Data Warehouse Logical Design

Letizia Tanca
Politecnico di Milano
(with the kind support of
Rosalba Rossato)

Data Mart logical models

- MOLAP (*Multidimensional On-Line Analytical Processing*) stores data by using a multidimensional data structure

- ROLAP (*Relational On-Line Analytical Processing*) uses the relational data model to represent multidimensional data

Data Mart logical models

MOLAP stands for Multidimensional OLAP. In MOLAP cubes the data aggregations and a copy of the fact data are stored in a multidimensional structure on the computer. It is best when extra storage space is available on the server and the best query performance is desired. MOLAP local cubes contain all the necessary data for calculating aggregates and can be used offline. MOLAP cubes provide the fastest query response time and performance but require additional storage space for the extra copy of data from the fact table.

ROLAP stands for Relational OLAP. ROLAP uses the relational data model to represent multidimensional data. In ROLAP cubes a copy of data from the fact table is not (necessarily) made and the data aggregates may be stored in tables in the source relational database. A ROLAP cube is best when there is limited space on the server and query performance is not very important. ROLAP local cubes contain the dimensions and cube definitions but aggregates are calculated when needed. ROLAP cubes requires less storage space than MOLAP and HOLAP cubes.

HOLAP stands for Hybrid OLAP. A HOLAP cube has a combination of the ROLAP and MOLAP cube characteristics. It does not necessarily create a copy of the source data; however, data aggregations are stored in a multidimensional structure on the server. HOLAP cubes are best when storage space is limited but faster query responses are needed.

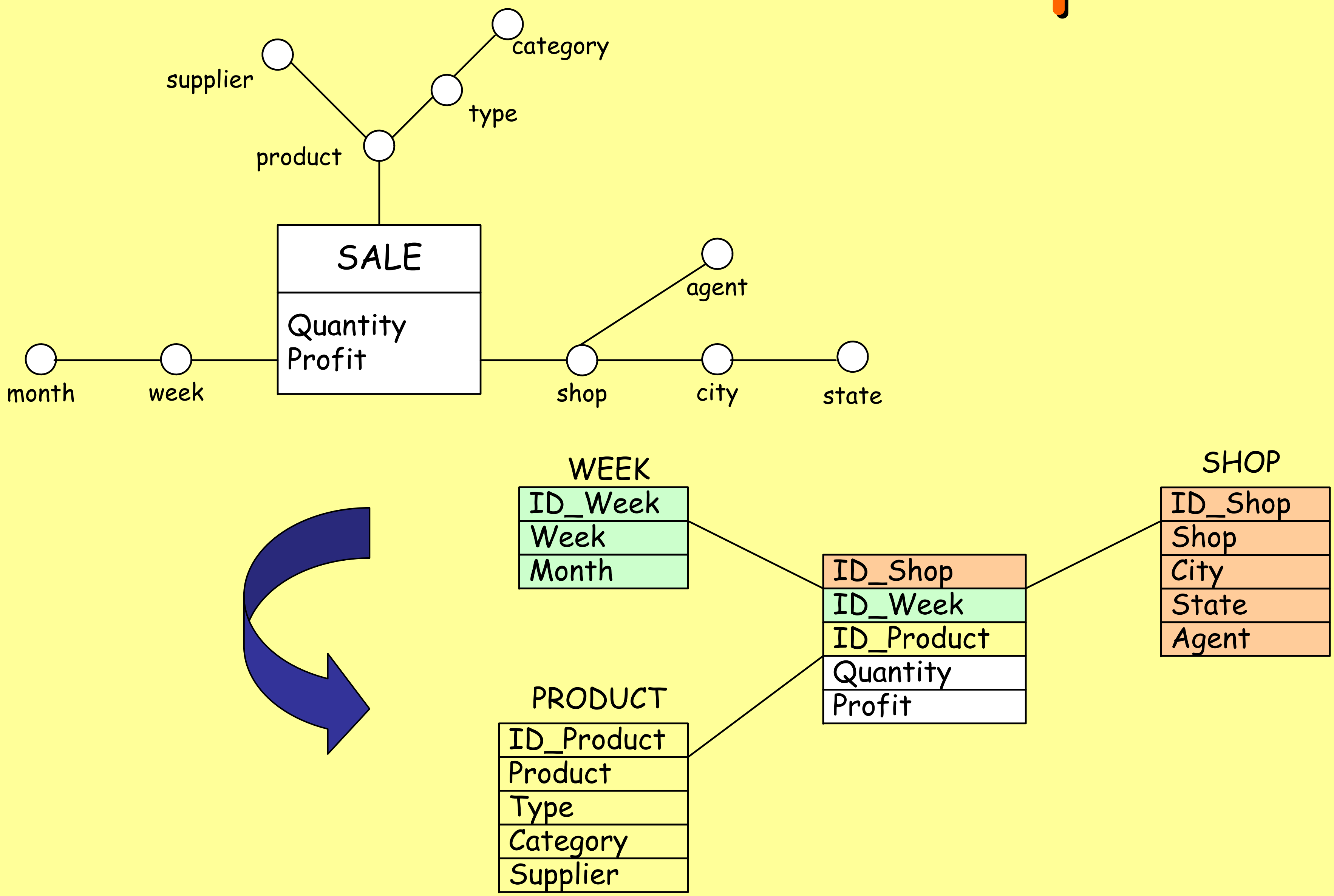
ROLAP

It is based on the Star Schema

A star schema is :

