RELATIONAL DBMS EXTENSIONS FOR DW

- SQL extensions
- Index and storage structures
- Star query physical plans
- Materialized views
- Optimization techniques for star queries with grouping and aggregations
- Query rewriting to use materialized views

THE QUERY REWRITING PROBLEM

A **materialized view V** is a query with a result that is materialized and stored in a table.

The **query rewriting problem:** *given a query Q and a materialized view V, is it possible to rewrite the query (plan of) Q using V?*

The query Q is not rewritten if the plan without V has a lower estimated computational cost than the plan using V, or if the plan with another materialized view has a lower cost.

- In general, a query rewritten in terms of a materialized view improves the execution of the query, because most of the query result has been precomputed.
- We omit cost-based evaluation of query plans in this course, see Advanced Databases.

Who write a query is not necessarily aware of materialized views, but the query optimizer consider the possibility of rewrite a query to use one of the available materialized views.

THE STAR SCHEMA: ASSUMPTIONS

Foreign and primary keys are the only attributes of the star schema with the same names and data types.

The fact and dimensional tables have attributes without null values.

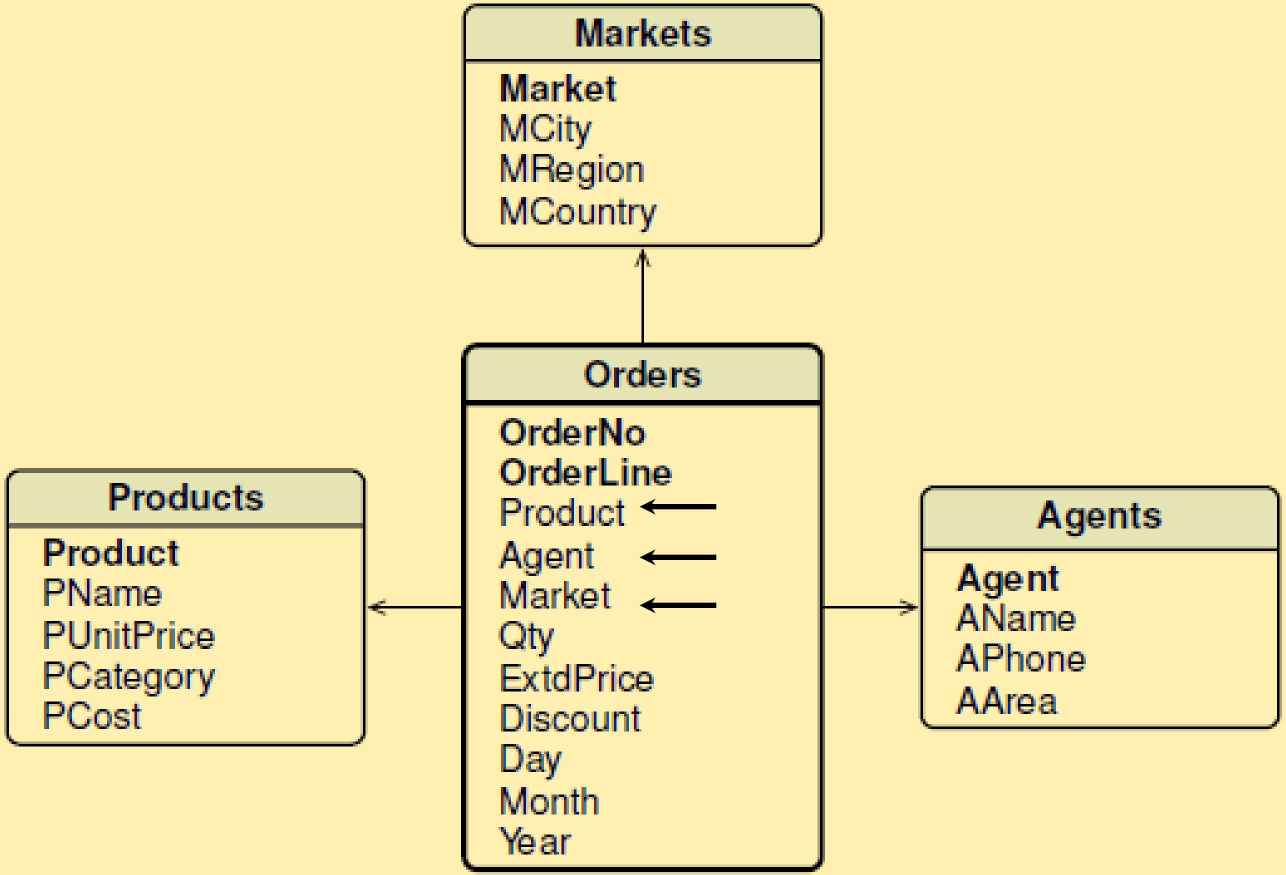
Star query only, namely natural joins of fact and dimension tables

- The joins are lossless and non-duplicating (a row in the fact table matches with one and only one row in the joined dimension tables)

Query rewriting in commercial DBMS make a number of such assumptions, e.g., by restricting the forms of admissible materialized views

