Exercise Series 4

Tasks 4.4 - 4.10 will be solved with a temporally enhanced PostgreSQL server (designed and implemented by Anton Dignös). You can download and install it from http://www.ifi.uzh.ch/dbtg/research/align.html. Alternatively, from inside the uzh network you can call psql from the command line: psql -h peter.ifi.uzh.ch -d align -U tsdm -p 5432. Yet another possibility is to use pgadmin3.

Task 4.1: For a snapshot relation it is common to to define a primary key. Generalize a primary key to a valid time relation.

As an example consider relation schema $R(\text{Name}, \text{Adr}, \text{VT})$ with AccountNr as a primary key. Describe the different possibilities and illustrate them in terms of a real world example.

Task 4.2: Consider the following two relations:

<table>
<thead>
<tr>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>[6,9]</td>
</tr>
<tr>
<td>8</td>
<td>[3,10]</td>
</tr>
<tr>
<td>8</td>
<td>[8,15]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>[7,10]</td>
</tr>
<tr>
<td>9</td>
<td>[10,20]</td>
</tr>
</tbody>
</table>

and the SQL statement

\[
\text{select * from R where not exists ( select * from S where X > A )}
\]

Use snapshot reducibility and change preservation to define a temporal semantics for this statement and show the result of the statement.

Task 4.3: Show an algebra expression for which it is not possible to define the semantics in terms of snapshot reducibility.

Task 4.4: Consider relations emp1 and emp2 as shown below. Compute the temporal difference $emp1 -^T emp2$. 
Task 4.5: Consider relations emp1 as shown below. Compute the count of employees at each point in time. Determine the query plan of PostgreSQL (use EXPLAIN before the statement).

Task 4.6: Determine the person with the largest budget at each point in time.

Task 4.7: Use PostgreSQL with interval adjustment to formulate the six example queries (cf. lecture slides) over the checkout instance.

Task 4.8: Assume table res with hotel room reservation. For each point in time determine the average duration of reservations. Formulate a query that retrieves the adjusted intervals.

Task 4.9: Compute a temporal full outer natural join between emp1 and proj. Include in the result the old and new interval timestamps. First, determine the correct result on paper. Next, implement your solution with PostgreSQL with interval adjustment.

Task 4.10: Compute a temporal join and a temporal difference between proj and emp2. Scale the budget to the reported result periods.

Task 4.11: Consider the following relations:

<table>
<thead>
<tr>
<th></th>
<th>R</th>
<th>T</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td></td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>r1</td>
<td>a</td>
<td>[5,8]</td>
<td>a</td>
<td>[6,7]</td>
</tr>
<tr>
<td>r2</td>
<td>b</td>
<td>[7,9]</td>
<td>b</td>
<td>[8,14]</td>
</tr>
<tr>
<td>r3</td>
<td>b</td>
<td>[10,11]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Determine the lineage sets for \( \vartheta_{\text{COUNT}(A)}^T(R) \), \( \pi_5^T(R) \), \( R -^T S \), and \( R \bowtie_{A=B}^T S \). Assume set semantics (each snapshot id duplicate free).
Database instance:

create table emp1(emp varchar(3), dept varchar(2), ts integer, te integer);
insert into emp1 values ('Sam', 'DB', 1, 6);
insert into emp1 values ('Ann', 'DB', 3, 8);
insert into emp1 values ('Ann', 'AI', 9, 15);
insert into emp1 values ('Joe', 'DB', 14, 19);

create table emp2(emp varchar(3), dept varchar(2), ts integer, te integer);
insert into emp2 values ('Sam', 'DB', 4, 11);
insert into emp2 values ('Joe', 'DB', 12, 21);

create table proj(
    proj varchar(2), dept varchar(2), budg integer, ts integer, te integer);
insert into proj values ('P1', 'DB', 100, 2, 7);
insert into proj values ('P2', 'DB', 580, 12, 18);

create table checkout(
    custid varchar(7), tapenum varchar(5), ts date, te date);
insert into checkout values ('c101', 't1234', '2009/8/3', '2009/8/6');
insert into checkout values ('c102', 't1245', '2009/8/5', '2009/8/8');
insert into checkout values ('c102', 't1234', '2009/8/9', '2009/8/13');
insert into checkout values ('c102', 't1245', '2009/8/19', '2009/8/21');
insert into checkout values ('c102', 't1245', '2009/8/21', '2009/8/23');

create table res(name varchar(5), ts date, te date);
insert into res values ('Ann', '2012/1/1', '2012/8/1');
insert into res values ('Joe', '2012/2/1', '2012/7/1');
insert into res values ('Ann', '2012/8/1', '2012/12/1');
insert into res values ('Per', '2012/4/1', '2012/9/1');

create table budg(name varchar(5), amnt integer, ts date, te date);
insert into budg values ('Joe', 5, '2012/2/1', '2012/9/1');
insert into budg values ('Ann', 7, '2012/5/1', '2012/9/1');
insert into budg values ('Per', 3, '2012/4/1', '2012/10/1');