- ✓ A set of relations DT_1, DT_2, …DT_n - dimension tables - each corresponding to a dimension.
- ✓ Each DT_i is characterized by a primary key d_i and by a set of attributes describing the analysis dimensions with different aggregation levels
- ✓ A relation FT, fact table, that imports the primary keys of dimensions tables. The primary key of FT is $d_1\ d_2\ \dots\ d_n$; FT contains also an attribute for each measure

Star schema: example

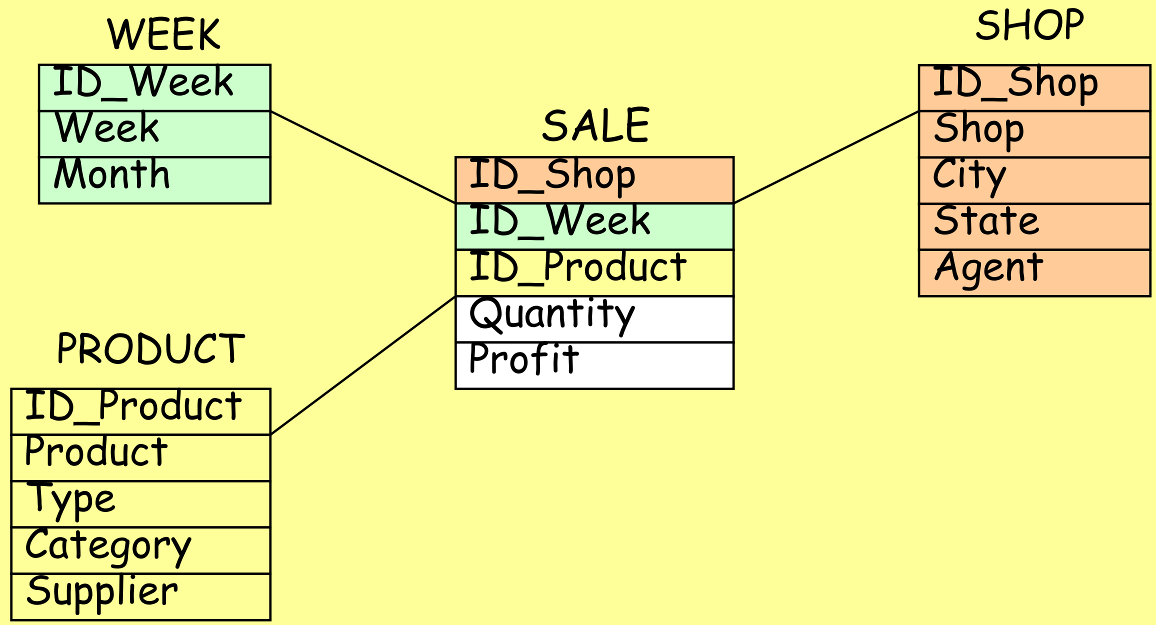


Star schema: considerations

- Dimension table keys are <u>surrogates</u>, for space efficiency reasons
- Dimension tables are de-normalized

  product → type→ category
  is a transitive dependency
- De-normalization introduces redundancy, but fewer joins to do
- The fact table contains information expressed at different aggregation levels

OLAP queries on Star Schema




```sql
select City, Week, Type, sum(Quantity)
from   Week, Shop, Product, Sale
where  Week.ID_Week=Sale.ID_Week and
       Shop.ID_Shop=Sale.ID_Shop and
       Product.ID_Product=Sale.ID_Product and
       Product.Category = 'FoodStuff'
group by City,Week,Type
```