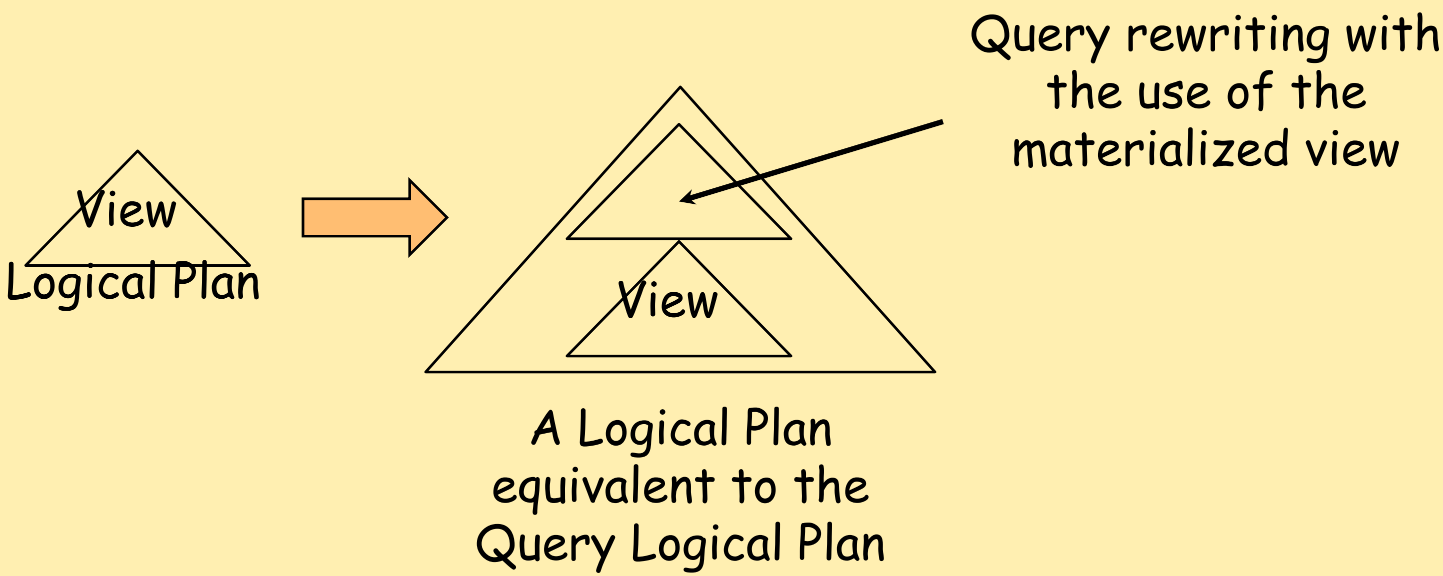
QUERY REWRITING TO USE MATERIALIZED VIEWS

Let us consider a database with a **Star Schema**.



QUERY REWRITING TO USE MATERIALIZED VIEWS: APPROACH 1

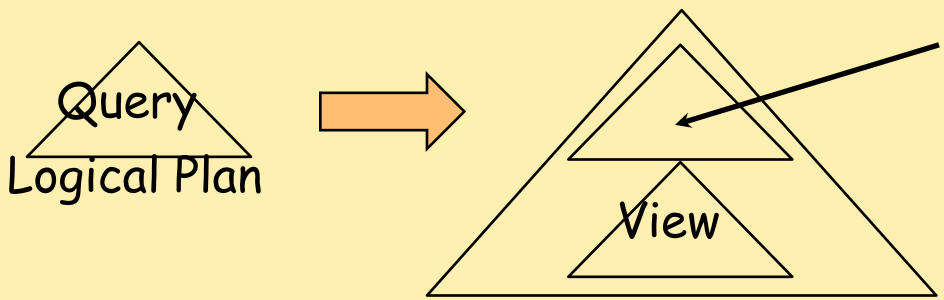
First approach. Add a logical plan to the **top of the View logical plan** so that the **global plan is equivalent to the Query logical plan**.



QUERY REWRITING TO USE MATERIALIZED VIEWS: APPROACH 2

Second approach. Transform the Query logical plan so that a **bottom portion** of it **is equivalent to the View logical plan**.

Query rewriting with the use of the materialized view



The Query Logical Plan with a bottom equivalent to View Logical Plan

EXAMPLE

A trivial case

$$Q = {}_{\mathrm{A}}\gamma_{\mathrm{SUM(C)}\ \mathbf{AS}\ \mathrm{S}}(R) \qquad V = {}_{\mathrm{A}}\gamma_{\mathrm{SUM(C)}\ \mathbf{AS}\ \mathrm{S}}(R)$$

R

A	B	C
a	2	2
a	2	3
b	3	4
b	3	5

A simple case

$$Q = {}_{\mathrm{A}}\gamma_{\mathrm{SUM(C)}\ \mathbf{AS}\ \mathrm{SQ}}(R) \qquad V = {}_{\mathrm{A,\ B}}\gamma_{\mathrm{SUM(C)}\ \mathbf{AS}\ \mathrm{SV}}(R)$$

Assume that **A** → **A,B** and **A,B** → **A** hold. |V| = |Q| ?

In general, that **g(Q)** → **g(V)** and **g(V)** → **g(Q)** hold.

Is Q rewritable using V ?

$$Q = \pi^{b}{}_{\mathrm{A,\ SV}\ \mathbf{AS}\ \mathrm{SQ}}(\underbrace{{}_{\mathrm{A,\ B}}\gamma_{\mathrm{SUM(C)}\ \mathbf{AS}\ \mathrm{SV}}(R)}_{V})$$

EXAMPLE

Another case

$$Q = {}_{\mathrm{A}}\gamma_{\mathrm{SUM(C)}\ \mathbf{AS}\ \mathrm{SQ}}(R) \qquad V = {}_{\mathrm{A,\ B}}\gamma_{\mathrm{SUM(C)}\ \mathbf{AS}\ \mathrm{SV}}(R)$$

Assume that **g(V) → g(Q) only holds.** |V| > |Q| ?

