

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.

```
1  -- Case 4
2  SELECT UA.StartDate AS StartDate, UA.EndDate AS EndDate
3  FROM UserAddress UA, Commune C
4  WHERE UA.UID = 1 AND ST_Intersects(UA.Point, C.Geom) AND C.CommuneName = 'Ixelles'
5  AND NOT EXISTS (
6      SELECT *
7      FROM Subscription S, CommuneSubscription CS
8      WHERE S.UID = 1 AND CS.CommuneName = 'Ixelles' AND CS.SID = S.SID
9      AND ((S.StartDate >= UA.StartDate AND S.StartDate < UA.EndDate)
10         OR
11         (S.EndDate > UA.StartDate AND S.EndDate <= UA.EndDate))
12
13 UNION
14 -- Case 1
15 SELECT UA.StartDate AS StartDate, S.StartDate AS EndDate
16 FROM UserAddress UA, Commune C, Subscription S, CommuneSubscription CS
17 WHERE UA.UID = 1 AND ST_Intersects(UA.Point, C.Geom) AND C.CommuneName = 'Ixelles'
18 AND S.UID = 1 AND CS.CommuneName = 'Ixelles' AND CS.SID = S.SID
19 AND UA.StartDate < S.StartDate AND S.StartDate < UA.EndDate
20 AND NOT EXISTS (
21     SELECT *
22     FROM Subscription S2, CommuneSubscription CS2
23     WHERE S2.UID = 1 AND CS2.CommuneName = 'Ixelles' AND CS2.SID = S.SID
24     AND UA.StartDate < S2.EndDate AND S2.FromDate < S.EndDate)
25
26 UNION
27 -- Case 2
28 SELECT S.EndDate AS StartDate, UA.EndDate AS EndDate
29 FROM UserAddress UA, Commune C, Subscription S, CommuneSubscription CS
30 WHERE UA.UID = 1 AND ST_Intersects(UA.Point, C.Geom) AND C.CommuneName = 'Ixelles'
31 AND S.UID = 1 AND CS.CommuneName = 'Ixelles' AND CS.SID = S.SID
32 AND UA.StartDate < S.EndDate AND S.EndDate < UA.EndDate
33 AND NOT EXISTS (
34     SELECT *
35     FROM Subscription S2, CommuneSubscription CS2
36     WHERE S2.UID = 1 AND CS2.CommuneName = 'Ixelles' AND CS2.SID = S.SID
37     AND S.EndDate < S2.EndDate AND S2.FromDate < UA.EndDate)
38
39 UNION
40 -- Case 3
41 SELECT S1.EndDate AS StartDate, S2.StartDate AS EndDate
42 FROM UserAddress UA, Commune C, Subscription S1, Subscription S2, Commune Subscription CS1,
43      Commune Subscription CS2
44 WHERE UA.UID = 1 AND ST_Intersects(UA.Point, C.Geom) AND C.CommuneName = 'Ixelles'
45 AND S1.UID = 1 AND CS1.CommuneName = 'Ixelles' AND CS1.SID = S1.SID
46 AND S2.UID = 1 AND CS2.CommuneName = 'Ixelles' AND CS2.SID = S2.SID
47 AND UA.StartDate < S1.EndDate AND S1.EndDate < UA.EndDate
48 AND UA.StartDate < S2.StartDate AND S2.StartDate < UA.EndDate
49 AND NOT EXISTS (
50     SELECT *
51     FROM Subscription S3, CommuneSubscription CS3
```

```

51 WHERE S3.UID = 1 AND CS3.CommuneName = 'Ixelles' AND CS3.SID = S.SID
52 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.