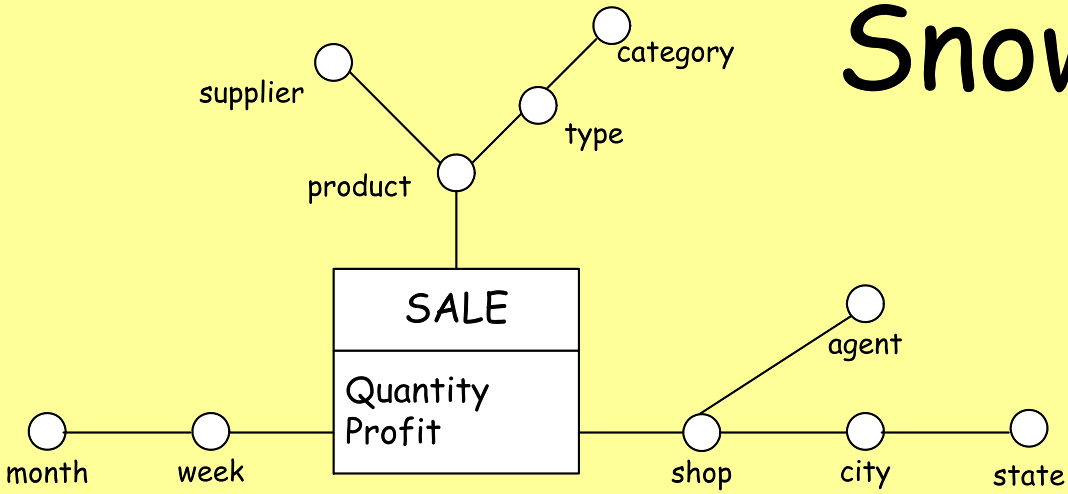
Snowflake schema

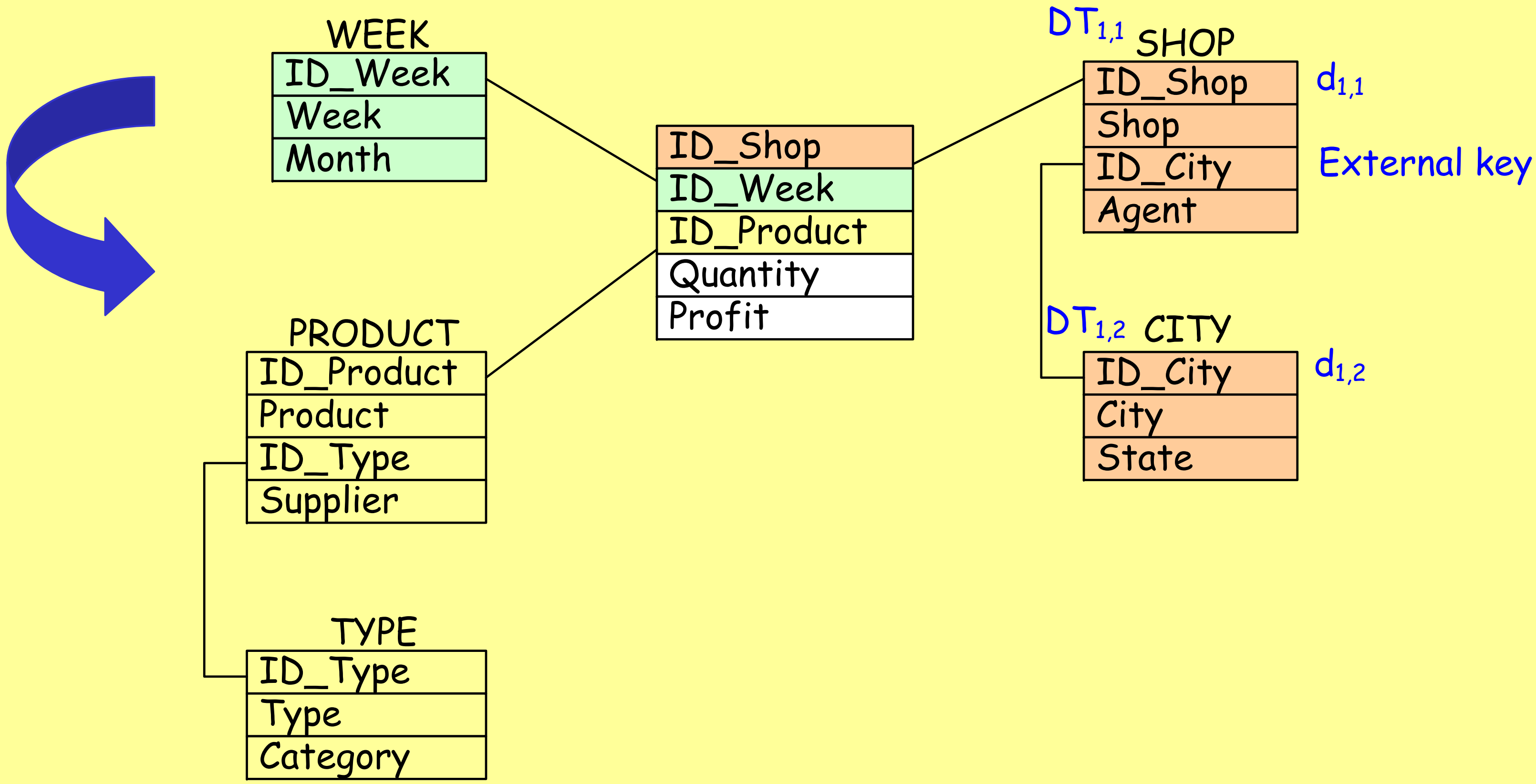
- The snowflake schema reduces the de-normalization of the dimensional tables DT_i of a star schema
  - ✓Removal of some transitive dependencies
- Dimensions tables of a snowflake schema are composed by
  - ✓A primary key $d_{i,j}$
  - ✓A subset of DT_i attributes that directly depends by $d_{i,j}$
  - ✓Zero or more external keys that allow to obtain the entire information

Snowflake schema

- In a snowflake schema
  - ✓Primary dimension tables: their keys are imported in the fact table
  - ✓Secondary dimension tables

