Second MidTerm of *Decision Support Systems / Decision Support Databases*

It is forbidden to consult any material during the test. Duration of written exam is 1.5h.

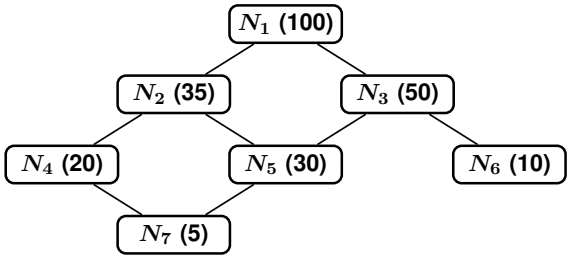
1. **(4 points)** Consider the FoodMart datawarehouse.

Write an **analytic** SQL query to solve the following problem: the list of store_city and product_id's such that the product_id is the most sold one (wrt store_sales) in at least one store_id of the city.

Solution:

```
WITH tmp AS (
  SELECT store_city, S.store_id, product_id,
    RANK() OVER (PARTITION BY store_city, S.store_id
                 ORDER BY SUM(store_sales) DESC) AS rk
  FROM sales_fact S, store T
  WHERE S.store_id = T.store_id
  GROUP BY store_city, S.store_id, product_id)
SELECT DISTINCT store_city, product_id
FROM tmp
WHERE rk=1
ORDER BY store_city, product_id
```

2. **(4 points)** Let us consider the following lattice of possible candidate views to materialize. The numbers associated with the nodes represent the view size, measured in terms of the number of tuples in the view. Select 2 views to materialize, different from N_1, with the greedy algorithm HRU.



Solution:

Benefits at the first step are:

N_2 $(100-35)\cdot 4 = \underline{260}$

N_3 $(100-50)\cdot 4 = 200$

N_4 $(100-20)\cdot 2 = 160$

N_5 $(100-30)\cdot 2 = 140$

N_6 $(100-10)\cdot 4 = 90$

N_7 $(100-5)\cdot 1 = 95$

Benefits at the second step are:

N_3 $(100-50)\cdot 2 + 0\cdot 2 = \underline{100}$

N_4 $(35-20)\cdot 2 = 30$

N_5 $(35-30)\cdot 2 = 10$

N_6 $(100-10)\cdot 4 = 90$

N_7 $(35-5)\cdot 1 = 30$

Thus, HRU selects $\{N1, N2, N3\}$.

3. **(4 points)** Consider the data mart about bus tickets, without null values, and the query:

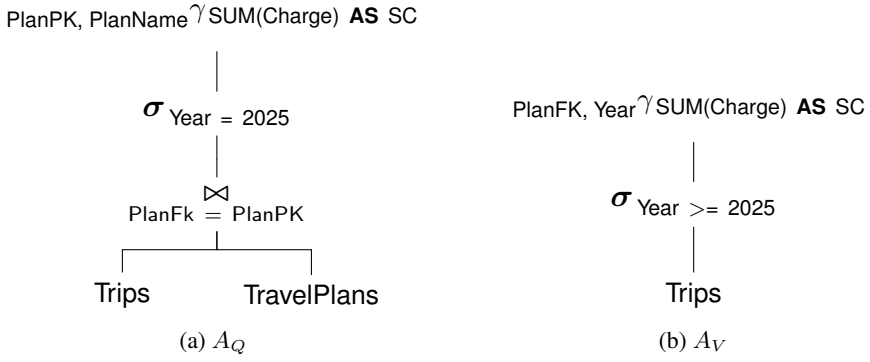
Travellers(TPk, Name, Address)
TravelPlans(PlanPk, PlanName)
Trips(TFk, PlanFk, Day, Month, Year, Duration, Charge)

```
Q:  SELECT      PlanPk, PlanName, SUM(Charge) AS SC
    FROM        Trips, TravelPlans
    WHERE       PlanFk = PlanPk AND Year = 2025
    GROUP BY    PlanPk, PlanName;
```

Show how to rewrite the query Q using the view V, if possible.

```
V:  SELECT      PlanFk, Year, SUM(Charge) AS SC
    FROM        Trips
    WHERE       Year >= 2020
    GROUP BY    PlanFk, Year;
```

Solution: Let us use the approach with a *compensation on the view* starting from the logical query plans of query and view.



(a) A_Q (b) A_V

Figure 1: Query and view logical query plans

(join) There is a partial match, which can be compensated by joining the view with `TravelPlans`. The compensation can float since `PlanFk` is in the output of the view.

(selection) There is a partial match, which can be compensated by restricting to `Year=2025`. The compensation can float since `Year` is in the output of the view.

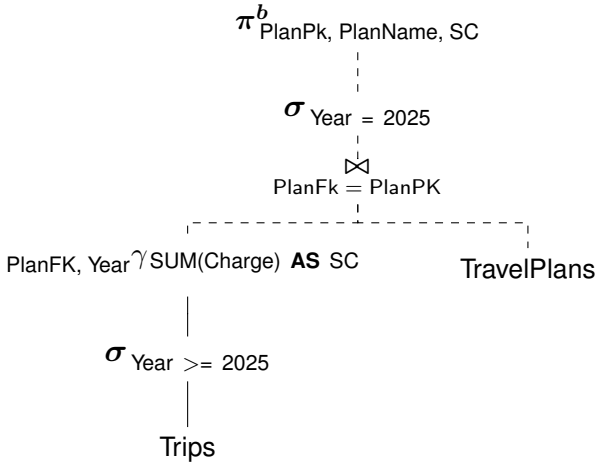
(grouping) Let us check the functional dependencies for $g(Q) = \{$`PlanPk, PlanName`$\}$ and $g(V) = \{$ `PlanFk, Year` $\}$:

$g(Q) \rightarrow q(V)$ is valid, in fact $\{$`PlanPk, PlanName`$\}^+ = \{$ `PlanPk, PlanName, PlanFk, Year, ...` $\}$ since `PlanPk=PlanFk` and `Year=2025`.

$g(V) \rightarrow q(Q)$ is valid, in fact $\{$`PlanFk, Year`$\}^+ = \{$ `PlanFk, Year, PlanPk, PlanName, ...` $\}$ since `PlanPk=PlanFk` and `PlanPk` is a key of `TravelPlans`.

Therefore, the compensation consists only of a projection.

In summary, we have the following compensation:



and the query Q can be rewritten as:

```
SELECT    PlanPk, PlanName, SC
FROM      V, TravelPlans
WHERE     PlanFk = PlanPk AND Year = 2025;
```

4. **(4 points)** Answer the following questions with reference to the FoodMart datawarehouse:

   (a) assuming a BMFCJ index on customer.city write the physical query plan for the following query:

   ```
   V:  SELECT      product_id, SUM(store_sales) AS TotalSales
       FROM        sales_fact AS S, customer AS C
       WHERE       S.customer_id = C.customer_id AND C.city='Los Angeles'
       GROUP BY    product_id;
   ```

   (b1) (for Italian speaking students) discuss the benefits of the optimizations implemented in the SadasDB DBMS over the FoodMart databases;

   (b2) (for non-Italian speaking students) write two SQL queries on FoodMart in order to discuss when column-based storage is better than traditional (heap-based) storage.

   Solution: (a) Let us call the index as `Idx`. We have: