Exam of *Decision Support Systems* / *Decision Support Databases*: Example 1

It is forbidden to consult any material during the test. Duration of written exam is 2h.

1. **(Mandatory)** Let us consider the following database, without null values:

Products

PkP	UnitPrice	...
10	5	...
20	10	...
30	20	...

Sales

FkP	Qty	...
10	50	...
20	10	...
30	20	...
10	30	...
20	100	...
30	10	...
10	30	...

  (a) **(1 point)** Write an SQL query to find the total sales revenue by product.

  (b) **(1 point)** Give a logical query plan for the SQL query, the type and the value of the result. Modify the logical query plan to consider only products with UnitPrice > 5 sold each of them more than 5 times.

  (c) **(1 point)** Modify the SQL query to find also the rank of the product total quantity sold (the highest is first)).

  (d) **(1 point)** Show the instance of an index on the attribute FkP.

  (e) **(1 point)** Show the instance of a *Foreign Column Join Index* on the attribute UnitPrice.
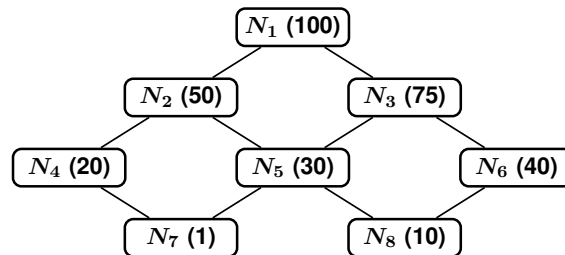
2. **(5 points)** Consider the FoodMart datawarehouse:
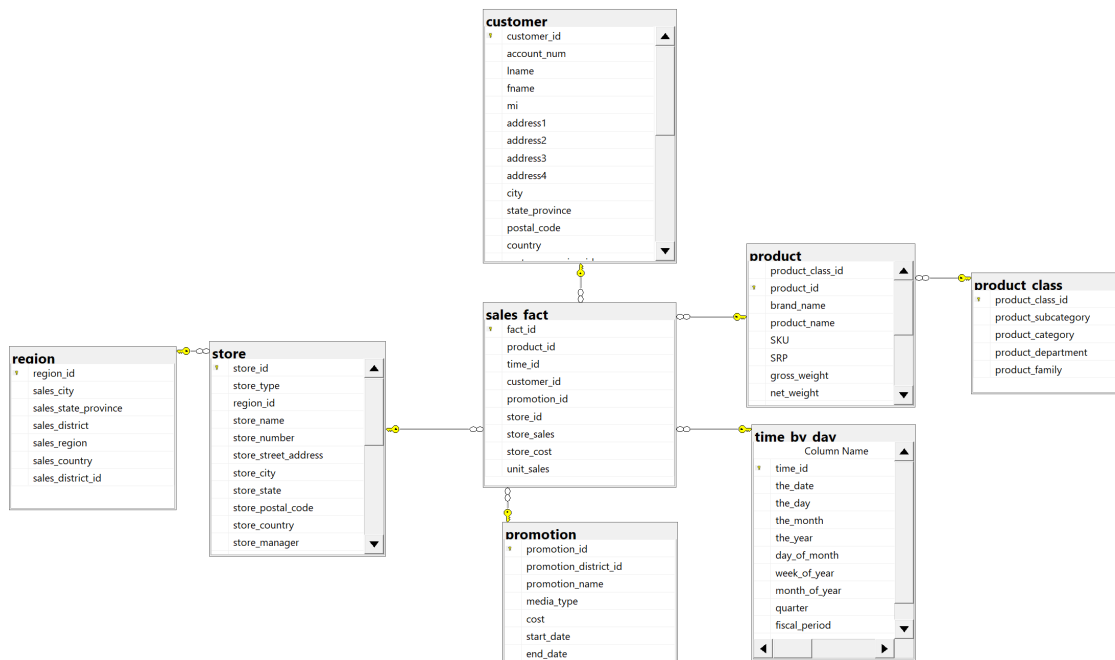
Consider a customer *customer_id* in a day *time_id* when she/he has made a purchase. The customer is *occasional* if she/he has made purchases for at most 10 euros in the previous 45 days. Give an analytic SQL query to output for every *customer_id* and *time_id* that are NOT occasional, the total sales in the previous 45 days.

3. See lecture notes Exercise A.4: Inventory.
   With respect to the above business scenario, answer the following questions:

   (a) **(6 points)** Design a conceptual schema for the data mart to support the business questions. Your schema should at least be able to satisfy the above mentioned analysis requirements. You may motivate other suitable attributes for the dimensions. Specify the fact granularity, and for each measure, if it is additive, semi-additive or non-additive, and for each dimensional attribute the type of updates of the attribute.

   (b) **(3 points)** Give a logical data mart design which is able to deal with updates of dimensional attributes.

   (c) **(1 point)** Write an SQL query to answer one business questions of your choice and write a physical query plan for the query using any index of your choice.

4. **(5 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 3 views to materialize, different from N_1, with the greedy algorithm HRU.

5. **(5 points)** Let us consider the logical schema of a data mart

   Customer(PkCustPhoneNo, CustName, CustCity)
   CallingPlans(PkPlanId, PlanName)
   Calls(PkCustPhoneNo, FkPlanId, Day, Month, Year, Duration, Charge)

   where PkPlanId e PlanName are two different keys, and the following query

Q:	**SELECT**	Year, PlanName, SUM(Charge) **AS** TC
	FROM	Calls, CallingPlans
	WHERE	FkPlanId = PkPlanId **AND** Year >= 2000 **AND** Year <=2005
	GROUP BY	Year, PlanName
	HAVING	SUM(Charge) > 1000;

   Answer **only one** of the following questions:

   (a) Show if and how the GROUP BY can be brought forward on the table Calls.