Snowflake schema





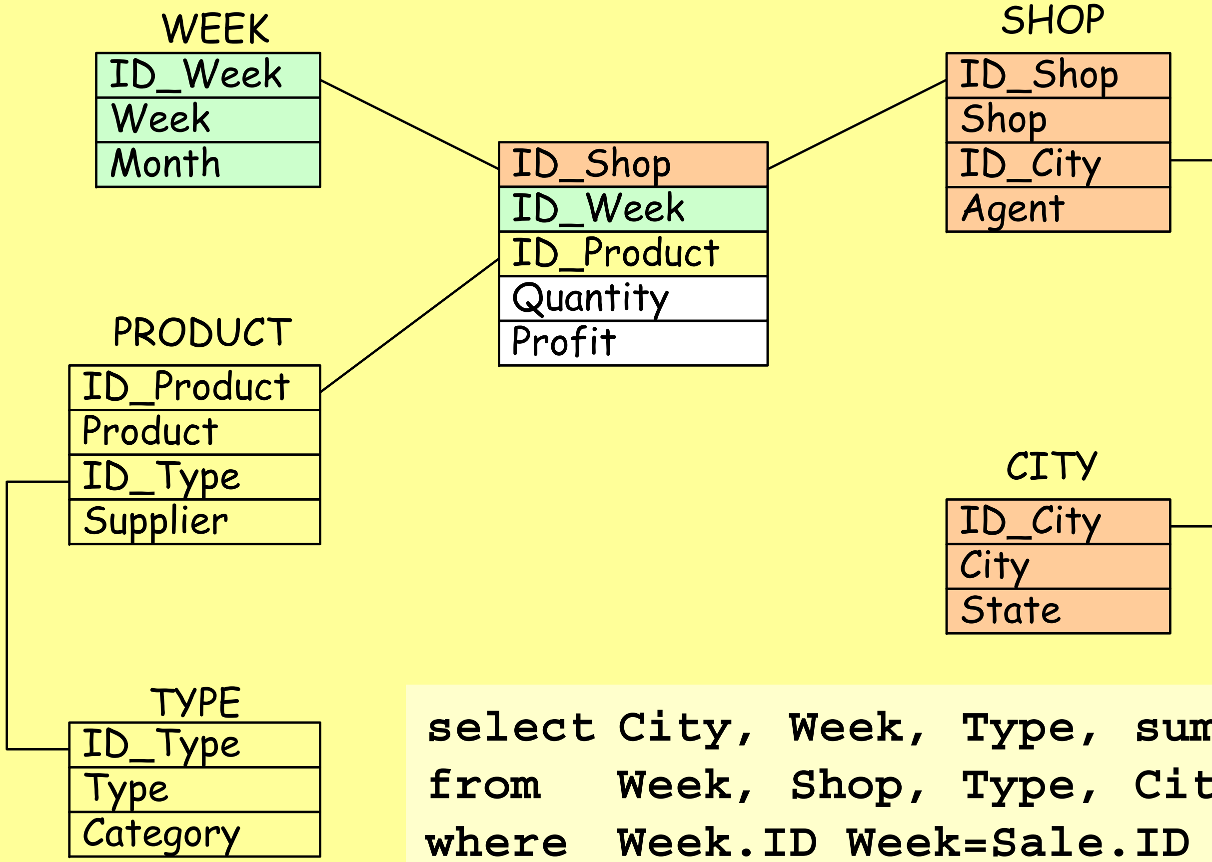
Snowflake schema: considerations

- Reduction of memory space
- New surrogate keys
- Advantages in the execution of queries related to attributes contained into fact and primary dimension tables

Normalization & Snowflake schema

- If there exists a cascade of transitive dependencies, attributes depending (transitively or not) on the snowflake attribute are placed in a new relation

OLAP queries on snowflake schema




```
select City, Week, Type, sum(Quantity)
from   Week, Shop, Type, City, Product, Sale
where  Week.ID_Week=Sale.ID_Week and
       Shop.ID_Shop=Sale.ID_Shop and
       Shop.ID_City=City.ID_City and
       Product.ID_Product=Sale.ID_Product and
       Product.ID_Type=Type.ID_Type and
       Product.Category = 'FoodStufs'
group by City,Week, Type
```

