SQL Server
Temporal Databases

Consider the above temporal conceptual schema

- Define a relational schema corresponding to the conceptual schema.
- Write the following queries in SQL:
  
  (1) Give the name of managers living currently in Houston
  
  (2) Give the name of employees working currently in the ‘Research’ department having a salary greater or equal than 45000
  
  (3) Give the name of current employees who does not work currently in any department
  
  (4) Give the name of the employee(s) that had the highest salary in 1/1/2002
  
  (5) Provide the salary and affiliation history for all employees
  
  (6) Give the name of supervisors but did not work in any project during the same period
  
  (7) Give the name of supervisors who had work on a project at some time
  
  (8) Give the name of employees and the date they changed their affiliation
  
  (9) Give the name of employees and the periods they worked on any project
  
  (10) Give the history of the maximum salary
  
  (11) Give by department the history of the maximum salary
  
  (12) Give the history of the number of projects of a department
  
  (13) Give the name of employees and the periods they worked on all projects of their department
• Ensure the following integrity constraints.

(1) An employee works in at most one department at any point in time.
(2) At any point in time an employee cannot work more than once in a project.
(3) The lifecycle of affiliation is included in the lifecycle of employee.
(4) The lifecycle of an employee is equal to the union of his/her affiliations.
(5) Employees have a contiguous lifecycle.
(6) The lifecycle of an employee is equal to the union of his/her affiliations, now taking into account that the lifecycle of employees is contiguous.
Relational Schema

- Employee(SSN, FName, MInit, LName, BirthDate, Sex)
- EmployeeLifecycle(SSN, FromDate, ToDate)
  SSN references Employee(SSN)
- EmployeeSalary(SSN, Salary, FromDate, ToDate)
  SSN references Employee(SSN)
- EmployeeAddress(SSN, Street, City, Zip, Country, FromDate, ToDate)
  SSN references Employee(SSN)
- Engineer(SSN, EngineerType, FromDate, ToDate)
  SSN references Employee(SSN)
  In this table the lifecycle of Engineer is kept as well as the attribute EngineerType. There will be redundancy if the lifecycle of Engineer is not continuous.
- EngineerDiplomas(SSN, Diploma)
  SSN references Engineer(SSN)
- AdministrativeLifecycle(SSN, FromDate, ToDate)
  SSN references Employee(SSN)
- AdminDependent(SSN, Name, Relationship, Sex, BirthDate, FromDate, ToDate)
  SSN references AdministrativeLifecycle(SSN)
  It is supposed that an employee does not have two dependents of the same name and the same relationship. An alternative will be to put BirthDate instead of Relationship as part of the key.
- Supervision(SSN, SuperSSN, FromDate, ToDate)
  SSN references Employee(SSN)
  SuperSSN references Employee(SSN)
- Affiliation(SSN, DNumber, FromDate, ToDate)
  SSN references Employee(SSN)
- Department(DNumber, DName, MgrSSN, MgrStartDate, FromDate, ToDate)
  MgrSSN references Employee(SSN)
  It is supposed that the lifecycle of departments is continuous. In this case an additional table for the lifecycle is not necessary.
- DeptLocations(DNumber, Location, FromDate, ToDate)
  DNumber references Department(DNumber)
- DepartmentBudget(DNumber, DBudget, FromDate, ToDate)
  DNumber references Department(DNumber)
- Department NbEmp(DNumber, NbEmp, FromDate, ToDate)
  DNumber references Department(DNumber)
- Project(PNumber, PName, PLocation, PBudget, FromDate, ToDate)
  It is supposed that the lifecycle of Project is continuous.
- Controls(PNumber, DNumber, FromDate, ToDate)
  PNumber references Project(PNumber)
  DNumber references Department(DNumber)
- WorksOn(SSN, PNumber, Hours, FromDate, ToDate)
  PNumber references Project(PNumber)
  SSN references Employee(SSN)
  It is supposed that the temporality of attribute Hours is the same as the lifecycle of the association. In this case two different tables are not necessary. To obtain the lifecycle of the association independently of the attribute hours a temporal projection is needed.
**Example Database**

Partial schema where not all entities and attributes are taken into account.