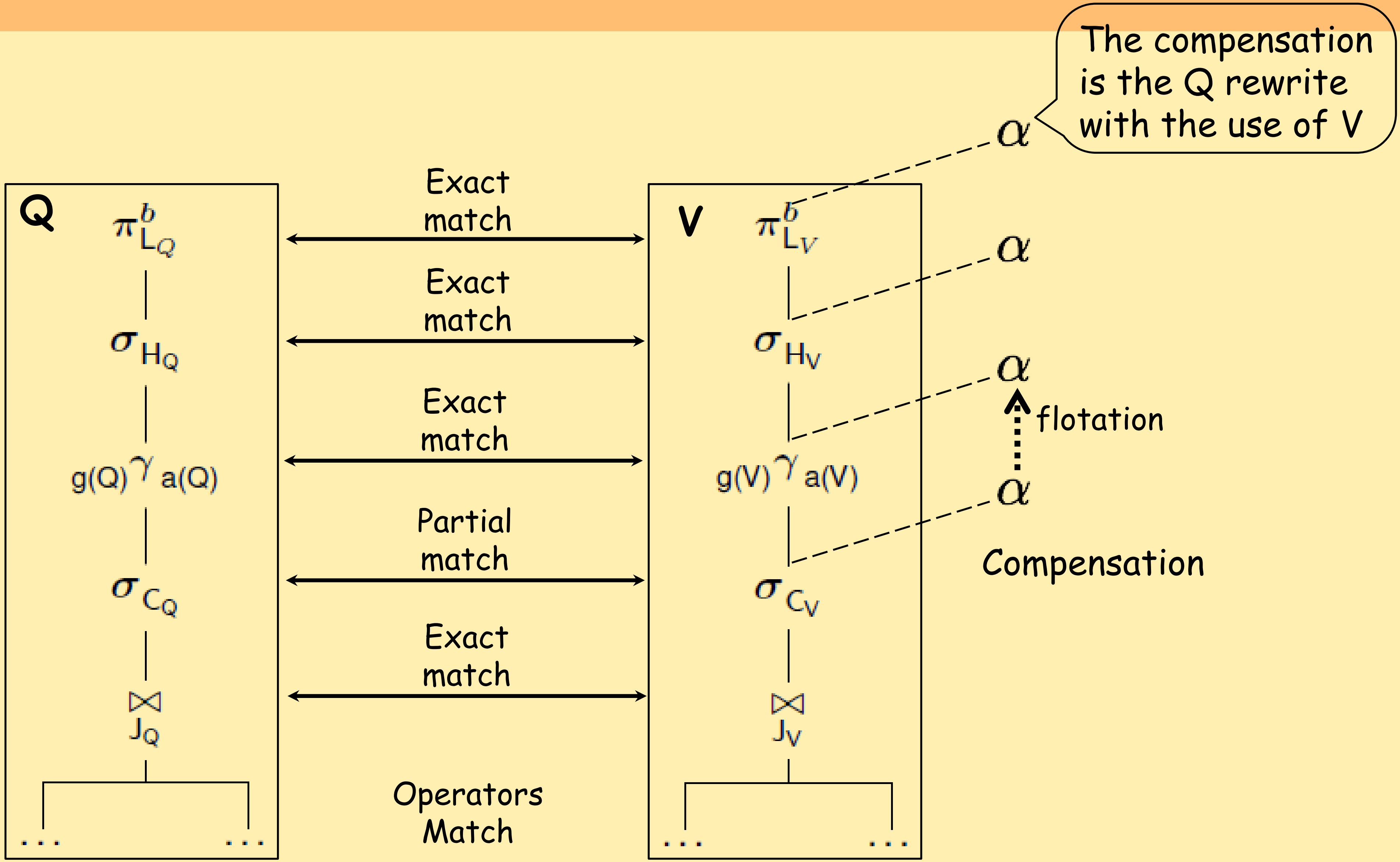
Is Q rewritable using V ?

$$Q = {}_{\mathrm{A}}\gamma_{\mathrm{SUM(SV)}\ \mathbf{AS}\ \mathrm{SQ}}(\underbrace{{}_{\mathrm{A,\ B}}\gamma_{\mathrm{SUM(C)}\ \mathbf{AS}\ \mathrm{SV}}(R)}_{V})$$

R

A	B	C
a	2	2
a	2	3
a	3	2
a	3	3
b	2	4
b	2	5
b	3	4
b	3	5

FIRST APPROACH: MATCHES AND COMPENSATIONS



FIRST APPROACH: MATCHES AND COMPENSATIONS

The algorithm idea.

The Q and V logical plans operators are pair-wise compared bottom-up, and if they are not **equivalent** a **compensation** is added to the view operator, that is a set of logical operations that have to be performed on the view operator to produce the same result.

When the matching view operator **has an operand with a compensation**, it must float on the operator because it must be included in the compensation for the operator match.

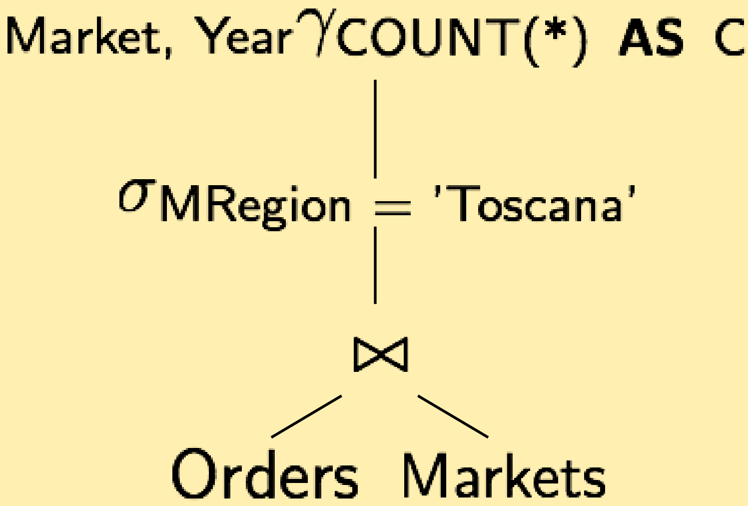
The final compensation on the view root is the rewriting of the query