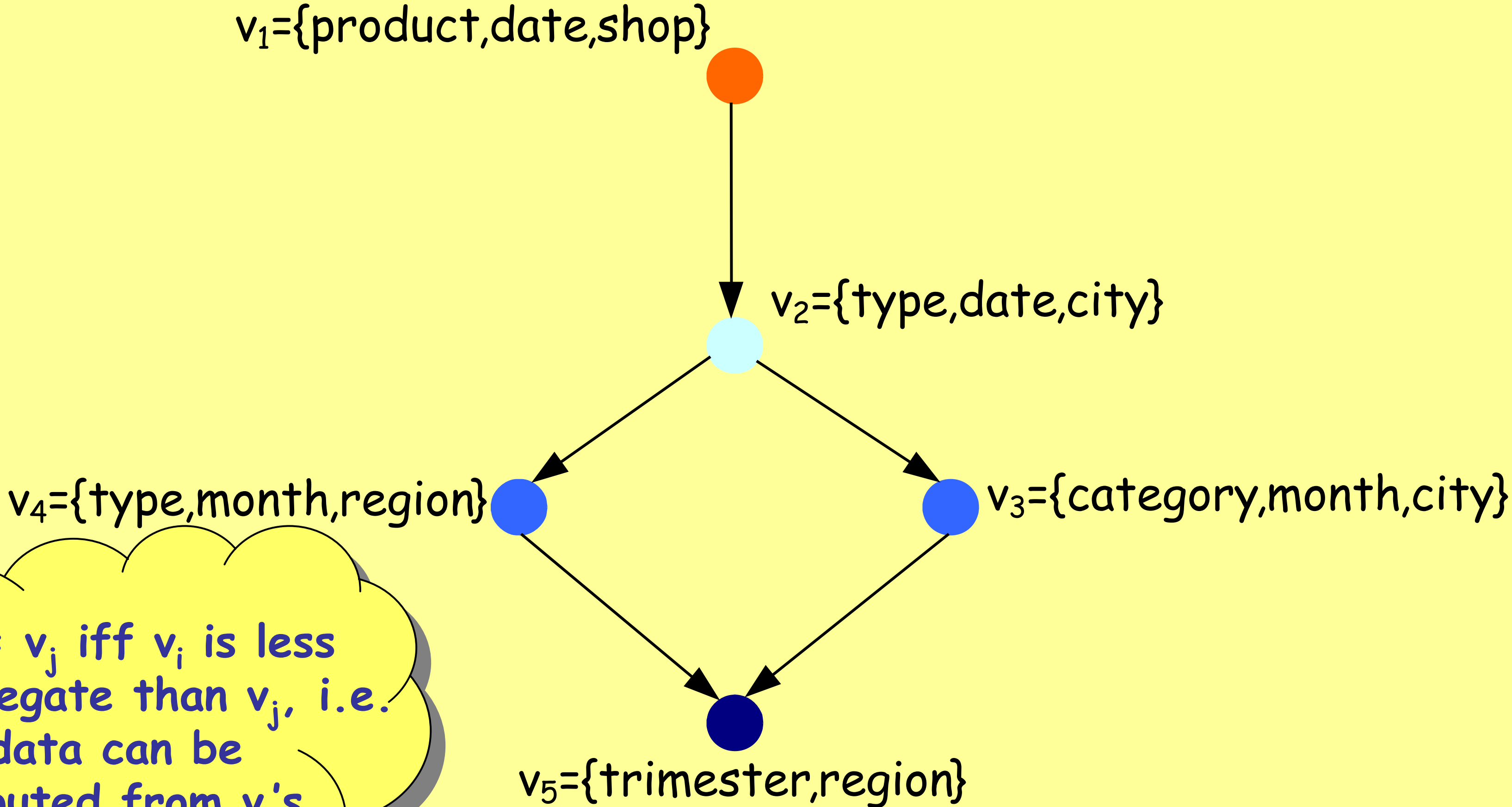
Views

- Aggregation allows to consider concise (summarized) information
- Aggregation computation is very expensive→ pre-computation
- A view denotes a fact table containing aggregate data

Views

- A view can be characterized by its aggregation level (pattern)
  - Primary views: correspond to the primary aggregation levels
  - Secondary views: correspond to secondary aggregation levels (secondary events)

Views
(MultiDimensional Lattice)



$v_i <= v_j$ iff v_i is less aggregate than v_j, i.e. v_j's data can be computed from v_i's data

Partial aggregations

- Sometimes it is useful to introduce new measures in order to manage aggregations correctly
  - Derived measures: obtained by applying mathematical operators to two or more values of the same tuple

Partial aggregations

Profit=Quantity*Price

Type	Product	Quantity	Price	Profit
T1	P1	5	1,00	5,00
T1	P2	7	1,50	10,50
T2	P3	9	0,80	7,20

22,70
(total profits)

SUM AVG

Type	Quantity	Price	Profit
T1	12	1,25	15,00
T2	9	0,80	7,20

22,20

We can't just sum up profits as before!!

The correct solution consists in the aggregation of data on the primary table

Aggregate operators

- Distributive operator: allows to aggregate data starting from partially aggregated data (e.g. sum, max, min)
- Algebraic operator: requires further information to aggregate data (e.g. avg)
- Holistic operator: it is not possible to obtain aggregate data starting from partially aggregate data (e.g. mode, median)

Aggregate operators

- Currently, aggregate navigators are included in the commercial DW system
- They allow to re-formulate OLAP queries on the "best" view
- They manage aggregates only by means of distributive operators

Relational schema and aggregate data

- It is possible to define different variants of the star schema in order to manage aggregate data
- First solution: data of primary and secondary views are stored in the same fact table
  - NULL values for attributes having aggregation levels finer than the current one

Aggregate data in a unique fact table

1° row represents sale values for the single shop, 2° row represents aggregate values for Roma, 3° row represents aggregate values for Lazio, etc…

SALE

Shop_key	Date_key	Prod_key	qty	profit	…
1	1	1	170	85	…
2	1	1	300	150	…
3	1	1	1700	850	…
…	…	…	…	…	…

