TempJoin as T1
        WHERE T1.UID = F.UID AND T1.CommuneName = F.CommuneName
          AND T1.StartDate < F.StartDate AND T.StartDate <= T1.EndDate ))
  AND NOT EXISTS(
    SELECT *
    FROM TempJoin T2
    WHERE T2.UID = F.UID AND T2.CommuneName = F.CommuneName
      AND (
        (T2.StartDate < F.StartDate AND F.StartDate <= T2.EndDate)
        OR
        (T2.StartDate >= L.EndDate AND L.EndDate < T2.EndDate)
        ))
```

4) Give for each region, the number of users which were living in this region during the complete [01/01/2000,01/01/2001] period. User that moved from one address to another (both addresses in the same region) should be counted. You should assume that the different addresses of any user form a continuous and uninterrupted lifecycle (if a user has a registered address at a time instant A and another or the same registered address at a later time instant B, then he also has a registered address for each time instant C such that $C \in [A, B]$). Furthermore, a user cannot have two different addresses at the same time. We suppose you can compare time points to strings of the form "dd/mm/yyyy" using standard orginal comparators (>,<,=,...).

```
CREATE VIEW Temp_User_Region(UID, RegionName, StartDate, EndDate) AS
  SELECT UA.UID, C.RegionName, UA.StartDate, UA.EndDate
  FROM UserAddress UA, Commune C
  WHERE ST_Intersects(UA.Point, C.Geom)

CREATE VIEW Temp_U_R_Coal(UID, RegionName, StartDate, EndDate) AS
  SELECT DISTINCT F.UID, F.CommuneName, F.StartDate, L.EndDate
  FROM Temp_User_Region F, Temp_User_Region L
  WHERE F.StartDate < L.EndDate AND F.UID = L.UID AND F.CommuneName = L.CommuneName
    AND NOT EXISTS(
      SELECT *
      FROM Temp_User_Region T
      WHERE T.UID = F.UID AND T.CommuneName = F.CommuneName
        AND F.StartDate < T.StartDate AND T.StartDate < L.EndDate
        AND NOT EXISTS(
          SELECT * FROM Temp_User_Region as T1
          WHERE T1.UID = F.UID AND T1.CommuneName = F.CommuneName
            AND T1.StartDate < F.StartDate AND T.StartDate <= T1.EndDate ))
    AND NOT EXISTS(
      SELECT *
      FROM Temp_User_Region T2
      WHERE T2.UID = F.UID AND T2.CommuneName = F.CommuneName
        AND (
          (T2.StartDate < F.StartDate AND F.StartDate <= T2.EndDate)
          OR
          (T2.StartDate >= L.EndDate AND L.EndDate < T2.EndDate)
          ))

SELECT RegionName, COUNT(UID)
FROM Temp_User_Region
WHERE StartDate <= '01/01/2000' AND '01/01/2001' <= EndDate
GROUP BY RegionName
```

5) List the different regions with their total area

```
SELECT RegionName, SUM(ST_AREA(Geom))
FROM Commune
GROUP BY RegionName
```

6) For each commune, provide the distance between its centroid and the centroid of the capital of the region they are in.

```
SELECT C.CommuneName, C.RegionName, ST_Distance(ST_Centroid(C.Geom), ST_Centroid(Cap.Geom)) AS
    DistToCapital
FROM Commune C, Region R, Commune Cap
WHERE C.RegionName = R.RegionName AND R.Capital = Cap.CommuneName
```

7) Give the car trip (CID and StartDate time) which has performed the longest segment in the commune of Ixelles.

```
SELECT CT.CID, CT.StartTime, ST_Length(ST_Intersection(ST.Itinerary, C.Geom) AS Length
FROM CarTrip CT, Commune C
WHERE C.CommuneName = 'Ixelles' AND ST_Intersects(CT.Itinerary, C.Geom)
ORDER BY ST_Length(ST_Intersection(ST.Itinerary, C.Geom)
LIMIT 1
```

8) List for each trip the altitude of the lowest and highest point of the trip.