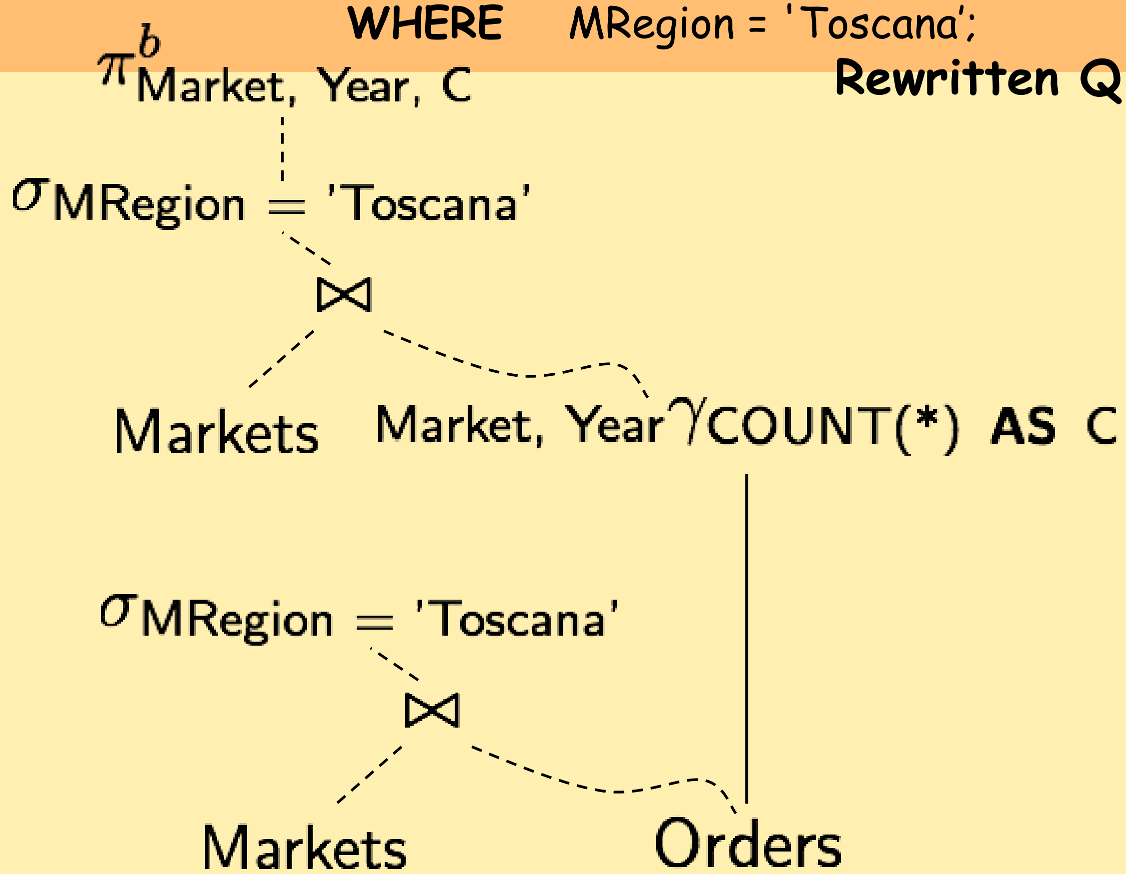
EXAMPLE: IS Q REWRITABLE WITH V ?

```
SELECT   Market, Year, C
FROM     Markets NATURAL JOIN V
WHERE    MRegion = 'Toscana';
```

Rewritten Q

Rewritable!






```
SELECT   Market, Year, COUNT(*) AS C
FROM     Orders NATURAL JOIN Markets
WHERE    MRegion = 'Toscana'
GROUP BY Market, Year;
```

Q

```
SELECT   Market, Year,COUNT(*) AS c
FROM     Orders
GROUP BY Market, Year;
```

V

$$A_Q = \gamma_{G_Q}(\sigma_{C_Q}(\bowtie R_Q)) \qquad A_V = \gamma_{G_V}(\sigma_{C_V}(\bowtie R_V))$$

$$R_V \subseteq R_Q$$

REWRITING ALGORITHM

1. $(\bowtie)$

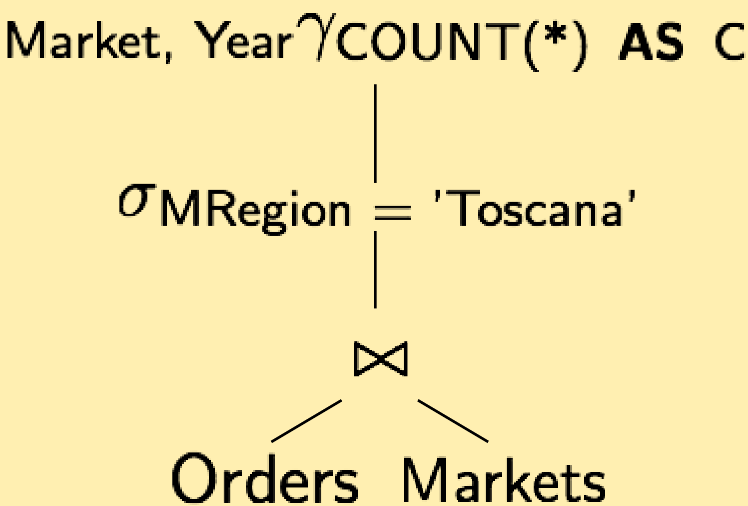
If the joins do not **match** (ie, $R_V \subset R_Q$), a **compensation** is added to the view operator as follows:

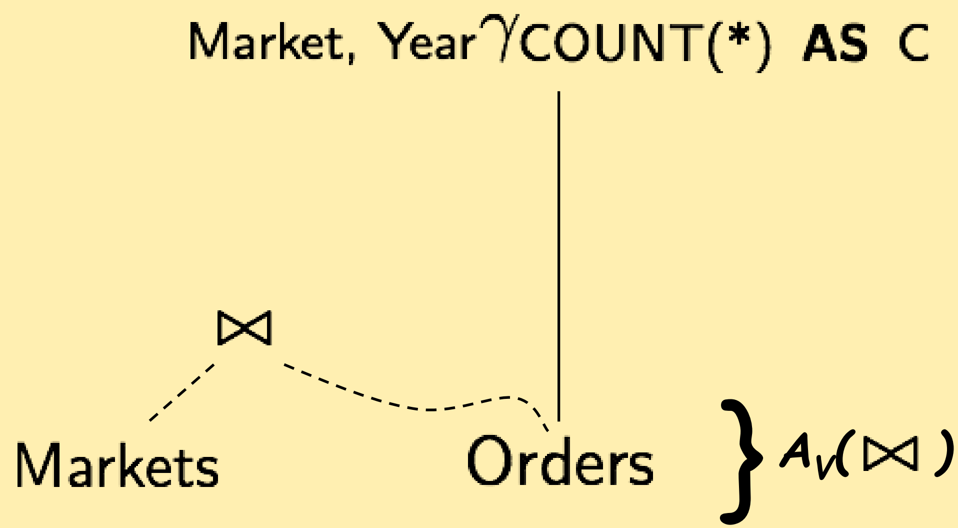
Let $W = R_Q - R_V$ be the tables in Q but not in V, then the compensation is the join $\alpha_{\bowtie} = (\bowtie W(A_V(\underbrace{\bowtie}_{\text{Root of compensation tree} = \bowtie R_V})))$

Condition: the compensation can float on γ_{G_V} if G_V contains the foreign keys for the tables W. **Otherwise Q is not rewritable**

EXAMPLE: IS Q REWRITABLE WITH V ?