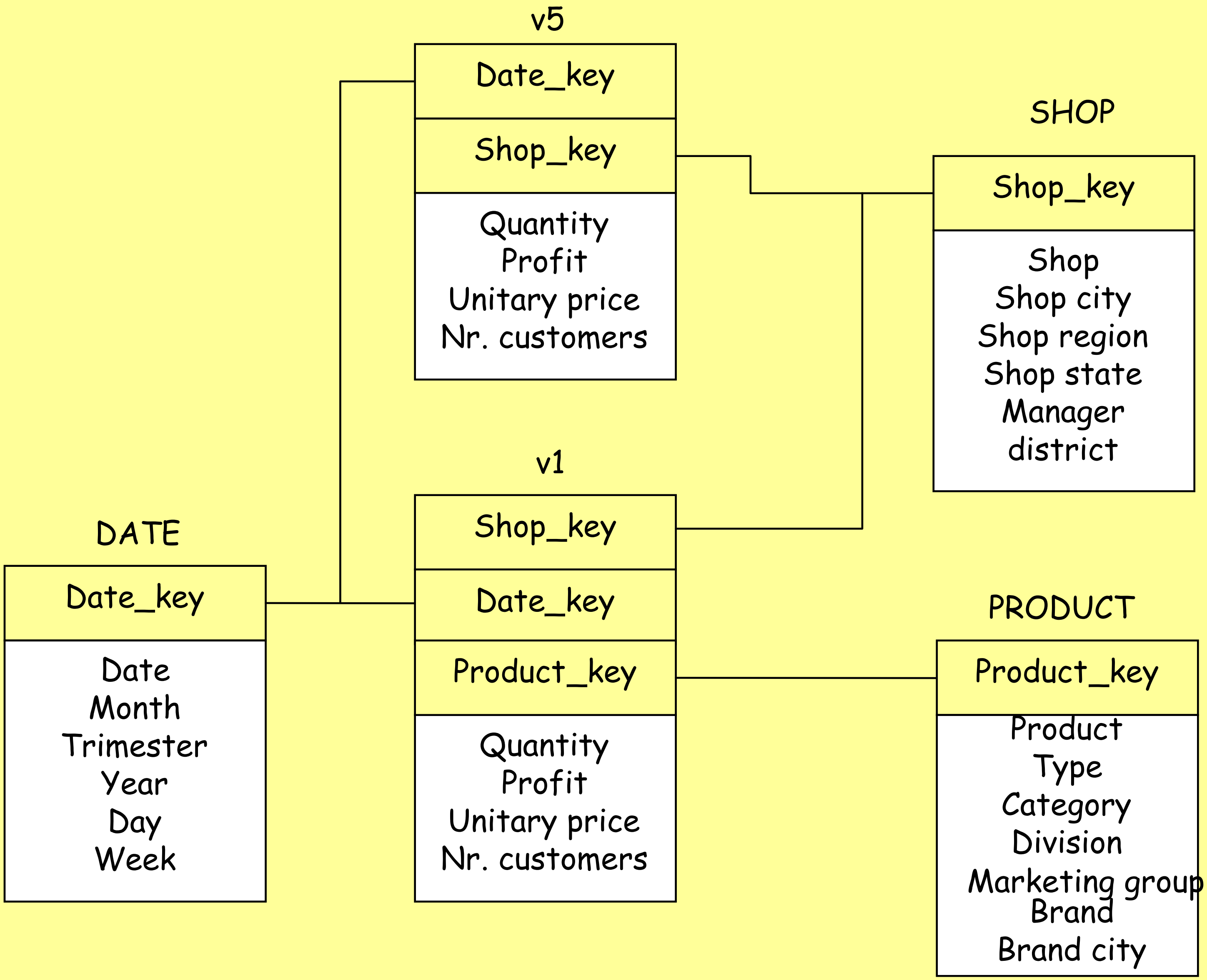
SHOP

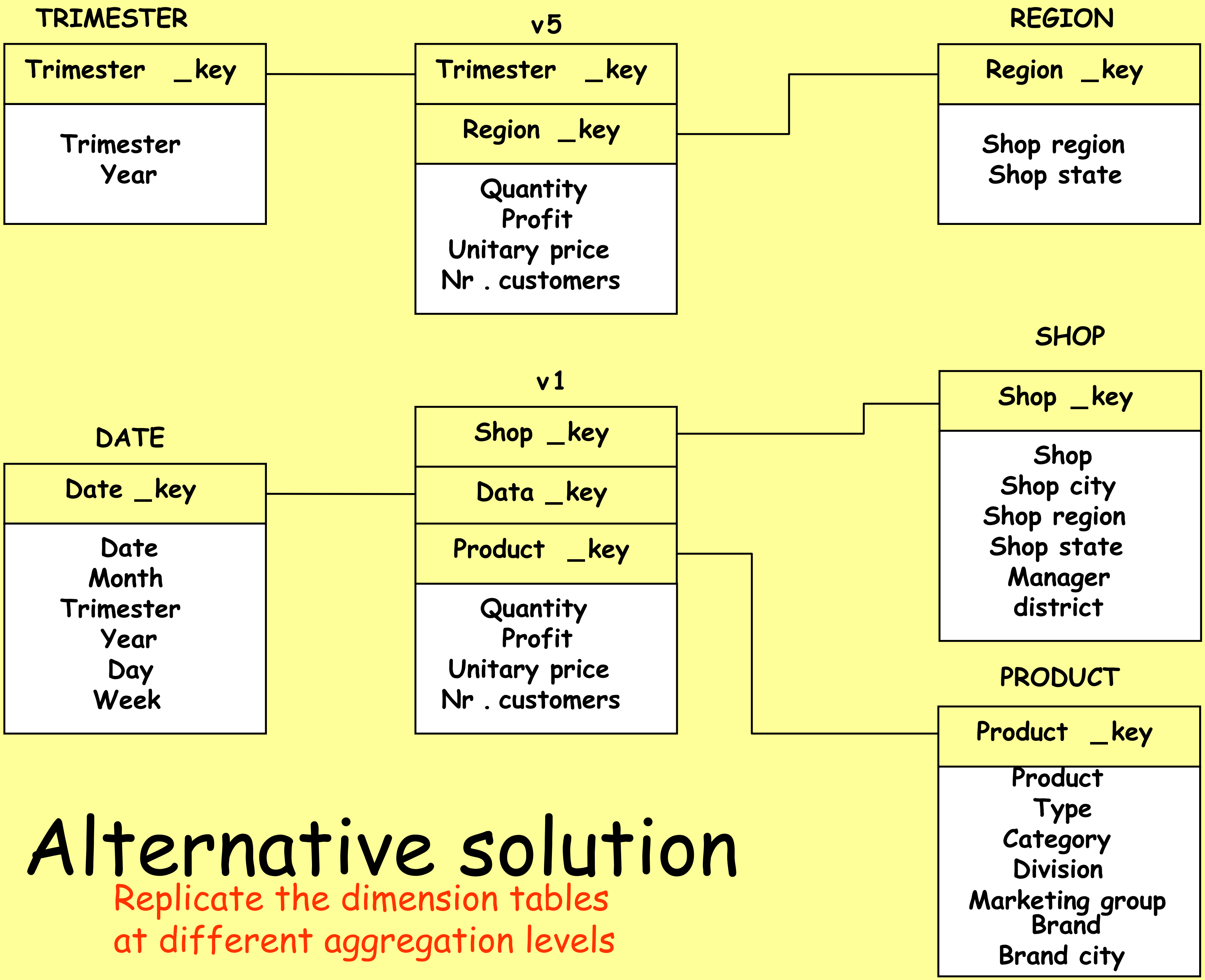
Shop_key	shop	city	region	…
1	COOP1	Bologna	E.R.	…
2	-	Roma	Lazio	…
3	-	-	Lazio	…
…	…	…	…	…