```

1 CREATE VIEW SubsChanges(Day) AS
2 SELECT DISTINCT StartDate
3 FROM Subscription
4
5 UNION
6
7 SELECT DISTINCT EndDate
8 FROM Subscription
9
10 CREATE VIEW SubsPeriods(StartDate, EndDate) AS
11 SELECT C1.Day, C2.Day
12 FROM SubsChanges C1, SubsChanges C2
13 WHERE C1.Day < C2.Day
14 AND NOT EXISTS(
15     SELECT *
16     FROM SubsChanges C3
17     WHERE C1.Day < C3.Day AND C3.Day < C2.Day)
18
19 CREATE VIEW TempTotal(CS.CommuneName, TotalSubs, StartDate, EndDate) AS
20 SELECT CS.CommuneName, COUNT(DISTINCT UID), P.StartDate, P.EndDate
21 FROM Subscription S, CommuneSubscription CS, SubsPeriods P
22 WHERE S.SID = CS.SID
23 AND S.StartDate <= P.StartDate AND P.EndDate <= S.EndDate
24 GROUP BY CS.CommuneName, P.StartDate, P.EndDate
25
26 CREATE VIEW TempMax(MaxSubs, StartDate, EndDate) AS
27 SELECT MAX(TotalSubs), StartDate, EndDate
28 FROM TempTotal
29 GROUP BY StartDate, EndDate
30
31 CREATE VIEW TempResult(CommuneName, MaxSubs, StartDate, EndDate) AS
32 SELECT TT.CommuneName, TM.MaxSubs, StartDate, EndDate
33 FROM TempTotal TT, TempMax TM
34 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.

```

1 CREATE VIEW TempJoin(UID, CommuneName, StartDate, EndDate) AS
2 SELECT UA.UID, CS.CommuneName, UA.StartDate, UA.EndDate
3 FROM UserAddress UA, Subscription S, CommuneSubscription CS
4 WHERE S.UID = UA.UID AND S.SID = CS.SID
5 AND S.StartDate < UA.StartDate AND UA.EndDate <= S.EndDate
6
7 UNION ALL
8
9 SELECT UA.UID, CS.CommuneName, UA.StartDate, S.EndDate
10 FROM UserAddress UA, Subscription S, CommuneSubscription CS
11 WHERE S.UID = UA.UID AND S.SID = CS.SID
12 AND S.StartDate <= UA.StartDate AND UA.StartDate < S.EndDate AND UA.EndDate > S.EndDate
13
14 UNION ALL
15
16 SELECT UA.UID, CS.CommuneName, S.StartDate, UA.EndDate
17 FROM UserAddress UA, Subscription S, CommuneSubscription CS
18 WHERE S.UID = UA.UID AND S.SID = CS.SID
19 AND UA.StartDate <= S.StartDate AND S.StartDate < UA.ToDate AND S.EndDate > UA.EndDate
20
21 UNION ALL
22
23 SELECT UA.UID, CS.CommuneName, S.StartDate, S.EndDate
24 FROM UserAddress UA, Subscription S, CommuneSubscription CS
25 WHERE S.UID = UA.UID AND S.SID = CS.SID
26 AND UA.StartDate < S.StartDate AND S.EndDate <= UA.EndDate
27
28 SELECT DISTINCT F.UID, F.CommuneName, F.StartDate, L.EndDate

```

```

29 FROM TempJoin F, TempJoin L
30 WHERE F.StartDate < L.EndDate AND F.UID = L.UID AND F.CommuneName = L.CommuneName
31 AND NOT EXISTS(
32     SELECT *
33     FROM TempJoin T
34     WHERE T.UID = F.UID AND T.CommuneName = F.CommuneName
35     AND F.StartDate < T.StartDate AND T.StartDate < L.EndDate
36     AND NOT EXISTS(
37         SELECT * FROM TempJoin as T1
38         WHERE T1.UID = F.UID AND T1.CommuneName = F.CommuneName
39         AND T1.StartDate < F.StartDate AND T.StartDate <= T1.EndDate ))
40 AND NOT EXISTS(
41     SELECT *
42     FROM TempJoin T2
43     WHERE T2.UID = F.UID AND T2.CommuneName = F.CommuneName
44     AND (
45         (T2.StartDate < F.StartDate AND F.StartDate <= T2.EndDate)
46         OR
47         (T2.StartDate >= L.EndDate AND L.EndDate < T2.EndDate)
48     ))

```

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 original comparators ($>$, $<$, $=$, ...).

```

1 CREATE VIEW Temp_User_Region(UID, RegionName, StartDate, EndDate) AS
2 SELECT UA.UID, C.RegionName, UA.StartDate, UA.EndDate
3 FROM UserAddress UA, Commune C
4 WHERE ST_Intersects(UA.Point, C.Geom)
5
6 CREATE VIEW Temp_U_R_Coal(UID, RegionName, StartDate, EndDate) AS
7 SELECT DISTINCT F.UID, F.CommuneName, F.StartDate, L.EndDate
8 FROM Temp_User_Region F, Temp_User_Region L
9 WHERE F.StartDate < L.EndDate AND F.UID = L.UID AND F.CommuneName = L.CommuneName
10 AND NOT EXISTS(
11     SELECT *
12     FROM Temp_User_Region T
13     WHERE T.UID = F.UID AND T.CommuneName = F.CommuneName
14     AND F.StartDate < T.StartDate AND T.StartDate < L.EndDate
15     AND NOT EXISTS(
16         SELECT * FROM Temp_User_Region as T1
17         WHERE T1.UID = F.UID AND T1.CommuneName = F.CommuneName
18         AND T1.StartDate < F.StartDate AND T.StartDate <= T1.EndDate ))
19 AND NOT EXISTS(
20     SELECT *
21     FROM Temp_User_Region T2
22     WHERE T2.UID = F.UID AND T2.CommuneName = F.CommuneName
23     AND (
24         (T2.StartDate < F.StartDate AND F.StartDate <= T2.EndDate)
25         OR
26         (T2.StartDate >= L.EndDate AND L.EndDate < T2.EndDate)
27     ))
28
29 SELECT RegionName, COUNT(UID)
30 FROM Temp_User_Region
31 WHERE StartDate <= '01/01/2000' AND '01/01/2001' <= EndDate
32 GROUP BY RegionName

```

5) List the different regions with their total area

```

1 SELECT RegionName, SUM(ST_AREA(Geom))
2 FROM Commune
3 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.

```
1 SELECT C.CommuneName, C.RegionName, ST_Distance(ST_Centroid(C.Geom), ST_Centroid(Cap.Geom)) AS
   DistToCapital
2 FROM Commune C, Region R, Commune Cap
3 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.

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

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

```
1 SELECT CT.CID, (stats).max highest, (stats).min lowest
2 FROM (
3   SELECT CT.CID, ST_SummaryStats(ST_Clip(C.altitude, 1, CT.Itinerary, TRUE)) AS stats
4   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

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

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

```
1 -- If a user can only be subscribe to one commune at a time
2 CREATE TRIGGER PK_User_Subs ON Subscription FOR INSERT, UPDATE AS
3 IF EXISTS(
4   SELECT *
5   FROM Subscription S1
6   WHERE 1 < (
7     SELECT COUNT(S2.UID)
8     FROM Subscription S2
9     WHERE S1.UID = S2.UID
10    AND S1.StartDate < S2.EndDate AND S1.EndDate > S2.StartDate)
11 BEGIN
12   RAISERROR('A user can only have one subscription at a time',1,2)
13   ROLLBACK TRANSACTION
14 END
15
16 -- If a user can be subscribed to several communes at a time, but only once at each
17 CREATE TRIGGER PK_User_Subs ON Subscription FOR INSERT, UPDATE AS
18 IF EXISTS(
19   SELECT *
20   FROM Subscription S1
21   WHERE 1 < (
22     SELECT COUNT(S2.UID)
23     FROM Subscription S2
24     WHERE S1.UID = S2.UID AND S1.SID = S2.SID
```

```

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

```

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

```

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

```

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

```

1 CREATE TRIGGER CountryUnionComm ON Commune FOR INSERT, UPDATE AS
2 IF EXISTS(
3     SELECT ST_DIFFERENCE(Com.Geom, C.Geom)
4     FROM Country C
5           JOIN Region R ON C.CountryName = R.Country
6           JOIN Commune Com ON Com.RegionName = R.RegionName
7
8     UNION
9
10    SELECT ST_DIFFERENCE(C.Geom, ST_UNION(Com.Geom))
11    FROM Country C
12           JOIN Region R ON C.CountryName = R.Country
13           JOIN Commune Com ON Com.RegionName = R.RegionName
14    GROUP BY C.CountryName)
15 BEGIN
16     RAISERROR('A country must be equal to the union of its composing communes',1,2)
17     ROLLBACK TRANSACTION
18 END
19
20 CREATE TRIGGER CountryUnionComm2 ON Country FOR INSERT, UPDATE AS
21 IF EXISTS(
22     SELECT ST_DIFFERENCE(Com.Geom, C.Geom)
23     FROM Country C
24           JOIN Region R ON C.CountryName = R.Country
25           JOIN Commune Com ON Com.RegionName = R.RegionName
26
27     UNION
28
29    SELECT ST_DIFFERENCE(C.Geom, ST_UNION(Com.Geom))
30    FROM Country C
31           JOIN Region R ON C.CountryName = R.Country
32           JOIN Commune Com ON Com.RegionName = R.RegionName
33    GROUP BY C.CountryName)
34 BEGIN
35     RAISERROR('A country must be equal to the union of its composing communes',1,2)
36     ROLLBACK TRANSACTION
37 END

```