<table>
<thead>
<tr>
<th>Employee</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>FName</td>
</tr>
<tr>
<td>123456789</td>
<td>John</td>
</tr>
<tr>
<td>333445555</td>
<td>Franklin</td>
</tr>
<tr>
<td>999887777</td>
<td>Alicia</td>
</tr>
<tr>
<td>987654321</td>
<td>Jennifer</td>
</tr>
<tr>
<td>666884444</td>
<td>Ramesh</td>
</tr>
<tr>
<td>453453453</td>
<td>Joyce</td>
</tr>
<tr>
<td>987987987</td>
<td>Ahmad</td>
</tr>
<tr>
<td>888665555</td>
<td>James</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EmployeeLifecycle</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>FromDate</td>
</tr>
<tr>
<td>123456789</td>
<td>01-01-1985</td>
</tr>
<tr>
<td>333445555</td>
<td>01-01-1982</td>
</tr>
<tr>
<td>999887777</td>
<td>01-01-1985</td>
</tr>
<tr>
<td>987654321</td>
<td>01-01-1982</td>
</tr>
<tr>
<td>666884444</td>
<td>01-01-1985</td>
</tr>
<tr>
<td>453453453</td>
<td>01-01-1985</td>
</tr>
<tr>
<td>987987987</td>
<td>01-01-1985</td>
</tr>
<tr>
<td>888665555</td>
<td>01-01-1980</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EmployeeSalary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>Salary</td>
</tr>
<tr>
<td>123456789</td>
<td>30000</td>
</tr>
<tr>
<td>333445555</td>
<td>40000</td>
</tr>
<tr>
<td>333445555</td>
<td>45000</td>
</tr>
<tr>
<td>999887777</td>
<td>25000</td>
</tr>
<tr>
<td>987654321</td>
<td>43000</td>
</tr>
<tr>
<td>666884444</td>
<td>38000</td>
</tr>
<tr>
<td>453453453</td>
<td>25000</td>
</tr>
<tr>
<td>987987987</td>
<td>25000</td>
</tr>
<tr>
<td>888665555</td>
<td>55000</td>
</tr>
<tr>
<td>888665555</td>
<td>58000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EmployeeAddress</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>Street</td>
</tr>
<tr>
<td>123456789</td>
<td>731 Fondren</td>
</tr>
<tr>
<td>333445555</td>
<td>638 Voss</td>
</tr>
<tr>
<td>999887777</td>
<td>3321 Castle</td>
</tr>
<tr>
<td>987654321</td>
<td>291 Berry</td>
</tr>
<tr>
<td>666884444</td>
<td>975 Fire Oak</td>
</tr>
<tr>
<td>453453453</td>
<td>5631 Rice</td>
</tr>
<tr>
<td>987987987</td>
<td>980 Dallas</td>
</tr>
<tr>
<td>888665555</td>
<td>450 Stone</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supervision</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>SuperSSN</td>
</tr>
<tr>
<td>123456789</td>
<td>333445555</td>
</tr>
<tr>
<td>333445555</td>
<td>888665555</td>
</tr>
<tr>
<td>999887777</td>
<td>987654321</td>
</tr>
<tr>
<td>987654321</td>
<td>888665555</td>
</tr>
<tr>
<td>666884444</td>
<td>333445555</td>
</tr>
<tr>
<td>453453453</td>
<td>333445555</td>
</tr>
<tr>
<td>987987987</td>
<td>987654321</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Affiliation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>DNumber</td>
</tr>
<tr>
<td>123456789</td>
<td>1</td>
</tr>
<tr>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>999887777</td>
<td>4</td>
</tr>
<tr>
<td>987654321</td>
<td>4</td>
</tr>
<tr>
<td>666884444</td>
<td>5</td>
</tr>
<tr>
<td>453453453</td>
<td>5</td>
</tr>
<tr>
<td>987987987</td>
<td>4</td>
</tr>
<tr>
<td>888665555</td>
<td>1</td>
</tr>
<tr>
<td>DNumber</td>
<td>DName</td>
</tr>
<tr>
<td>---------</td>
<td>------------------</td>
</tr>
<tr>
<td>1</td>
<td>Headquarters</td>
</tr>
<tr>
<td>4</td>
<td>Administration</td>
</tr>
<tr>
<td>5</td>
<td>Research</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DNumber</th>
<th>NbEmp</th>
<th>FromDate</th>
<th>ToDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>4</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DNumber</th>
<th>DLocation</th>
<th>FromDate</th>
<th>ToDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Houston</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>4</td>
<td>Stafford</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>5</td>
<td>Bellaire</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>5</td>
<td>Sugarland</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>5</td>
<td>Houston</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PNumber</th>
<th>PName</th>
<th>PLocation</th>
<th>FromDate</th>
<th>ToDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ProductX</td>
<td>Bellaire</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>2</td>
<td>ProductY</td>
<td>Sugarland</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>3</td>
<td>ProductZ</td>
<td>Houston</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>10</td>
<td>Computerization</td>
<td>Stafford</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>20</td>
<td>Reorganization</td>
<td>Houston</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>30</td>
<td>Newbenefits</td>
<td>Stafford</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PNumber</th>
<th>DNumber</th>
<th>FromDate</th>
<th>ToDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>30</td>
<td>4</td>
<td>01-01-1980</td>
<td>01-01-2079</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>PNumber</th>
<th>Hours</th>
<th>FromDate</th>
<th>ToDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>1</td>
<td>32.5</td>
<td>01-01-1985</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>123456789</td>
<td>2</td>
<td>7.5</td>
<td>01-01-1985</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>333445555</td>
<td>1</td>
<td>10</td>
<td>01-01-1982</td>
<td>01-01-2000</td>
</tr>
<tr>
<td>333445555</td>
<td>2</td>
<td>10</td>
<td>01-01-1982</td>
<td>01-01-2002</td>
</tr>
<tr>
<td>333445555</td>
<td>3</td>
<td>20</td>
<td>01-01-2005</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>453453453</td>
<td>1</td>
<td>20</td>
<td>01-01-1985</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>453453453</td>
<td>2</td>
<td>20</td>
<td>01-01-1985</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>666684444</td>
<td>3</td>
<td>40</td>
<td>01-01-1985</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>888655555</td>
<td>20</td>
<td>30.0</td>
<td>01-01-1983</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>987654321</td>
<td>10</td>
<td>5.0</td>
<td>01-01-1982</td>
<td>01-01-2000</td>
</tr>
<tr>
<td>987654321</td>
<td>20</td>
<td>15.0</td>
<td>01-01-1982</td>
<td>01-01-2001</td>
</tr>
<tr>
<td>987654321</td>
<td>30</td>
<td>20.0</td>
<td>01-01-1982</td>
<td>01-01-2002</td>
</tr>
<tr>
<td>987987987</td>
<td>10</td>
<td>35.0</td>
<td>01-01-1985</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>987987987</td>
<td>30</td>
<td>5.0</td>
<td>01-01-1985</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>999887777</td>
<td>10</td>
<td>10.0</td>
<td>01-01-1985</td>
<td>01-01-2079</td>
</tr>
<tr>
<td>999887777</td>
<td>30</td>
<td>30.0</td>
<td>01-01-1985</td>
<td>01-01-2079</td>
</tr>
</tbody>
</table>
Queries