   (b) Show if and how the query can be rewritten using the materialized view

V1:	**SELECT**	FkPlanId, Month, Year, SUM(Charge) **AS** C
	FROM	Calls
	WHERE	Year >= 2000
	GROUP BY	FkPlanId, Month, Year;

Exam of *Decision Support Databases: Solution Example 1 (Hints)*

1. (a) The SQL query is:

SELECT	PkP, SUM(UnitPrice*Qty) **AS** TR
FROM	Sales, Products
WHERE	FkP = PkP
GROUP BY	PkP;

   (b) The logical query plan is:

   $$PkP \, \gamma \, \text{SUM(UnitPrice*Qty)} \textbf{ AS } TR$$

   $$\bowtie \\ FkP = PkP$$
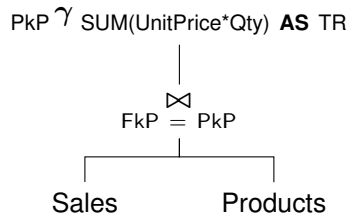
   Sales   Products

   The type of the result is $\{(\text{ PkP:int, TR:int })\}$. The value of the result is:

   Products

PkP	TR
10	550
20	1100
30	600

   The modified logical query plan is:

   $$\sigma_{C \, > \, 5}$$

   $$PkP \, \gamma \, \text{SUM(UnitPrice*Qty)} \textbf{ AS } TR, \text{COUNT(*)} \textbf{ AS } C$$

   $$\bowtie \\ FkP = PkP$$

   Sales $\sigma_{\text{UnitPrice} \, > \, 5}$
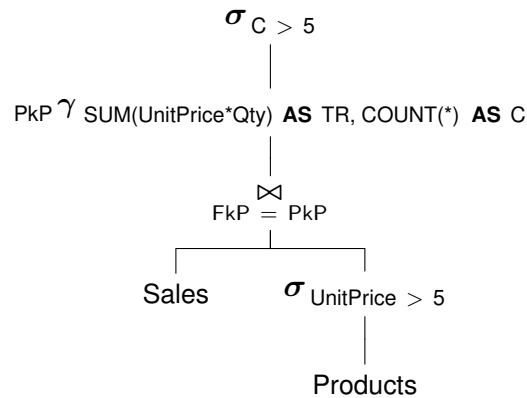
   Products

   (c) The modified SQL query is:

SELECT	PkP, SUM(UnitPrice*Qty) **AS** TR, **RANK** () **OVER** (**ORDER BY** SUM(Qty) **DESC**) **AS** Rk
FROM	Sales, Products
WHERE	FkP = PkP
GROUP BY	PkP;

(d) and (e) These are the instances of the indexes:

Index on FkP		FCJI on UnitPrice	
FkP	**RID**	**UnitPrice**	**RID**
10	1	5	1
10	4	5	4
10	7	5	7
20	2	10	2
20	5	10	5
30	3	20	3
30	6	20	6

2. **WITH** tmp **AS** (
   SELECT customer_id, time_id,
         SUM(SUM(store_sales)) **OVER** (
          PARTITION BY customer_id
          ORDER BY time_id
          RANGE BETWEEN 45 **PRECEDING AND CURRENT ROW**) **AS** tot
   FROM   sales_fact
   GROUP BY customer_id, time_id
)
SELECT $*$
FROM   tmp
WHERE tot$>$10
ORDER BY customer_id, time_id;

3. (a) **Requirements specification and the conceptual design of a data mart for the inventory**. See lecture notes solution to Exercise A.4: Inventory.

   (b) **Logical design of the data mart and the SQL queries**. See lecture notes solution to Exercise A.4: Inventory.

   (c) **Physical query plan**. See lecture notes.

4. The application of the HRU algorithm yields:

	First Choice	Second Choice	Third Choice
N_2	$\mathbf{50 \times 5 = 250}$		
N_3	$25 \times 5 = 125$	$25 \times 2 = 50$	25
N_4	$80 \times 2 = 160$	$30 \times 2 = 60$	$\mathbf{30 \times 2 = 60}$
N_5	$70 \times 3 = 210$	$20 \times 3 = 60$	$20 \times 2 + 10 = 50$
N_6	$60 \times 2 = 120$	$\mathbf{60 + 10 = 70}$	
N_7	$99 \times 1 = 99$	$49 \times 1 = 49$	49
N_8	$90 \times 1 = 90$	$40 \times 1 = 40$	30

5. (a) Show if and how the **GROUP BY** can be brought forward on the table Calls.
   The selection on Year can be pushed on Calls below the join.
   The group-by can be pushed on Calls below the join because the *invariant grouping* property holds: the aggregate function SUM(Charge) uses an attribute from Calls, and (Year, PlanName $\to$ FkPlanId) because

$$(\text{Year, PlanName})^+ = \{\text{Year, PlanName, PkPlanId, FkPlanId}\}$$

4

$$\sigma_{\text{TC} > 1000}$$

$$_{\text{Year, PlanName}}\gamma_{\text{SUM(Charge)} \textbf{ AS } \text{TC}}$$

$$\sigma_{\text{Year} >= 2000 \text{ AND Year} <= 2005}$$

$$\bowtie_{\text{FkPlanId} = \text{PkPlanId}}$$

Calls   CallingPlans

Figure 1: Logical query plan

$$\pi^b_{\text{Year, PlanName, TC}}$$

$$\sigma_{\text{TC} > 1000}$$

$$\bowtie_{\text{FkPlanId} = \text{PkPlanId}}$$

$$_{\text{Year, FkPlanId}}\gamma_{\text{SUM(Charge)} \textbf{ AS } \text{TC}}$$ CallingPlans

$$\sigma_{\text{Year} >= 2000 \text{ AND Year} <= 2005}$$
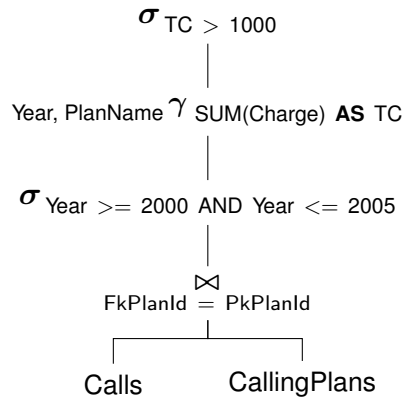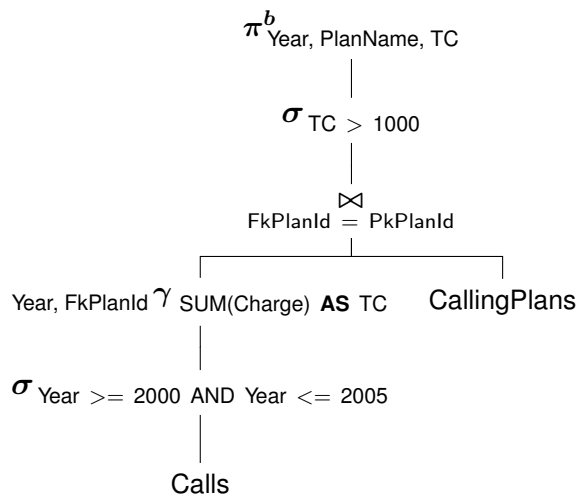
Calls

Figure 2: Logical query plan with the group-by pushed below the join

(b) Show if and how the query can be rewritten using the materialized view

V1:	**SELECT**	FkPlanId, Month, Year, SUM(Charge) **AS** C
	FROM	Calls
	WHERE	Year $>= 2000$
	GROUP BY	FkPlanId, Month, Year;

$$_{\text{FkPlanId, Month, Year}}\gamma_{\text{SUM(Charge)} \textbf{ AS } \text{C}}$$