Rewritable!





SELECT Market, Year, COUNT(*) **AS** C
FROM Orders **NATURAL JOIN** Markets
WHERE MRegion = 'Toscana'
GROUP BY Market, Year;

Q

SELECT Market, Year,COUNT(*) **AS** c
FROM Orders
GROUP BY Market, Year;

V

$$A_Q = \gamma_{G_Q}(\sigma_{C_Q}(\bowtie R_Q)) \qquad A_V = \gamma_{G_V}(\sigma_{C_V}(\bowtie R_V))$$
$$R_V \subseteq R_Q$$

REWRITING ALGORITHM

2. (σ)

If on the operand exists a compensation it floats on the operator.

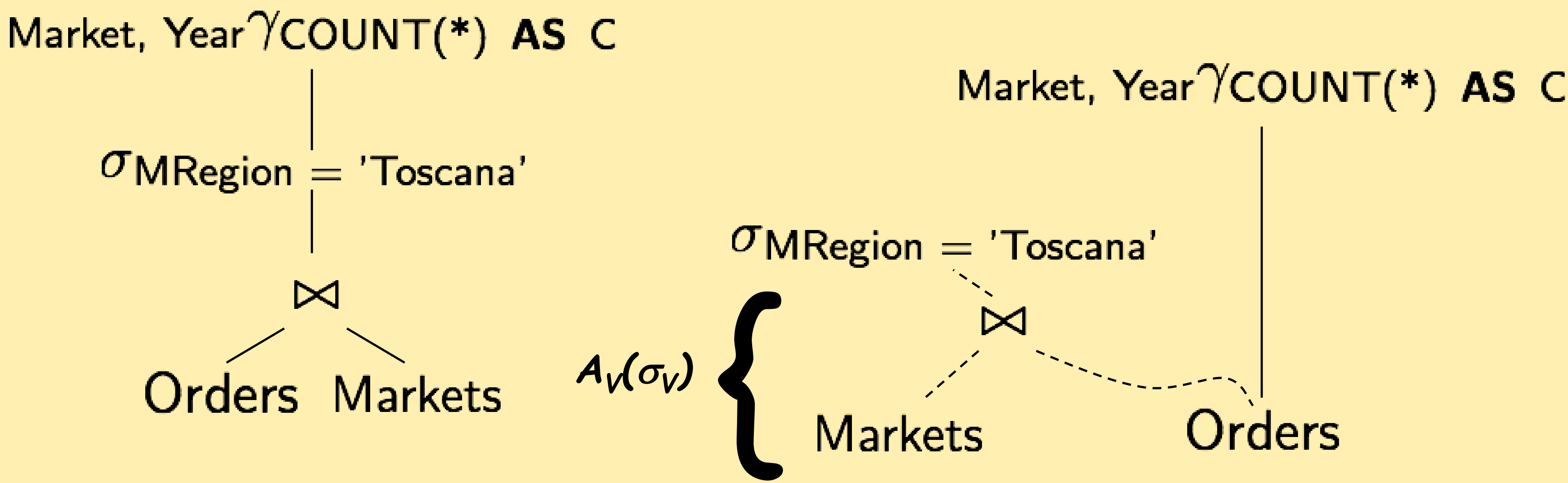
If the selections do not **match**, and $C_Q = C_V \wedge C$, then **compensation** to add is

$$\alpha_\sigma = \sigma_C(\underbrace{A_V}_{\text{Root of compensation tree}}(\sigma_V))$$

Condition: the compensation can float on γ_{G_V} if C uses only attributes in G_V or attributes of tables with foreign keys in G_V. **Otherwise Q is not rewritable**

EXAMPLE: IS Q REWRITABLE WITH V ?

Rewritable!




```
SELECT    Market, Year, COUNT(*) AS C
FROM      Orders NATURAL JOIN Markets
WHERE     MRegion = 'Toscana'
GROUP BY Market, Year;
```

Q

```
SELECT    Market, Year,COUNT(*) AS c
FROM      Orders
GROUP BY Market, Year;
```

V

$$A_Q = \gamma_{G_Q}(\sigma_{C_Q}(\bowtie R_Q)) \qquad A_V = \gamma_{G_V}(\sigma_{C_V}(\bowtie R_V))$$
$$R_V \subseteq R_Q$$

REWRITING ALGOR

see Lecture Notes for other aggregates

3. $({}_{g(Q)}\gamma_{SUM(A)\ AS\ S})$

If on the operand exists a compensation it floats on the operator.

Case WITHOUT grouping

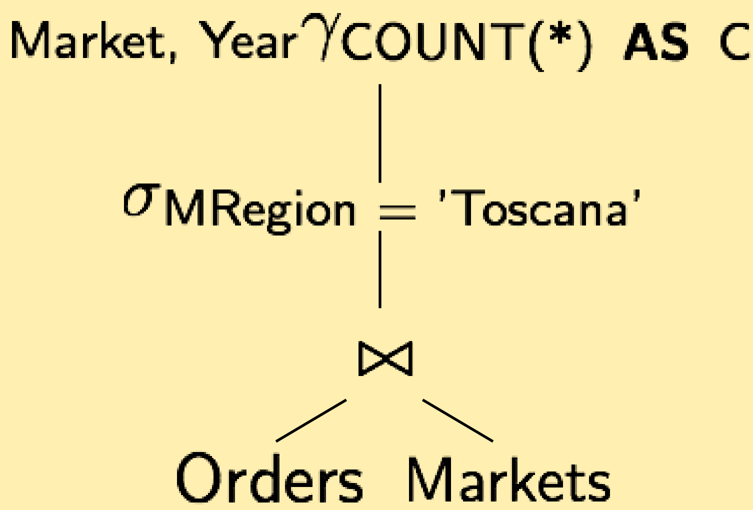
The rewriting is without grouping when the groupings in Q and V partition data into the same number of groups, e.g., if

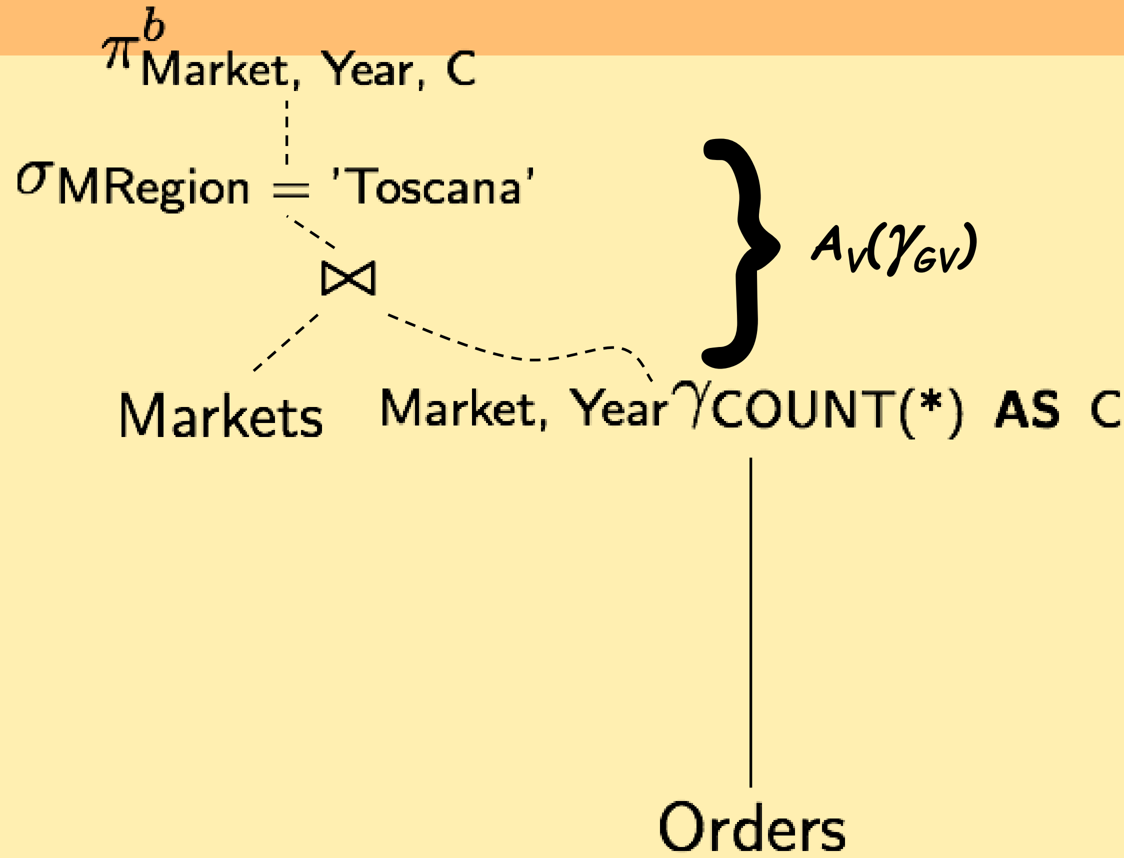
g(Q) → g(V) and g(V) → g(Q)

The **compensation** to add is $\alpha = \pi^b_{g(Q)\cup\{S\}}(\overbrace{A_V}^{\text{Root of compensation tree}}(\gamma_{GV}))$

EXAMPLE: IS Q REWRITABLE WITH V ?

Rewritable!






```
SELECT    Market, Year, COUNT(*) AS C
FROM      Orders NATURAL JOIN Markets
WHERE     MRegion = 'Toscana'
GROUP BY Market, Year;
```

Q

```
SELECT    Market, Year,COUNT(*) AS c
FROM      Orders
GROUP BY Market, Year;
```

V

$$A_Q = \gamma_{G_Q}(\sigma_{C_Q}(\bowtie R_Q)) \qquad A_V = \gamma_{G_V}(\sigma_{C_V}(\bowtie R_V))$$
$$R_V \subseteq R_Q$$

REWRITING ALGORITHM

3. $({}_{g(Q)}\gamma_{\text{SUM(A) AS S}})$

If on the operand exists a compensation it floats on the operator.

Case WITH grouping

The rewriting is without grouping when the groupings V partitions data into more groups than the grouping in Q, e.g., if

$$\mathbf{g(V) \rightarrow g(Q)}$$

The **compensation** to add is $\alpha = {}_{g(Q)}\gamma_{\text{SUM(S') AS S}}(\overbrace{A_V(\gamma_{G_V})}^{\text{Root of compensation tree}})$ where **SUM(A) AS S' is in G_V**.

EXERCISE: IS Q REWRITABLE WITH V ?

Rewritable ?

$$_{\text{Market}}\gamma_{\text{COUNT(*) AS C}}\left(\sigma_{\text{MRegion} = \text{'Toscana'}}\left(\text{Orders} \bowtie \text{Markets}\right)\right)$$

$$_{\text{Market, Year}}\gamma_{\text{COUNT(*) AS C}}\left(\text{Orders}\right)$$

```
SELECT     Market, COUNT(*) AS C
FROM       Orders NATURAL JOIN Markets
WHERE      MRegion = 'Toscana'
GROUP BY Market;
```

Q

```
SELECT     Market, Year,COUNT(*) AS c
FROM       Orders
GROUP BY Market, Year;
```

V

EXERCISE AT HOME: IS Q REWRITABLE WITH V ?