(1) Give the name of the managers living currently in Houston

```sql
select E.FName, E.LName
from Employee E, EmployeeAddress A, Department D
where E.SSN = A.SSN and E.SSN = D.MgrSSN
and A.City = 'Houston'
and A.FromDate <= getdate() and getdate() < A.ToDate
and D.FromDate <= getdate() and getdate() < D.ToDate
```

(2) Give the name of employees working currently in the ‘Research’ department and having a salary greater or equal than 45000

```sql
select E.FName, E.LName
from Employee E, EmployeeSalary S, Affiliation A, Department D
where E.SSN = S.SSN and E.SSN = A.SSN and A.DNumber = D.DNumber
and D.DName = 'Research' and S.Salary >= 45000
and S.FromDate <= getdate() and getdate() < S.ToDate
and A.FromDate <= getdate() and getdate() < A.ToDate
```

(3) Give the name of current employees who do not work currently in any department

```sql
select distinct E.FName, E.LName
from Employee E, EmployeeLifecycle L
where E.SSN = L.SSN
and L.FromDate <= getdate() and getdate() < L.ToDate
and not exists (
    select * from Affiliation A
    where E.SSN = A.SSN
    and A.FromDate <= getdate() and getdate() < A.ToDate )
```

(4) Give the name of the employee(s) that had the highest salary on 1/1/2002

```sql
select E.FName, E.LName
from Employee E, EmployeeSalary S
where E.SSN = S.SSN
and salary = ( select max(salary) from EmployeeSalary
    where FromDate <= '2002-01-01' and '2002-01-01' < ToDate )
and S.FromDate <= '2002-01-01' and '2002-01-01' < S.ToDate
```