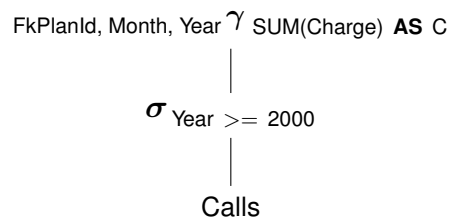$$\sigma_{\text{Year} >= 2000}$$

Calls

Figure 3: Logical view plan

Let us use the approach with a *transformation of the logical query plan* in Figure 2, which can be rewritten as follows with a subtree identical to V:

$$\pi^b_{\text{Year, PlanName, TC}}$$

$$\sigma_{\text{TC} > 1000}$$

$$\bowtie_{\text{FkPlanId} = \text{PkPlanId}}$$

$${}_{\text{Year, FkPlanId}}\gamma_{\text{SUM(C) AS TC}} \quad \text{CallingPlans}$$

$$\sigma_{\text{Year} <= 2005}$$

$${}_{\text{Month, Year, FkPlanId}}\gamma_{\text{SUM(Charge) AS C}}$$

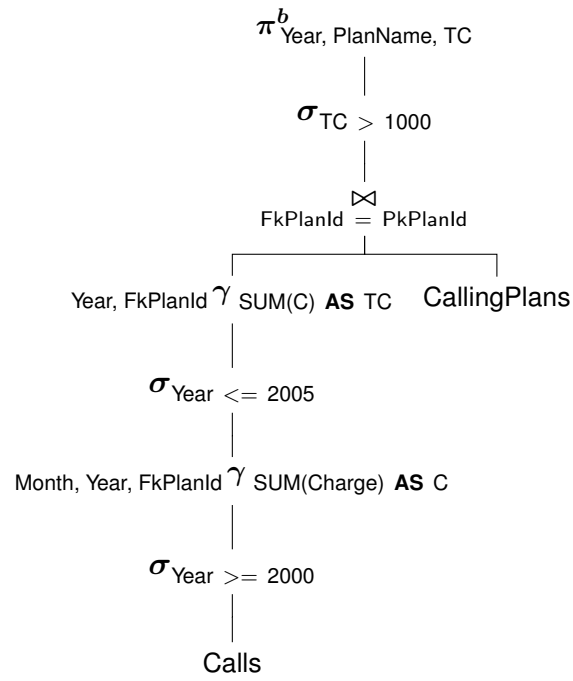$$\sigma_{\text{Year} >= 2000}$$

$$\text{Calls}$$

Figure 4: Logical query plan with the logical view subplan

The logical query plan with the logical view subplan is then rewritten as follows to translate it in SQL. With the rewriting of the group-by over the join, the attribute PlanName of CallingPlans, used by the projection, can be added to grouping attributes because (FkPlanId = PkPlanId) and PkPlanId $\rightarrow$ PlanName.

$$\pi^b_{\text{Year, PlanName, TC}}$$

$$\sigma_{\text{TC} > 1000}$$

$$\text{Year, FkPlanId, PlanName}\ \gamma\ \text{SUM(C)}\ \textbf{AS}\ \text{TC}$$

$$\sigma_{\text{Year} <= 2005}$$

$$\bowtie_{\text{FkPlanId} = \text{PkPlanId}}$$

Month, Year, FkPlanId γ SUM(Charge) **AS** C   CallingPlans

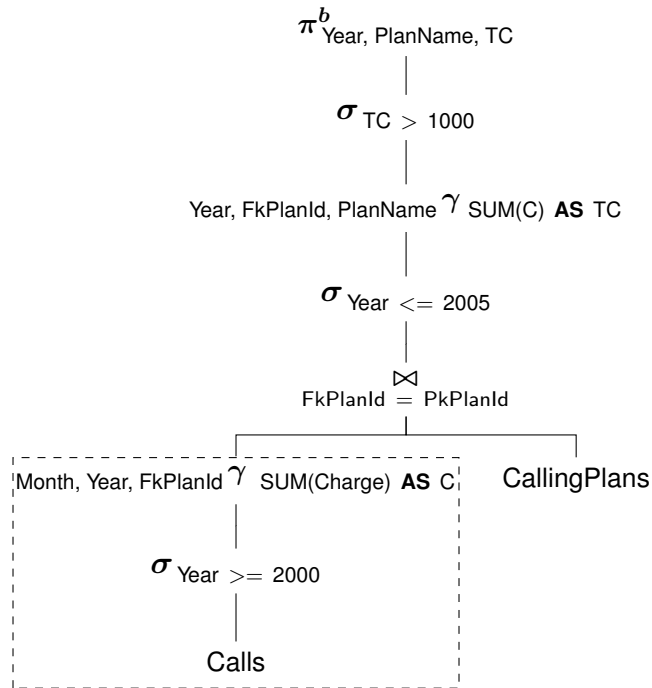$$\sigma_{\text{Year} >= 2000}$$

Calls

Figure 5: Final version of the logical query plan with the logical view subplan

So the rewriting of Q succeeds:

```
Q1": SELECT    Year, PlanName, SUM(C) AS TC
     FROM      V1, CallingPlans
     WHERE     FkPlanId = PkPlanId AND Year <=2005
     GROUP BY  Year, FkPlanId, PlanName
     HAVING    SUM(C) > 1000;
```

Exam of *Decision Support Systems / Decision Support Databases*: Example 2

*It is forbidden to consult any material during the test. Duration of written exam is 2h.*

1. **(Mandatory)** Let us consider the following database, without null values (F(Fk :int, B : int, C :int) and D(Pk :int, E :string)) and the query:

```
SELECT      Fk, COUNT(*) AS Cn
FROM        F, D
WHERE       Fk = Pk AND E <> 'd3'
GROUP BY    Fk
HAVING      SUM(C) < 100;
```

D

Pk	E
1	d1
2	d2
3	d3

F

Fk	B	C
1	10	60
1	20	20
2	30	80
2	20	25
3	30	80

   (a) **(1 point)** Give a logical query plan for the SQL query, the type and the value of the result.

   (b) **(1 point)** Show the instance of a bitmap index on the attribute Fk.

   (c) **(1 point)** Show the instance of a *Bitmap Foreign Column Join Index* on the attribute E.

   (d) **(2 points)** Explain the meaning of a "semi-additive" measure. Let F(FkD1, FkD2, M) be a table with the "semi-additive" measure M with respect to the dimension D1. Give a correct and a wrong query on the schema with the aggregation SUM(M).

2. **(5 points)** Consider the FoodMart datawarehouse:
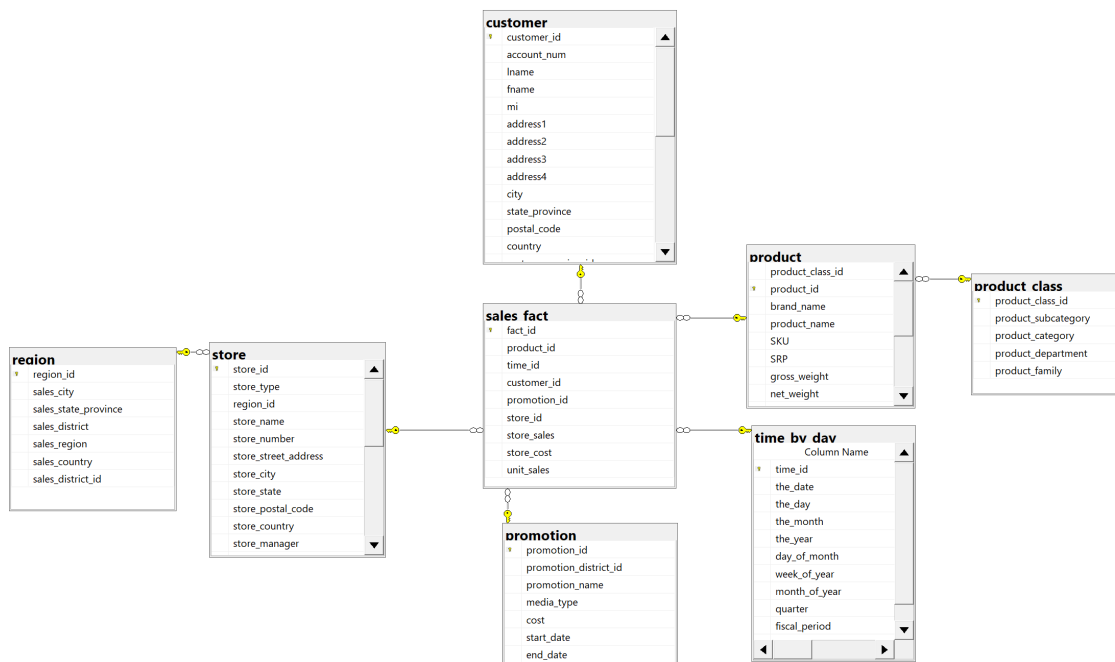
A customer is called a *commuter* if she buys at least 70% of her purchases (total store sales) in stores outside her city. Give an SQL query to find the number of commuters by customer city. The query must return 0 for cities with no commuter.

3. See lecture notes Exercise A.5: Hotels.
   With respect to the above business scenario, answer the following questions:

   (a) **(6 points)** Design a conceptual schema for the data mart to support the business questions. Your schema should at least be able to satisfy the above mentioned analysis requirements. You may motivate other suitable attributes for the dimensions. Specify the fact granularity, and for each measure, if it is additive, semi-additive or non-additive, and for each dimensional attribute the type of updates of the attribute.

   (b) **(3 points)** Give a logical data mart design which is able to deal with updates of dimensional attributes.

   (c) **(1 point)** Write an SQL query to answer one business questions of your choice and write a physical query plan for the query using any index of your choice.

4. **(5 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 3 views to materialize, different from N_1, with the greedy algorithm HRU. Determine the various possible results of HRU on the basis of the unknown value X.
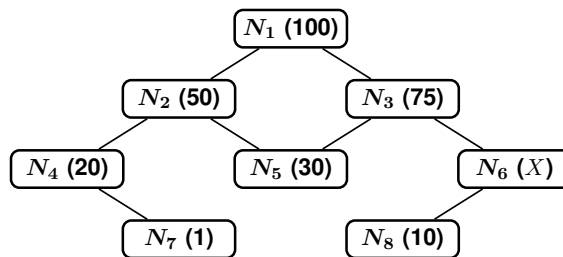
5. **(5 points)** Let us consider the database without null values:

   Customer(PKCustomer, CName, CCity)
   Order(PKOrder, FKCustomer, ODate)
   Product(PKProduct, PName, PCost)
   OrderLine(LineNo, FKOrder, FKProduct, Quantity, ExtendedPrice, Discount, Revenue)

   and the query

Q:	**SELECT**	CCity, AVG(Revenue) **AS** avgR
	FROM	OrderLine, Order, Customer
	WHERE	FKOrder = PKOrder **AND** FKCustomer = PKCustomer
	GROUP BY	CCity, FKCustomer
	HAVING	SUM(Revenue) > 1000;

   Answer **only one** of the following questions:

   (a) Show if and how the **GROUP BY** can be pushed on the join
      (OrderLine $\underset{\text{FKOrder = PKOrder}}{\bowtie}$ Order),
      and show if and how the **GROUP BY** can be pushed on the relation OrderLine.

   (b) Show if and how the query Q can be rewritten using the materialized view V

V:	**SELECT**	FKCustomer, SUM(Revenue) **AS** TR, COUNT(*) **AS** Cnt
	FROM	OrderLine, Order
	WHERE	FKOrder = PKOrder
	GROUP BY	FKCustomer;

Exam of *Decision Support Databases: Solution Example 2 (Hints)*

1. (a) The logical query plan is:

$$\pi^b_{\text{Fk, COUNT(}*\text{)} \textbf{ AS } \text{Cn}}$$

$$\sigma_{\text{SUM(C)} < 100}$$

$$_{\text{Fk}}\gamma_{\text{COUNT(}*\text{), SUM(C)}}$$

$$\bowtie_{\text{Fk = Pk}}$$

F $\sigma_{\text{E} <> \text{'d3'}}$

D
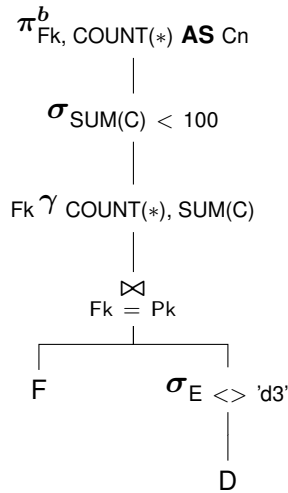
The type of the result is $\{(\text{ Fk:int, Cn:int })\}$. The value of the result is:

Products	
Fk	**Cn**
1	2

(b) and (c) These are the instances of the indexes are:

Bitmap Index on Fk			BFCJI on E	
Fk	**Bitmap**		**Fk**	**Bitmap**
1	11000		d1	11000
2	00110		d2	00110
3	00001		d3	00001

(d) A measure M is semi-additive w.r.t. a dimension D when it cannot be summed up for different values of dimension D. E.g., the monthly balance cannot be summed for different months, but it can be summed for different account holder at the same month. Here there are a correct and wrong query:

Correct:
```
SELECT      FkD1, SUM(M)
FROM        F
GROUP BY    FkD1;
```

Wrong:
```
SELECT      FkD2, SUM(M)
FROM        F
GROUP BY    FkD2;
```

2. **WITH** base **AS** (
        SELECT   F.customer_id, C.city,
                  CASE WHEN S.store_city=C.city **THEN** 0 **ELSE** 1 **END AS** other,
                  SUM(store_sales) **AS** purchases
        FROM   sales_fact F, customer C, store S
        WHERE   F.customer_id = C.customer_id **AND** F.store_id = S.store_id
        GROUP BY F.customer_id, C.city, **CASE WHEN** S.store_city=C.city **THEN** 0 **ELSE** 1 **END**)
), tmp **AS** (
        SELECT   customer_id, city, other,
                  100*SUM(purchases) **OVER** (**PARTITION BY** customer_id, other) /
                        SUM(purchases) **OVER** (**PARTITION BY** customer_id) **AS** Pct
        FROM   base
)
SELECT   city, SUM(**CASE WHEN** other=1 **AND** Pct $>=$ 70 **THEN** 1 **ELSE** 0 **END**) **AS** N
FROM   tmp
GROUP BY city;

3. (a) **Requirements specification and the conceptual design of a data mart for hotel room type utilization**. See lecture notes solution to Exercise A.5: Hotels.

   (b) **Logical design of the data mart and the SQL queries**. See lecture notes solution to Exercise A.5: Hotels.

   (c) **Physical query plan** for the query

      SELECT   H.Name
                  , **SUM**(F.NOccupiedRooms) / (**SUM**(F.NOccupiedRooms) +
                                      SUM(F.NVacantRooms) +
                                    SUM(F.NUnavailableRooms))
                    AS OccupancyRate
      FROM   RoomTypeUtilization F, Hotel H
      WHERE   F.HotelFK = H.HotelPK **AND** F.DateFK = 20100717
                  AND H.City = 'Florence'
      GROUP BY F.HotelFK, H.Name;

   Let us assume that there is (a) a bitmap index on the fact table attribute DateFK, (b) a bitmapped foreign column join index on the Hotel dimensional attribute City, and (c) an index on the primary key of the dimensional tables.
   Let us use the following abbreviations:

   – F for RoomTypeUtilization.

   – NOR for NOccupiedRooms.

   – NVR for NVacantRooms.

   – NUR for NUnavailableRooms.

   – OR for OccupancyRate.

   The physical query plan for the first data analysis is

Project
({Name,SUM(NOR)/(SUM(NOR)+SUM(NVR)+SUM(NUR)) AS OR})
|
HashGroupBy
({HotelFK, Name}, {SUM(NOR), SUM(NVR), SUM(NUR)})
|
IndexNestedLoop
(HotelFK=HotelPK)
|
┌──────────────┴──────────────┐
TableAccess   **IndexFilter**
(F)   (Hotel, IdxPkH, HotelPK=<u>HotelFK</u>)
|
RIDFromBM
|
BMAnd
|
┌──────────────┴──────────────┐
BMFCJIndexFilter   **BMIndexFilter**
(IdxHF, City = 'Florence')   (IdxDF, DateFK = 20100717)
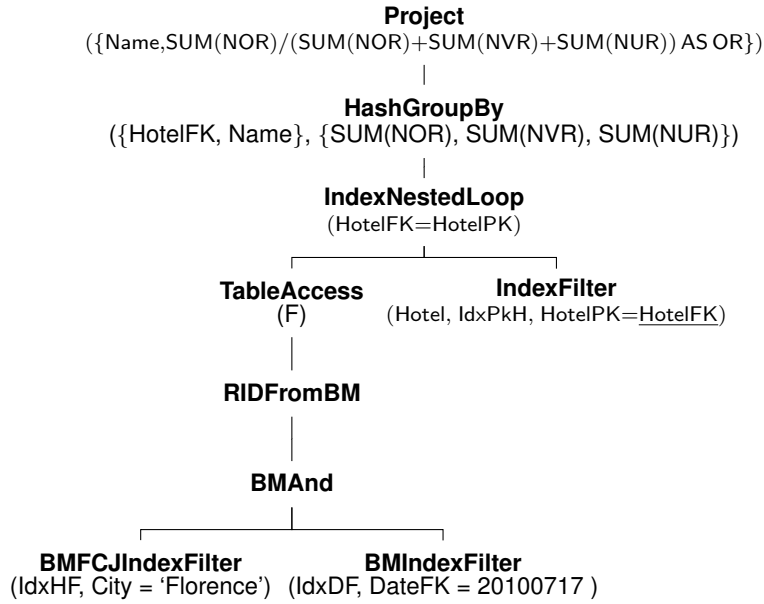
4. First, we observe that $10 \le X \le 75$. The application of the HRU algorithm yields:

	First Choice	Second Choice	Third Choice ($X \le 55$)	Third Choice ($X > 55$)
N_2	$50 \times 4 = 200$			
N_3	$25 \times 4 = 100$	$25 \times 3 = 75$	$25 \times 1 = 25$	$25 \times 2 = 50$
N_4	$80 \times 2 = 160$	$30 \times 2 = 60$	$\mathbf{30 \times 2 = 60}$	$\mathbf{30 \times 2 = 60}$
N_5	$70 \times 1 = 70$	$20 \times 1 = 20$	$20 \times 1 = 20$	$20 \times 1 = 20$
N_6	$(100 - X) \times 2$	$(100 - X) \times 2$		$(100 - X) \times 1$
N_7	$99 \times 1 = 99$	$49 \times 1 = 49$	$49 \times 1 = 49$	$49 \times 1 = 49$
N_8	$90 \times 1 = 90$	$90 \times 1 = 90$	$X - 10$	

Observe that:

- At the first choice, since $10 \le X$ then $(100 - X) \times 2 \le 180$. Hence N_2 is better than N_6.
- At the second choice, N_6 is selected if $(100 - X) \times 2 \ge 90$, which holds if $X \le 55$. Otherwise N_8 is selected.
- At the third choice with $X \le 55$, N_4 is selected because $X - 10 \le 45$.
- At the third choice with $X > 55$, N_4 is selected because $(100 - X) < 45$.