Rewritable ?

$$_{\text{Market}}\gamma_{\text{COUNT(*) }\mathbf{AS}\text{ C}}\left(\sigma_{\text{MRegion} = \text{'Toscana' } \boxed{\text{AND Year} = 2013}}\left(\text{Orders} \bowtie \text{Markets}\right)\right)$$

$$_{\text{Market, Year}}\gamma_{\text{COUNT(*) }\mathbf{AS}\text{ C}}\left(\text{Orders}\right)$$

SELECT Market, COUNT(*) **AS** C
FROM Orders **NATURAL JOIN** Markets
WHERE MRegion = 'Toscana'
AND Year = 2013
GROUP BY Market;

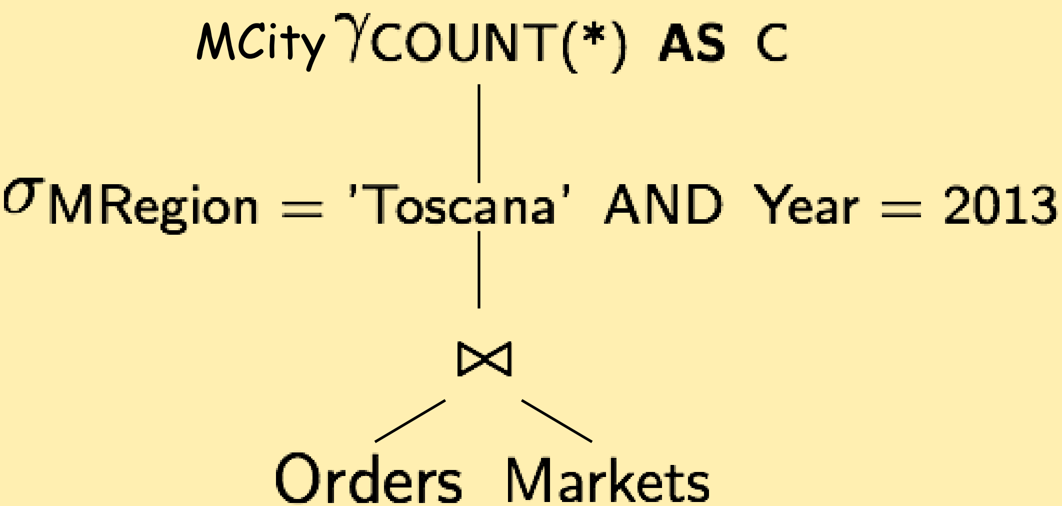
Q

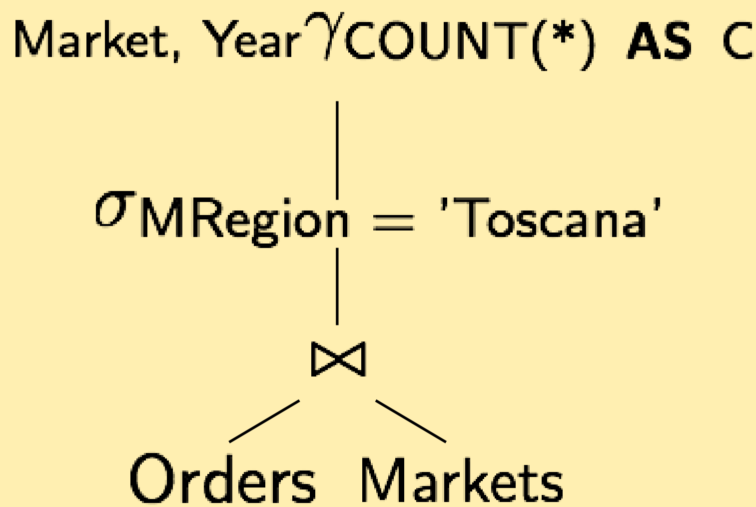
SELECT Market, Year,COUNT(*) **AS** c
FROM Orders
GROUP BY Market, Year;

V

EXERCISE: IS Q REWRITABLE WITH V ?

Rewritable ?






```
SELECT    MCity, COUNT(*) AS C
FROM      Orders NATURAL JOIN Markets
WHERE     MRegion = 'Toscana'
          AND Year = 2013
GROUP BY MCity;
```

Q

```
SELECT    Market, Year, COUNT(*) AS c
FROM      Orders NATURAL JOIN Markets
WHERE     MRegion = 'Toscana'
GROUP BY Market, Year;
```

V

$$A_Q = \gamma_{G_Q}(\sigma_{C_Q}(\bowtie R_Q)) \qquad A_V = \gamma_{G_V}(\sigma_{C_V}(\bowtie R_V))$$
$$R_V \subseteq R_Q$$

REWRITING ALGORITHM

3. $({}_{g(Q)}\gamma_{SUM(A)\ AS\ S})$

If on the operand exists a compensation it floats on the operator.

Case WITH grouping

The rewriting is without grouping when the groupings V partitions data into more groups than the grouping in Q, e.g., if

$$g(V) \rightarrow g(Q)$$

The **compensation** to add is $\alpha = {}_{g(Q)}\gamma_{SUM(S')\ AS\ S}(\overbrace{A_V(\gamma_{GV})}^{\text{Root of compensation tree}} \bowtie R)$

where the g(Q) attributes missing in $A_V(\gamma_{GV})$ can be retrieved with a lossless and non-duplicating join of $A_V(\gamma_{GV})$ with R, and **SUM(A) AS S' is in G_V**.

For example, $A_V(\gamma_{GV})$ contains the foreign keys fk of R.

IS Q REWRITABLE WITH V ?

Q:
```
SELECT   Market, SUM(Qty)
FROM     Orders
GROUP BY Market;
```

V:
```
SELECT   Market, SUM(Qty)
FROM     Orders NATURAL JOIN Markets
WHERE    MRegion = 'Toscana'
GROUP BY Market;
```

Q:
```
SELECT   SUM(Qty) AS Q
FROM     Orders
GROUP BY Market;
```

V:
```
SELECT   SUM(Qty) AS Q
FROM     Orders;
```

Q:
```
SELECT   SUM(Qty) AS Q
FROM     Orders
GROUP BY Market;
```

V:
```
SELECT   MAX(Qty) AS Q
FROM     Orders
GROUP BY Market;
```

SECOND APPROACH TO QUERY REWRITING: EXAMPLE

```
Q: SELECT   Product, SUM(Qty) AS T
   FROM     Orders NATURAL JOIN Markets
   WHERE    MRegion = 'Toscana'
   GROUP BY Product;
```

```
V: SELECT   Product, Market,
                SUM(Qty) AS S
   FROM     Orders
   GROUP BY Product, Market;
```

Rewritable?