(5) Provide the salary and affiliation history for all employees

```sql
create function minDate
    (@one smalldatetime, @two smalldatetime)
returns smalldatetime as
begin
    return CASE WHEN @one < @two then @one else @two end
end

go
create function maxDate
    (@one smalldatetime, @two smalldatetime)
returns smalldatetime as
begin
    return CASE WHEN @one > @two then @one else @two end
end

go
select E.FName, E.LName, D.DName, S.Salary,
    'Start Date' = dbo.maxDate(S.FromDate, A.FromDate),
    'End Date' = dbo.minDate(S.ToDate, A.ToDate)
from Employee E, EmployeeSalary S, Affiliation A, Department D
```
(6) Give the name of employees and the period of time in which they were supervisors but did not work in any project during the same period

--Case 1
select S.SuperSSN, S.FromDate, W1.FromDate as ToDate
from Supervision S, WorksOn W1
where S.SuperSSN = W1.SSN
and S.FromDate < W1.FromDate and W1.FromDate < S.ToDate
and not exists ( select * from WorksOn W2 where S.SuperSSN = W2.SSN
and S.FromDate < W2.ToDate and W2.FromDate < W1.FromDate )
union
--Case 2
select S.SuperSSN, W1.ToDate as FromDate, S.ToDate
from Supervision S, WorksOn W1
where S.SuperSSN = W1.SSN
and S.FromDate < W1.ToDate and W1.ToDate < S.ToDate
and not exists ( select * from WorksOn W2 where S.SuperSSN = W2.SSN
and W1.ToDate < W2.ToDate and W2.FromDate < S.ToDate )
union
--Case 3
select S.SuperSSN, W1.ToDate as FromDate, W2.FromDate as ToDate
from Supervision S, WorksOn W1, WorksOn W2
where S.SuperSSN = W1.SSN and S.SuperSSN = W2.SSN and W1.ToDate < W2.FromDate
and S.FromDate < W1.ToDate and W2.FromDate < S.ToDate
and not exists ( select * from WorksOn W3 where S.SuperSSN = W3.SSN
and W1.ToDate < W3.ToDate and W3.FromDate < W2.FromDate )
union
--Case 4
select SuperSSN, FromDate, ToDate from Supervision S
where not exists ( select * from WorksOn W where S.SuperSSN=W.SSN
and S.FromDate < W.ToDate and W.FromDate < S.ToDate )

(7) Give the name of supervisors who had work on a project at some time
select distinct E.FName, E.LName
from Employee E, Supervision S, WorksOn W
where E.SSN = S.SuperSSN and E.SSN = W.SSN

(8) Give the name of employees and the date they changed their affiliation
select distinct E.FName, E.LName, A1.ToDate
from Employee E, Affiliation A1, Affiliation A2
where E.SSN = A1.SSN and E.SSN = A2.SSN
and A1.ToDate = A2.FromDate and A1.DNumber <> A2.DNumber

(9) Give the name of employees and the periods they worked on any project
select distinct E.SSN, E.FName, E.LName, F.FromDate, L.ToDate
from Employee E, WorksOn F, WorksOn L
where E.SSN = F.SSN and E.SSN = L.SSN and F.FromDate < L.ToDate
and not exists ( select * from WorksOn M
where M.SSN = F.SSN
and F.FromDate < M.FromDate and M.FromDate <= L.ToDate
and not exists ( select * from WorksOn T1
where T1.SSN = F.SSN
and T1.FromDate < M.FromDate and M.FromDate <= T1.ToDate ) )
and not exists ( select * from WorksOn T2
and T2.ToDate < M.ToDate and M.ToDate <= T2.ToDate )

where T2.SSN = F.SSN
and ( ( T2.FromDate < F.FromDate and F.FromDate <= T2.ToDate )
or ( T2.FromDate <= L.ToDate and L.ToDate < T2.ToDate ) )

(10) Give the history of the maximum salary

   -- First step: Construct intervals during which no salary change occurred
   WITH Instants(Instant) AS ( select distinct E.FromDate from EmployeeSalary E
union select distinct E.ToDate from EmployeeSalary E ),
Intervals(FromDate,ToDate) AS ( select distinct I1.Instant, I2.Instant
from Instants I1, Instants I2
where I1.Instant < I2.Instant
and not exists ( select * from Instants I3
where I1.Instant < I3.Instant
and I3.Instant < I2.Instant ) ),
   -- Second step: Compute the maximum salary for these intervals
   TempMax(SalaryMax, FromDate, ToDate) AS ( select max(E.Salary), I.FromDate, I.ToDate
from EmployeeSalary E, Intervals I
where E.FromDate <= I.FromDate and I.ToDate <= E.ToDate
group by I.FromDate, I.ToDate )
   -- Third step: Coalescing the above table
   select distinct F.SalaryMax, F.FromDate, L.ToDate
from TempMax F, TempMax L
where F.FromDate < L.ToDate and F.SalaryMax = L.SalaryMax
and not exists ( select *
from TempMax M
where M.SalaryMax = F.SalaryMax
and F.ToDate < M.FromDate and M.FromDate <= L.FromDate
and not exists ( select *
from TempMax T1
where T1.SalaryMax = F.SalaryMax
and T1.FromDate < M.FromDate and M.FromDate <= T1.ToDate )
and not exists ( select *
from TempMax T2
where T2.SalaryMax = F.SalaryMax
and ( ( T2.FromDate < F.FromDate and F.FromDate <= T2.ToDate )
or ( T2.FromDate <= L.ToDate and L.ToDate < T2.ToDate ) )
order by F.FromDate

(11) Give by department the history of the maximum salary

   -- First step: Construct by department the intervals during
   -- which the maximum salary must be calculated.
   WITH Aff_Sal (DNumber, Salary, FromDate, ToDate) AS ( select distinct A.DNumber, S.Salary,
dbo.maxDate(S.FromDate,A.FromDate),
dbo.minDate(S.ToDate,A.ToDate)
from Affiliation A, EmployeeSalary S
where A.SSN = S.SSN
and dbo.maxDate(S.FromDate,A.FromDate) <
dbo.minDate(S.ToDate,A.ToDate) ),
   SalChanges(DNumber, Instant) AS ( select distinct DNumber, FromDate from Aff_Sal
union select distinct DNumber, ToDate from Aff_Sal ),
   SalIntervals(DNumber, FromDate, ToDate) AS ( select distinct P1.DNumber, P1.Instant, P2.Instant
order by P1.Instant
from SalChanges P1, SalChanges P2
where P1.DNumber = P2.DNumber and P1.Instant < P2.Instant
and not exists ( select * from SalChanges P3
    where P1.DNumber = P3.DNumber and P1.Instant < P3.Instant
    and P3.Instant < P2.Instant ) ),
-- Second step: Compute the maximum salary for the
-- above periods.
TempMaxDep(DNumber, MaxSalary, FromDate, ToDate) AS ( select P.DNumber, max(Salary), P.FromDate, P.ToDate from Aff_Sal A, SalIntervals P
where A.DNumber = P.DNumber
and A.FromDate <= P.FromDate and P.ToDate <= A.ToDate
group by P.DNumber, P.FromDate, P.ToDate )
-- Third step: Coalescing the above table
select distinct F.DNumber, F.MaxSalary, F.FromDate, L.ToDate
from TempMaxDep F, TempMaxDep L
where F.DNumber = L.DNumber and F.MaxSalary = L.MaxSalary
and F.FromDate < L.ToDate
and not exists ( select *
    from TempMaxDep M
    where F.DNumber = M.DNumber and F.MaxSalary = M.MaxSalary
    and F.ToDate < M.FromDate and M.FromDate <= L.FromDate
    and not exists ( select *
        from TempMaxDep T1
        where F.DNumber = T1.DNumber and F.MaxSalary = T1.MaxSalary
        and T1.FromDate < M.FromDate and M.FromDate <= L.ToDate ) )
and not exists ( select *
    from TempMaxDep T2
    where F.DNumber = T2.DNumber and F.MaxSalary = T2.MaxSalary
    and ( ( T2.FromDate < F.FromDate and F.FromDate <= T2.ToDate )
        or ( T2.FromDate < L.ToDate and L.ToDate < T2.ToDate ) )
) order by F.DNumber, F.FromDate