5. (a - first part) Show if and how the **GROUP BY** can be pushed on the join
   (OrderLine $\underset{\text{FKOrder = PKOrder}}{\bowtie}$ Order).

   The group-by can be pushed on the join (OrderLine $\underset{\text{FKOrder = PKOrder}}{\bowtie}$ Order) because the *invariant grouping* property holds: the aggregate functions use an attribute from the join result, and (CCity, FKCustomer → FKCustomer).
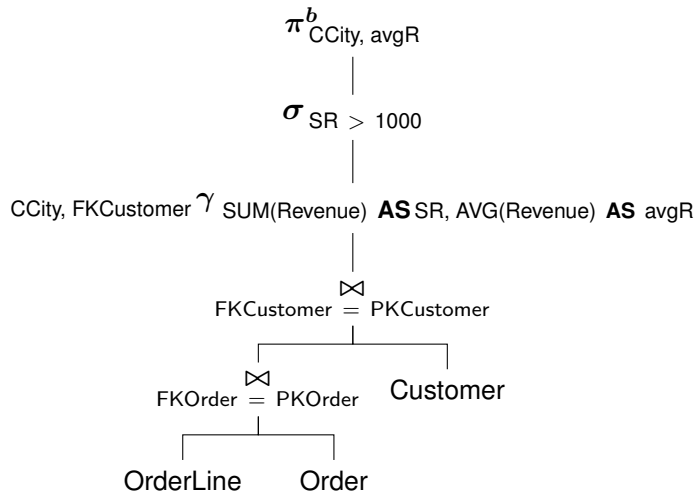
$$\pi^b_{\text{CCity, avgR}}$$

|

$$\sigma_{\text{SR} > 1000}$$

|

$${}_{\text{CCity, FKCustomer}}\gamma_{\text{SUM(Revenue) }\textbf{AS}\text{ SR, AVG(Revenue) }\textbf{AS}\text{ avgR}}$$

|

$$\bowtie_{\text{FKCustomer} = \text{PKCustomer}}$$

$$\bowtie_{\text{FKOrder} = \text{PKOrder}} \qquad \text{Customer}$$

OrderLine   Order

Figure 6: Logical query plan

$$\pi^b_{\text{CCity, avgR}}$$

|

$$\sigma_{\text{SR} > 1000}$$

|

$$\bowtie_{\text{FKCustomer} = \text{PKCustomer}}$$

$${}_{\text{FKCustomer}}\gamma_{\substack{\text{SUM(Revenue) }\textbf{AS}\text{ SR} \\ \text{, AVG(Revenue) }\textbf{AS}\text{ avgR}}} \qquad \text{Customer}$$

|

$$\bowtie_{\text{FKOrder} = \text{PKOrder}}$$
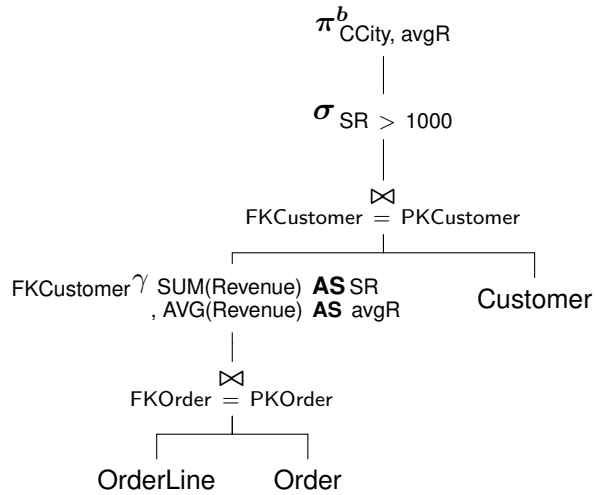
OrderLine   Order

Figure 7: Logical query plan: the **GROUP BY** is pushed below the first join with the invariant grouping

(a - second part) Show if and how the **GROUP BY** can be pushed on the relation OrderLine.

OrderLine does not have the invariant grouping property because Condition 1 does not hold: FKCustomer $\nrightarrow$ FKOrder. The double grouping can be applied, with the rewriting of the not decomposable aggregation function AVG(Revenue) as SUM(Revenue) / COUNT(Revenue), equivalent to SUM(Revenue) / COUNT($*$) because the data mart is without null values.
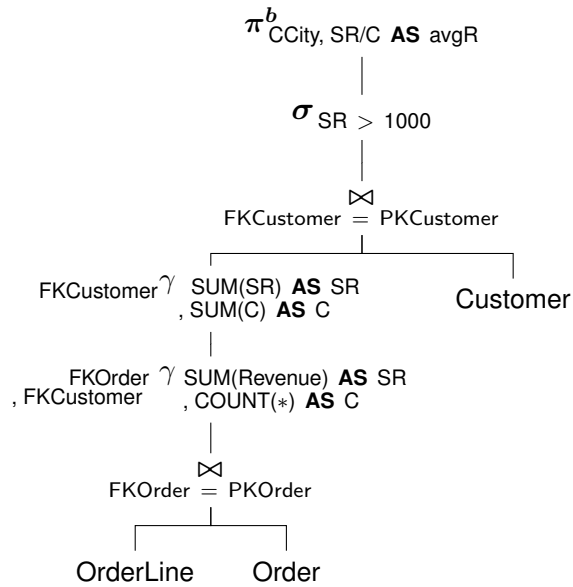
$$\pi^b_{\text{CCity, SR/C AS avgR}}$$

|

$$\sigma_{\text{SR} > 1000}$$

|

⋈
FKCustomer = PKCustomer

FKCustomer γ SUM(SR) **AS** SR
, SUM(C) **AS** C   Customer

|

FKOrder γ SUM(Revenue) **AS** SR
, FKCustomer   , COUNT(∗) **AS** C

|

⋈
FKOrder = PKOrder

OrderLine   Order

Figure 8: Logical query plan: the **GROUP BY** is rewritten with the double grouping and the rewriting of AVG