Relational schema and aggregate data

- Second solution: distinct aggregation patterns are stored in distinct fact tables: constellation schema
- Only the dimension of the fact table is optimized, but this is a great improvement already
- Max optimization level: separate fact tables, and also repeated dimension tables for different aggregation levels

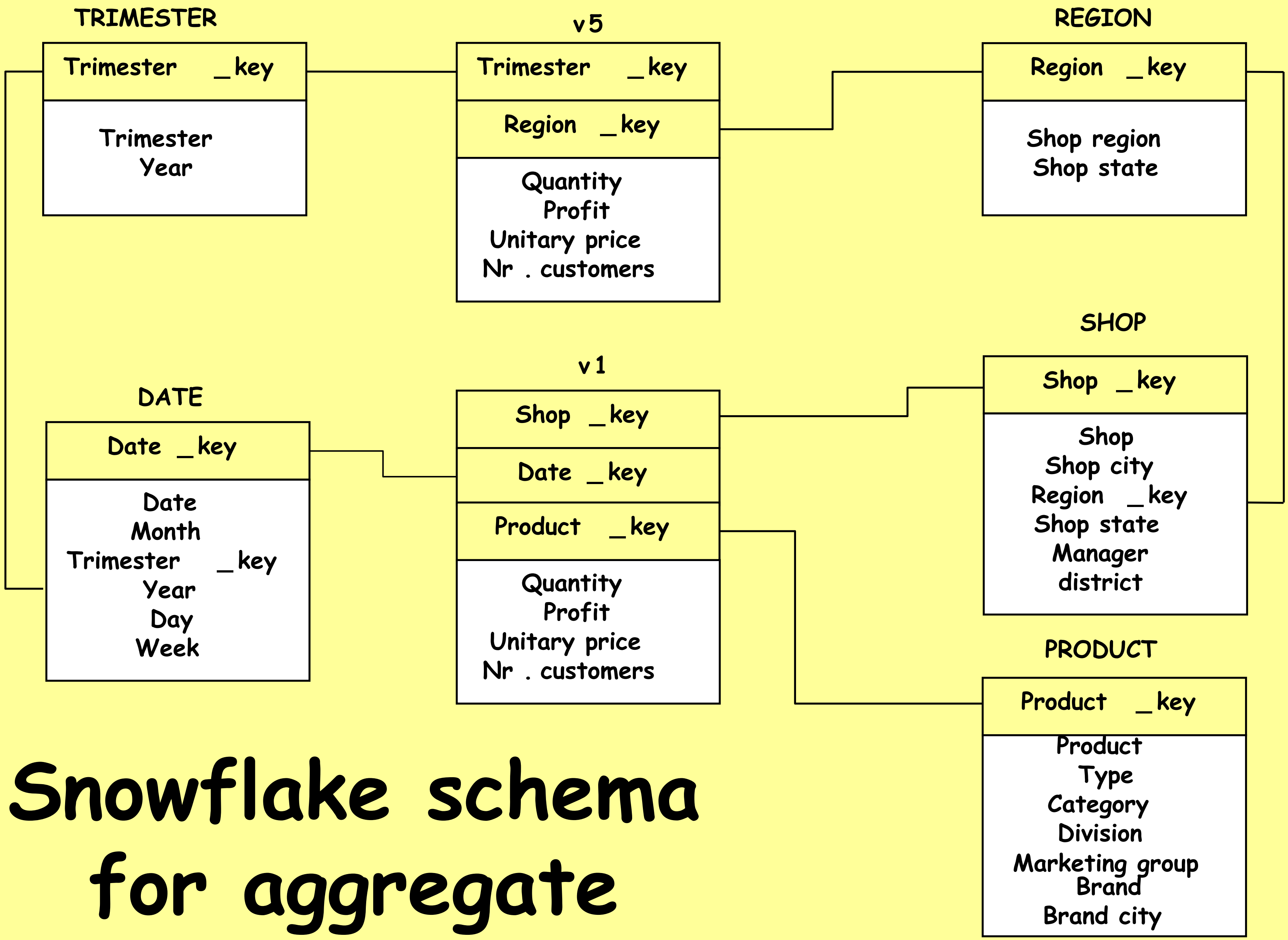
Constellation schema





Alternative solution

Replicate the dimension tables
at different aggregation levels



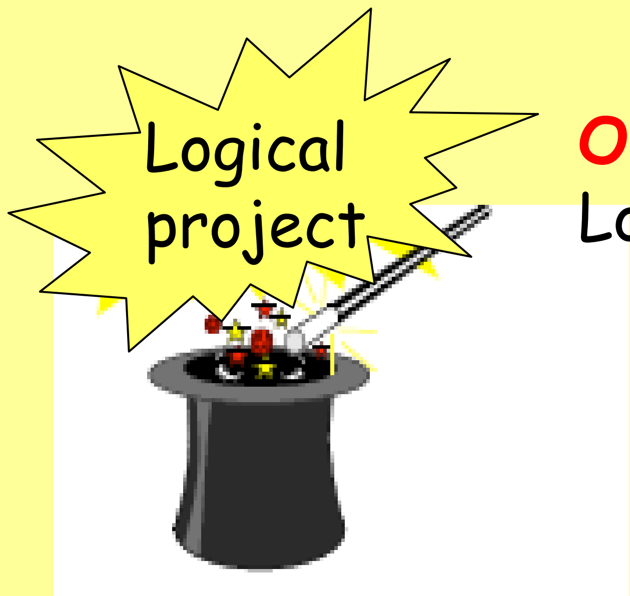
Snowflake schema for aggregate data

Logical design

Logical modelling

- Sequence of steps that, starting from the conceptual schema, allow one to obtain the logical schema for a specific data mart

INPUT
Conceptual Schema
WorkLoad
Data Volume
System constraints



OUTPUT
Logical Schema

Worklad

- In OLAP systems, workload is dynamic in nature and intrinsically extemporaneous
  - Users' interests change during time
  - Number of queries grows when users gain confidence in the system
  - OLAP should be able to answer any (unexpected) request
- During requirement collection phase, deduce it from:
  - Interviews with users
  - Standard reports

Worklad

- Characterize OLAP operations:
  - Based on the required aggregation pattern
  - Based on the required measures
  - Based on the selection clauses
- At system run-time, workload can be desumed from the system log

Data volume

- Depends on:
  - Number of distinct values for each attribute
  - Attribute size
  - Number of events (primary and secondary) for each fact
- Determines:
  - Table dimension