(12) Give the history of the number of projects of a department

-- First step: Construct intervals during which the number of
-- projects of a department does not change
WITH Instants(DNumber, Instant) AS ( select distinct DNumber, FromDate from Controls union select distinct DNumber, ToDate from Controls ),
Intervals(DNumber, FromDate, ToDate) AS ( select distinct I1.DNumber, I1.Instant, I2.Instant from Instants I1, Instants I2
where I1.DNumber = I2.DNumber
and I1.Instant < I2.Instant
and not exists ( select * from Instants I3
    where I1.DNumber = I3.DNumber
    and I1.Instant < I3.Instant
    and I3.Instant < I2.Instant ) ),
-- Second step: Compute the number of projects for these intervals
TempCountDep(DNumber, NbProjects, FromDate, ToDate) AS ( select I.DNumber, count(C.PNumber), I.FromDate, I.ToDate from Controls C, Intervals I
where C.DNumber = I.DNumber
and ( C.FromDate <= I.FromDate and I.ToDate <= C.ToDate )
group by I.DNumber, I.FromDate, I.ToDate )
-- Third step: Coalescing the above table
select distinct F.DNumber, F.NbProjects, F.FromDate, L.ToDate
from TempCountDep F, TempCountDep L
where F.DNumber = L.DNumber and F.FromDate < L.ToDate
and F.NbProjects = L.NbProjects
and not exists ( select *
from TempCountDep M
where M.DNumber = F.DNumber and M.NbProjects = F.NbProjects
and F.ToDate < M.FromDate and M.FromDate <= L.FromDate
and not exists ( select *
from TempCountDep T1
where T1.DNumber = F.DNumber and T1.NbProjects = F.NbProjects
and T1.FromDate < M.FromDate and M.FromDate <= T1.ToDate ) )
and not exists ( select *
from TempCountDep T2
where T2.DNumber = F.DNumber and T2.NbProjects = F.NbProjects
and ( ( T2.FromDate < F.FromDate and F.FromDate <= T2.ToDate )
or ( T2.FromDate <= L.ToDate and L.ToDate < T2.ToDate ) )
) order by F.DNumber, F.FromDate

(13) Give the name of employees and the periods they worked on all projects of their department

-- First step: Construct intervals during which the number of projects
-- of an employee does not change

WITH Aff_Cont(SSN, DNumber, PNumber, FromDate, ToDate) AS (
    select distinct A.SSN, A.DNumber, C.PNumber,
    dbo.maxDate(A.FromDate,C.FromDate),
    dbo.minDate(A.ToDate,C.ToDate)
    from Affiliation A, Controls C
    where A.DNumber = C.DNumber
    and dbo.maxDate(A.FromDate,C.FromDate) <
    dbo.minDate(A.ToDate,C.ToDate) ),

Aff_Cont_WO(SSN, DNumber, PNumber, FromDate, ToDate) AS (
    select distinct A.SSN, A.DNumber, W.PNumber,
    dbo.maxDate(A.FromDate,W.FromDate),
    dbo.minDate(A.ToDate,W.ToDate)
    from Aff_Cont A, WorksOn W
    where A.PNumber = W.PNumber and A.SSN = W.SSN
    and dbo.maxDate(A.FromDate,W.FromDate) <
    dbo.minDate(A.ToDate,W.ToDate) ),

ProjChanges(SSN, DNumber, Instant) AS (
    select distinct SSN, DNumber, FromDate from Aff_Cont
    union select distinct SSN, DNumber, ToDate from Aff_Cont
    union select distinct SSN, DNumber, FromDate from Aff_Cont_WO
    union select distinct SSN, DNumber, ToDate from Aff_Cont_WO
    union select SSN, DNumber, FromDate from Affiliation
    union select SSN, DNumber, ToDate from Affiliation ),

ProjIntervals(SSN, DNumber, FromDate, ToDate) AS (
    select distinct P1.SSN, P1.DNumber, P1.Instant, P2.Instant
    from ProjChanges P1, ProjChanges P2
    where P1.SSN = P2.SSN
    and P1.DNumber = P2.DNumber and P1.Instant < P2.Instant
    and not exists ( select * from ProjChanges P3
    where P1.SSN = P3.SSN and P1.DNumber = P3.DNumber
    and P1.Instant < P3.Instant and P3.Instant < P2.Instant ) ),