$$\pi^b_{\text{CCity, SR/C AS avgR}}$$

|

$$\sigma_{\text{SR} > 1000}$$

|

⋈
FKCustomer = PKCustomer

FKCustomer γ SUM(SR) **AS** SR
, SUM(C) **AS** C   Customer

|

⋈
FKOrder = PKOrder

FKOrder γ SUM(Revenue) **AS** SR
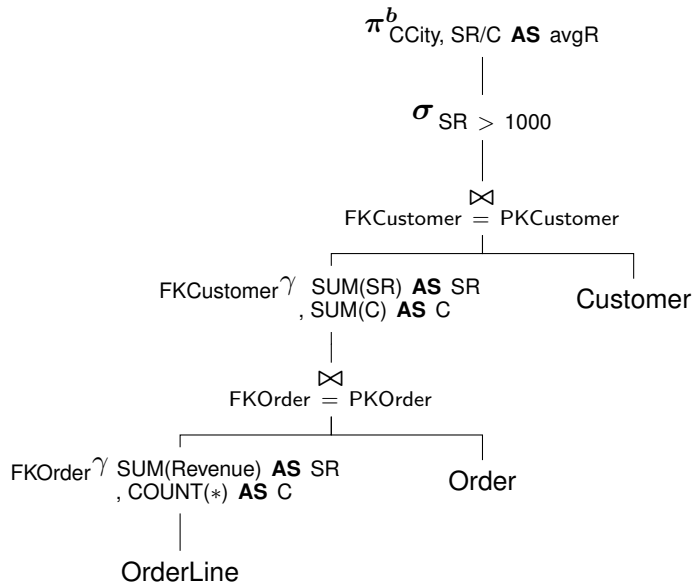, COUNT(∗) **AS** C   Order

|

OrderLine

Figure 9: Logical query plan: the second **GROUP BY** is pushed below the second join with the invariant grouping

(b) Show if and how the query Q can be rewritten using the materialized view V

Q:	**SELECT**	CCity, AVG(Revenue) **AS** avgR
	FROM	OrderLine, Order, Customer
	WHERE	FKOrder = PKOrder **AND** FKCustomer = PKCustomer
	GROUP BY	CCity, FKCustomer
	HAVING	SUM(Revenue) > 1000;
V:	**SELECT**	FKCustomer, SUM(Revenue) **AS** TR, COUNT(∗) **AS** Cnt

14

FROM	OrderLine, Order
WHERE	FKOrder = PKOrder
GROUP BY	FKCustomer;

Let us use the approach with a *compensation on the view*.

Since the approach requires that the **SELECT** and **HAVING** clauses may contain only the aggregate functions MIN, MAX, SUM and COUNT, the AVG function in Q is rewritten to compute it from given values for SUM and COUNT.
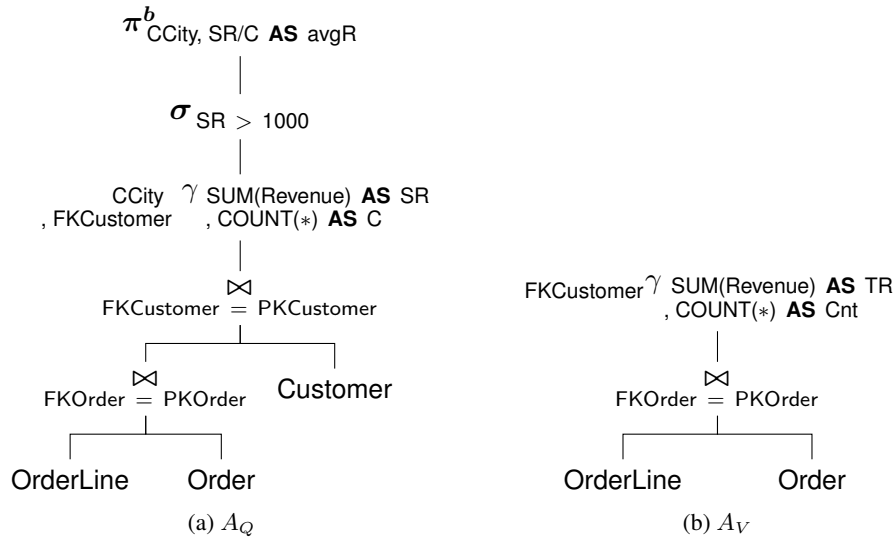

Figure 10: Query and view logical query plans

The join operations do not match. Therefore, a compensation is added as shown in the figure:
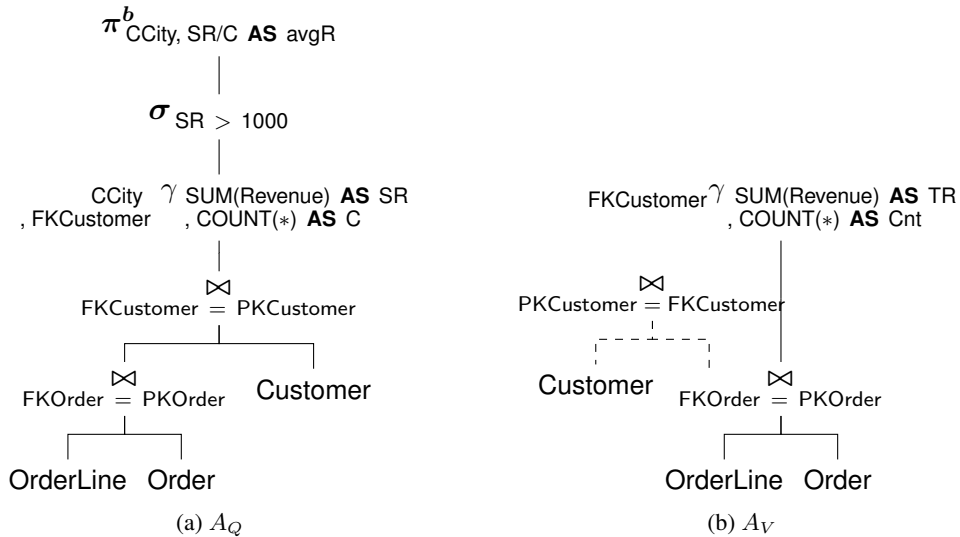

Figure 11: Join compensation

15

To match the groupings, the compensation on the operand of γ_V floats, and since $g(Q) \rightarrow g(V) \wedge g(V) \rightarrow g(Q)$, the rewriting does not require a grouping compensation, but an aggregate compensation only with a project, as shown in the figure:
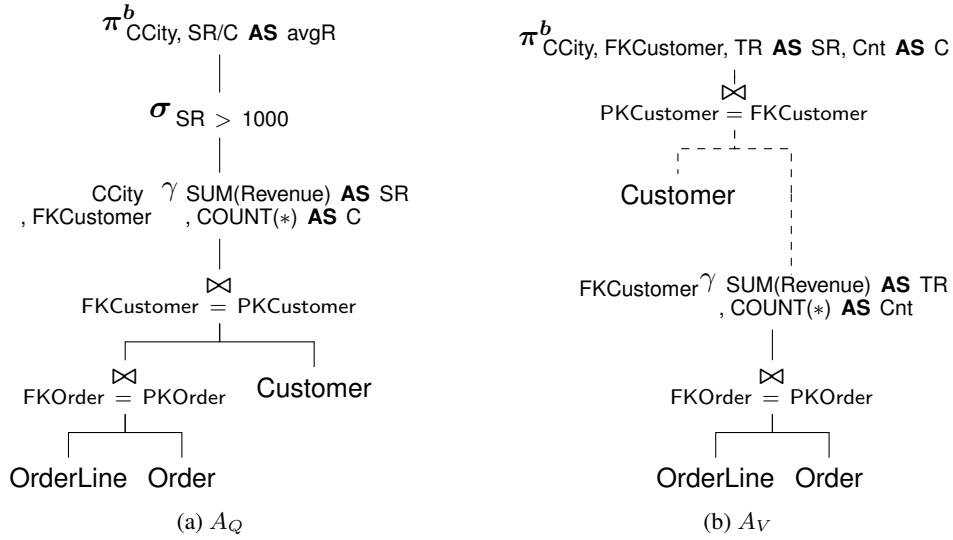