SECOND APPROACH TO QUERY REWRITING: EXAMPLE

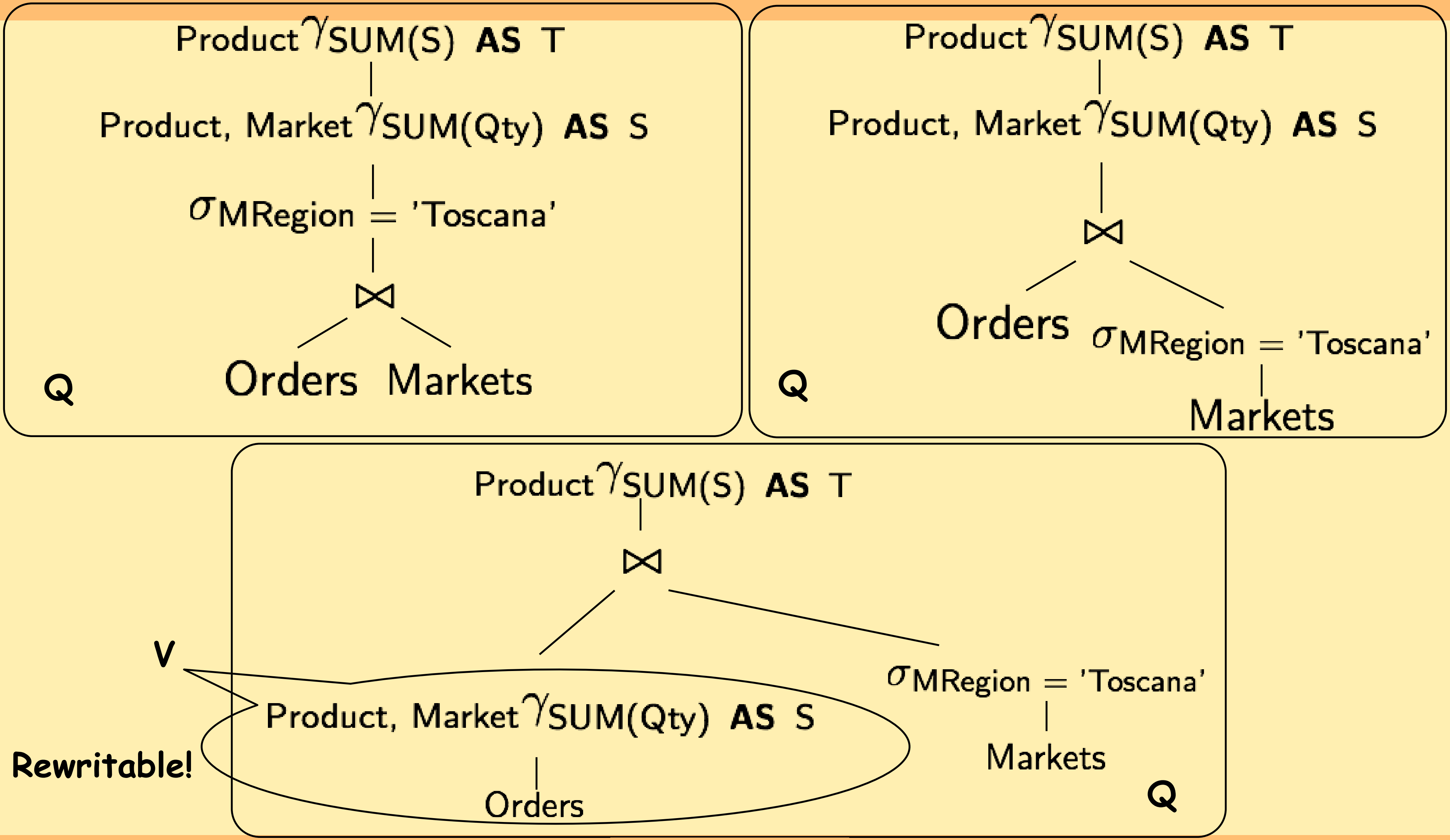
Transform the Query logical plan so that a **bottom portion** of it **is equivalent to the View logical plan**.

$$\text{Q:}\quad {}_{\text{Product}}\gamma_{\text{SUM(Qty)}\ \textbf{AS}\ \text{T}}\left(\sigma_{\text{MRegion} = \text{'Toscana'}}\left(\text{Orders} \bowtie \text{Markets}\right)\right)$$

$$\text{V:}\quad {}_{\text{Product, Market}}\gamma_{\text{SUM(Qty)}\ \textbf{AS}\ \text{S}}\left(\text{Orders}\right)$$

$$\text{Q:}\quad {}_{\text{Product}}\gamma_{\text{SUM(S)}\ \textbf{AS}\ \text{T}}\left({}_{\text{Product, Market}}\gamma_{\text{SUM(Qty)}\ \textbf{AS}\ \text{S}}\left(\sigma_{\text{MRegion} = \text{'Toscana'}}\left(\text{Orders} \bowtie \text{Markets}\right)\right)\right)$$

SECOND APPROACH TO QUERY REWRITING: EXAMPLE



SECOND APPROACH TO QUERY REWRITING: EXAMPLE

```
Q: SELECT    Product,  SUM(Qty) AS T
   FROM      Orders NATURAL JOIN Markets
   WHERE     MRegion = 'Toscana'
   GROUP BY  Product;
```

```
V: SELECT    Product, Market,
                   SUM(Qty) AS S
   FROM      Orders
   GROUP BY  Product, Market;
```

Rewritable!

Rewriting of Q using the second approach (but also using the first approach)

```
SELECT    Product,  SUM(S) AS T
FROM      V NATURAL JOIN Markets
WHERE     MRegion = 'Toscana'
GROUP BY  Product;
```

In general, the first and the second approach may yield different rewriting!

STRUCTURE OF EXAMS

Mandatory teaching material

- **[DW]** A. Albano, S. Ruggieri. Decision Support Databases Essentials, University of Pisa, 2 December 2021
- **[DB]** A. Albano. DB Essentials and solutions to exercises, University of Pisa, 1 December 2020. This is a English) from the book Fondamenti di basi di dati (in Italian, free download).
- Examples of written exams with solutions and written exam.

- **Written test** (2 hours)
  - See website for
    - Examples of written text
    - Dates and registration
  - Grading >= 17 admission to the oral part

- **Oral part**
  - Discussion of written part
  - Open questions on all topics of the course
  - Questions/small exercises using JRS and/or SQL Server

- **Overall Decision Support Systems (12 ECTS)**
  - Final grade is the average of the two modules DSD and LDS

TIME FOR FILLING STUDENT'S QUESTIONNARIES

https://esami.unipi.it