-- Second step: Compute the number of projects for these intervals

TempUnivQuant(SSN, FromDate, ToDate) AS ( 
    select distinct P.SSN, P.FromDate, P.ToDate
    from ProjIntervals P
    where not exists ( select * from Controls C 
    and P.FromDate <= C.FromDate and C.ToDate <= P.ToDate ) 
    )
where P.DNumber = C.DNumber
and C.FromDate <= P.FromDate and P.ToDate <= C.ToDate
and not exists ( select * from WorksOn W
where C.PNumber = W.PNumber and P.SSN = W.SSN
and W.FromDate <= P.FromDate and P.ToDate <= W.ToDate ) )

-- Third step: Coalescing the above table
select distinct F.SSN, F.FromDate, L.ToDate
from TempUnivQuant F, TempUnivQuant L
where F.SSN = L.SSN and F.FromDate < L.ToDate
and not exists ( select *
from TempUnivQuant M
where M.SSN = F.SSN
and F.ToDate < M.FromDate and M.FromDate <= L.FromDate
and not exists ( select *
from TempUnivQuant T1
where T1.SSN = F.SSN
and T1.FromDate < M.FromDate and M.FromDate <= T1.ToDate ) )
and not exists ( select *
from TempUnivQuant T2
where T2.SSN = F.SSN
and ( ( T2.FromDate < F.FromDate and F.FromDate <= T2.ToDate )
or ( T2.FromDate <= L.ToDate and L.ToDate < T2.ToDate ) )
order by F.SSN, F.FromDate

Constraints

(1) An employee works in at most one department at any point in time.
In other terms SSN is a sequenced primary key for Affiliation.

create trigger Seq_PK_Affiliation on Affiliation
after insert, update as
if exists ( select * from Inserted A1
where 1 < ( select count(*) from Affiliation A2
where A1.SSN = A2.SSN
and A1.FromDate < A2.ToDate and A2.FromDate < A1.ToDate ) )
begin
raiserror 13000
'An employee works in at most one department at any point in time'
rollback transaction
end

(2) At any point in time an employee cannot work more than once in a project.
In other terms (SSN,PNumber) is a sequenced primary key for WorksOn

create trigger Seq_PK_WorksOn on WorksOn
after insert, update as
if exists ( select * from Inserted W1
where 1 < ( select count(*) from WorksOn W2
where W1.SSN = W2.SSN and W1.PNumber = W2.PNumber
and W1.FromDate < W2.ToDate and W2.FromDate < W1.ToDate ) )
begin
raiserror 13000
'At any point in time an employee cannot work more than once in a project'
rollback transaction
end

(3) The lifecycle of affiliation is included in the lifecycle of employee.
In the following triggers it is assumed that the table EmployeeLifecycle is coalesced.
Therefore, every line in Affiliation must be covered by one line in EmployeeLifecycle.
create trigger Seq_FK_Affiliation_EmployeeLifecycle_1 on Affiliation
after insert, update as
if exists ( select * from Inserted A
    where not exists ( select * from EmployeeLifecycle E
        where A.SSN = E.SSN
        and E.FromDate <= A.FromDate and A.ToDate <= E.ToDate ) )
begin
    raiserror 13000
    'The lifecycle of affiliation must be included in the lifecycle of employee'
    rollback transaction
end

create trigger Seq_FK_Affiliation_EmployeeLifecycle_2 on EmployeeLifecycle
after update, delete as
if exists ( select * from Affiliation A
    where A.SSN IN ( select SSN from Deleted)
    and not exists ( select * from EmployeeLifecycle E
        where A.SSN = E.SSN
        and E.FromDate <= A.FromDate and A.ToDate <= E.ToDate ) )
begin
    raiserror 13000
    'The lifecycle of affiliation must be included in the lifecycle of employee'
    rollback transaction
end

(4) The lifecycle of an employee is equal to the union of his/her affiliations. It is
supposed that the previous trigger is activated, therefore it is sufficient to monitor
that an employee must be affiliated to a department throughout his/her lifecycle.

create trigger Seq_FK_EmployeeLifecycle_Affiliation_1 on Affiliation
after update, delete as
if exists ( select * from EmployeeLifecycle E
    where E.SSN in ( select SSN from Deleted )
    and not exists ( select * from Affiliation A
        where E.SSN = A.SSN
        and A.FromDate <= E.FromDate and E.FromDate < A.ToDate )
or not exists ( select * from Affiliation A
        where E.SSN = A.SSN
        and A.FromDate < E.ToDate and E.ToDate <= A.ToDate )
or exists ( select * from Affiliation A
        where E.SSN = A.SSN
        and E.FromDate < A.ToDate and A.ToDate <= E.ToDate
        and not exists ( select * from Affiliation A2
            where A2.SSN = A.SSN
            and A2.FromDate <= A.ToDate and A.ToDate < A2.ToDate ) )
begin
    raiserror 13000
    'An employee must be affiliated to a department throughout his/her lifecycle'
    rollback transaction
end

create trigger Seq_FK_EmployeeLifecycle_Affiliation_2 on EmployeeLifecycle
after insert, update as
if exists ( select * from Inserted E
    where not exists ( select * from Affiliation A
        where E.SSN = A.SSN
        and A.FromDate <= E.FromDate and E.FromDate < A.ToDate )
or not exists ( select * from Affiliation A
        where E.SSN = A.SSN
        and A.FromDate < E.ToDate and E.ToDate <= A.ToDate )
or not exists ( select * from Affiliation A
        where E.SSN = A.SSN
        and A.FromDate <= E.FromDate and E.FromDate < A.ToDate )
begin
    raiserror 13000
    'An employee must be affiliated to a department throughout his/her lifecycle'
    rollback transaction
end
and A.FromDate < E.ToDate and E.ToDate <= A.ToDate )
or exists ( select * from Affiliation A
    where E.SSN = A.SSN
    and E.FromDate < A.ToDate and A.ToDate < E.ToDate
    and not exists ( select * from Affiliation A2
        where A2.SSN = A.SSN
        and A2.FromDate <= A.ToDate and A.ToDate < A2.ToDate ) )
    begin
    raiserror 13000
    'An employee must be affiliated to a department throughout his/her lifecycle'
    rollback transaction
    end

(5) Employees have a contiguous lifecycle.

   alter table EmployeeLifecycle
   drop constraint PK_EmployeeLifecycle
   alter table EmployeeLifecycle
   add constraint PK_EmployeeLifecycle primary key (SSN)

(6) The lifecycle of an employee is equal to the union of his/her affiliations, now taking
    into account that the lifecycle of employees is contiguous.
    It is necessary to ensure that (1) the affiliations of an employee define a contiguous
    history, and (2) an employee must be affiliated to a department throughout his/her
    lifecycle.
    The following trigger ensures that the affiliations of an employee define a contiguous
    history.

   create trigger Contiguous_Hist_Affiliation on Affiliation
   after insert, update, delete as
   if exists ( select * from Affiliation A1, Affiliation A2
        where A1.SSN = A2.SSN and A1.ToDate < A2.FromDate
        and not exists ( select * from Affiliation A3
            where A1.SSN = A3.SSN
            and ( ( A3.FromDate <= A1.ToDate and A1.ToDate < A3.ToDate )
            or ( A3.FromDate < A2.FromDate and A2.FromDate <= A3.ToDate ) ) ) )
   begin
   raiserror 13000
   'The affiliations of an employee define a contiguous history'
   rollback transaction
   end

The following two triggers replaces those of question (4).

   alter trigger Seq_FK_EmployeeLifecycle_Affiliation_1 on Affiliation
   after update, delete as
   if exists ( select * from EmployeeLifecycle E
        where E.SSN in ( select SSN from Deleted )
        and not exists ( select * from Affiliation A
            where E.SSN = A.SSN
            and A.FromDate <= E.FromDate and E.FromDate < A.ToDate )
or not exists ( select * from Affiliation A
        where E.SSN = A.SSN
        and A.FromDate < E.ToDate and E.ToDate <= A.ToDate ) )
   begin
   raiserror 13000
   'An employee must be affiliated to a department throughout his/her lifecycle'
   rollback transaction
   end
alter trigger Seq_FK_EmployeeLifecycle_Affiliation_2 on EmployeeLifecycle
after insert, update as
if exists ( select * from Inserted E
where not exists ( select * from Affiliation A
where E.SSN = A.SSN
and A.FromDate <= E.FromDate and E.FromDate < A.ToDate )
or not exists ( select * from Affiliation A
where E.SSN = A.SSN
and A.FromDate < E.ToDate and E.ToDate <= A.ToDate ) )
begin
raiserror 13000
'An employee must be affiliated to a department throughout his/her lifecycle'
rollback transaction
end