Figure 12: The float of the join compensation and the groupings compensation

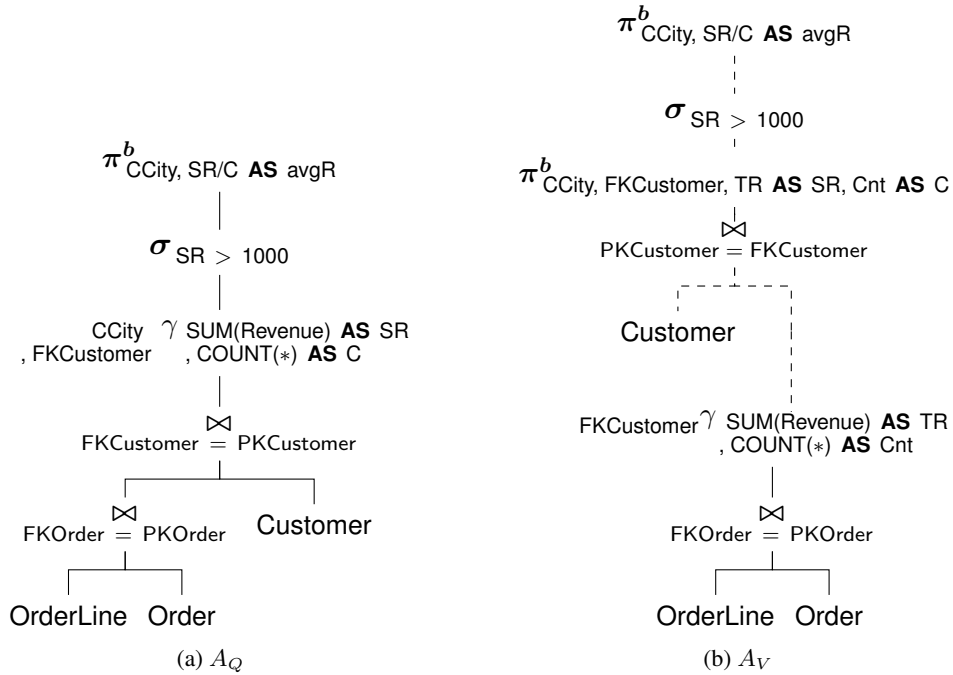The other compensations required for the σ and π^b in Q are shown in the figure:

Figure 13: Other compensations

Since the internal π^b of the compensation is useless, the final solution is the following:

$$\pi^b_{\text{CCity, SR/C AS avgR}}$$

$$\sigma_{\text{SR} > 1000}$$

CCity γ SUM(Revenue) **AS** SR
, FKCustomer , COUNT(∗) **AS** C

⋈
FKCustomer = PKCustomer

⋈
FKOrder = PKOrder   Customer

OrderLine Order

(a) A_Q

$$\pi^b_{\text{CCity, TR/Cnt AS avgR}}$$

$$\sigma_{\text{TR} > 1000}$$

⋈
PKCustomer = FKCustomer

Customer

FKCustomer γ SUM(Revenue) **AS** TR
, COUNT(∗) **AS** Cnt
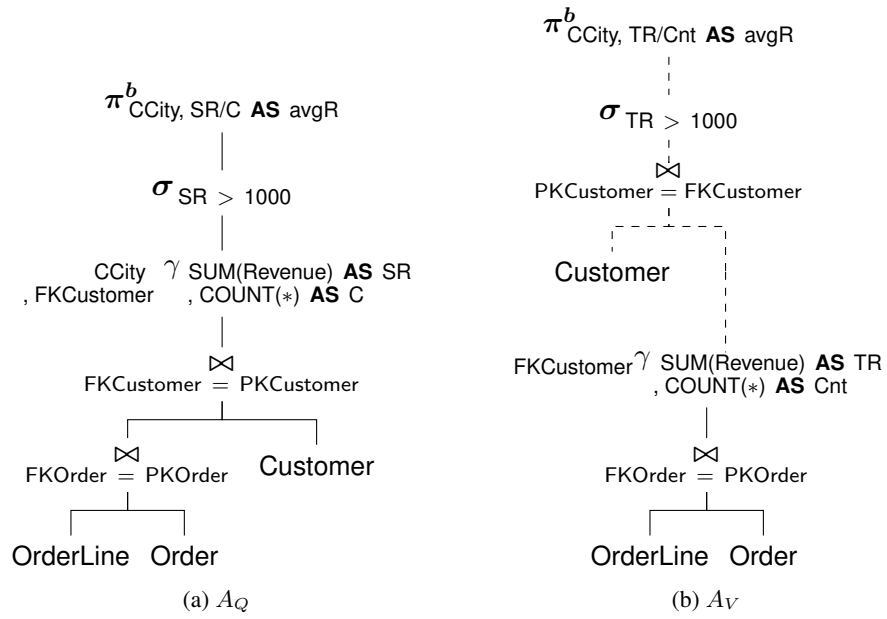
⋈
FKOrder = PKOrder

OrderLine Order

(b) A_V

Figure 14: Final solution

Rewriting of Q:

QR: **SELECT** CCity, TR/Cnt **AS** avgR
FROM V, Customer
WHERE FKCustomer = PKCustomer **AND** TR > 1000;

Let us use the approach with a *transformation of the logical query plan*.

The rewriting of Q with the γ pushed below the first join

$$(\text{OrderLine} \underset{\text{FKOrder = PKOrder}}{\bowtie} \text{Order})$$

produces a logical tree that, with the rewriting of AVG and the changing of SR and C in the γ with TR and Cnt, has as subtree the tree of the view.

Figure 15: The rewriting of Q to have V as a subtree

Rewriting of Q:

```
QR: SELECT      CCity, TR/Cnt AS avgR
    FROM        V, Customer
    WHERE       FKCustomer = PKCustomer AND TR > 1000;
```