```
SELECT CT.CID, (stats).max highest, (stats).min lowest
FROM (
  SELECT CT.CID, ST_SummaryStats(ST_Clip(C.altitude, 1, CT.Itinerary, TRUE)) AS stats
  FROM Country C JOIN CarTrip CT ON ST_Intersects(CT.Itinerary, C.altitude)) AS tmp
```

2 Active DB

9) Ensure with a trigger that at any instant a user has a single address

```
CREATE TRIGGER PK_User_Addr ON UserAddress FOR INSERT, UPDATE AS
IF EXISTS(
  SELECT *
  FROM UserAddress UA1
  WHERE 1 < (
    SELECT * COUNT(UA2.UID)
    FROM UserAddress UA2
    WHERE UA1.UID = UA2.UID
      AND UA1.StartDate < UA2.EndDate AND UA1.EndDate > UA2.StartDate)
BEGIN
  RAISERROR('A user can only have one address at a time',1,2)
  ROLLBACK TRANSACTION
END
```

10) Ensure with a trigger that at any instant a user has a single subscription to a commune.

```
-- If a user can only be subscribe to one commune at a time
CREATE TRIGGER PK_User_Subs ON Subscription FOR INSERT, UPDATE AS
IF EXISTS(
  SELECT *
  FROM Subscription S1
  WHERE 1 < (
    SELECT COUNT(S2.UID)
    FROM Subscription S2
    WHERE S1.UID = S2.UID
      AND S1.StartDate < S2.EndDate AND S1.EndDate > S2.StartDate)
BEGIN
  RAISERROR('A user can only have one subscription at a time',1,2)
  ROLLBACK TRANSACTION
END

-- If a user can be subscribed to several communes at a time, but only once at each
CREATE TRIGGER PK_User_Subs ON Subscription FOR INSERT, UPDATE AS
IF EXISTS(
  SELECT *
  FROM Subscription S1
  WHERE 1 < (
    SELECT COUNT(S2.UID)
    FROM Subscription S2
    WHERE S1.UID = S2.UID AND S1.SID = S2.SID
```

```
      AND S1.StartDate < S2.EndDate AND S1.EndDate > S2.StartDate)
BEGIN
  RAISERROR('A user can only have one subscription at a time',1,2)
  ROLLBACK TRANSACTION
END
```

11) Ensure with a trigger that the communes in table Commune do not overlap.

```
CREATE TRIGGER NonOverlap_Comm ON Commune FOR INSERT, UPDATE AS
IF EXISTS(
  SELECT *
  FROM Commune C, Inserted I
  WHERE C.CommuneName <> I.CommuneName
    AND ST_Intersects(ST_Interior(C.geom),ST_Interior(I.Geom))
BEGIN
  RAISERROR('Two communes cannot overlap',1,2)
  ROLLBACK TRANSACTION
END
```

12) Ensure with a trigger that the geometry of a Country is equal to the union of its composing communes.

```
CREATE TRIGGER CountryUnionComm ON Commune FOR INSERT, UPDATE AS
IF EXISTS(
  SELECT ST_DIFFERENCE(Com.Geom, C.Geom)
  FROM Country C
    JOIN Region R ON C.CountryName = R.Country
    JOIN Commune Com ON Com.RegionName = R.RegionName

  UNION

  SELECT ST_DIFFERENCE(C.Geom, ST_UNION(Com.Geom))
  FROM Country C
    JOIN Region R ON C.CountryName = R.Country
    JOIN Commune Com ON Com.RegionName = R.RegionName
  GROUP BY C.CountryName)
BEGIN
  RAISERROR('A country must be equal to the union of its composing communes',1,2)
  ROLLBACK TRANSACTION
END

CREATE TRIGGER CountryUnionComm2 ON Country FOR INSERT, UPDATE AS
IF EXISTS(
  SELECT ST_DIFFERENCE(Com.Geom, C.Geom)
  FROM Country C
    JOIN Region R ON C.CountryName = R.Country
    JOIN Commune Com ON Com.RegionName = R.RegionName

  UNION

  SELECT ST_DIFFERENCE(C.Geom, ST_UNION(Com.Geom))
  FROM Country C
    JOIN Region R ON C.CountryName = R.Country
    JOIN Commune Com ON Com.RegionName = R.RegionName
  GROUP BY C.CountryName)
BEGIN
  RAISERROR('A country must be equal to the union of its composing communes',1,2)
  ROLLBACK TRANSACTION
END
```