  - Index dimension
  - Access time

Logical modelling: steps

1. Choice of the logical schema (star/snowflake schema)
2. Conceptual schema translation
3. Choice of the materialized views
4. Optimization

From fact schema to star schema

- Create a fact table containing measures and descriptive attributes directly connected to the fact
- For each hierarchy, create a dimension table containing all the attributes

Guidelines

- Descriptive attributes (e.g. color)
  - If it is connected to a dimensional attribute, it has to be included in the dimension table containing the attribute (see slide n. 13, snowflake example, agent)
  - If it is connected to a fact, it has to be directly included in the fact schema
- Optional attributes (e.g. diet)
  - Introduction of null values or ad-hoc values

Guidelines

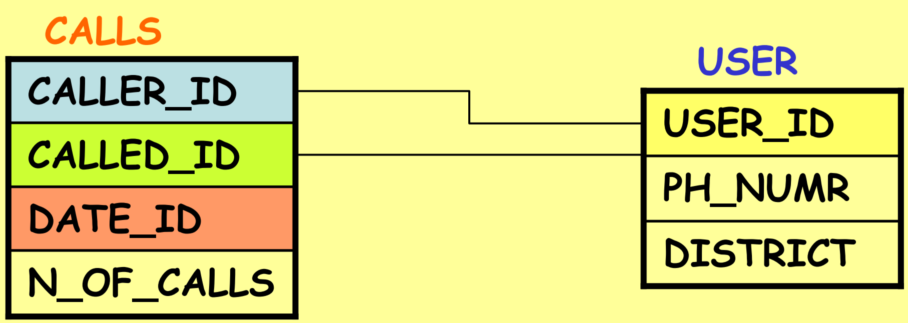
- Cross-dimensional attributes (e.g. VAT)
  - A cross-dimensional attribute b defines an N:M association between two or more dimensional attributes $a_1, a_2, \ldots, a_k$
  - It requires to create a new table including b and having as key the attributes $a_1, a_2, \ldots, a_k$

Guidelines

- Shared hierarchies and convergence
  - A shared hierarchy is a hierarchy which refers to different elements of the fact table (e.g. caller number, called number)
  - The dimension table should not be duplicated
  - Two different situations:
    - The two hierarchies contain the same attributes, but with different meanings (e.g. phone call → caller number, phone call → called number)
    - The two hierarchies contain the same attributes only for part of the hierarchy trees

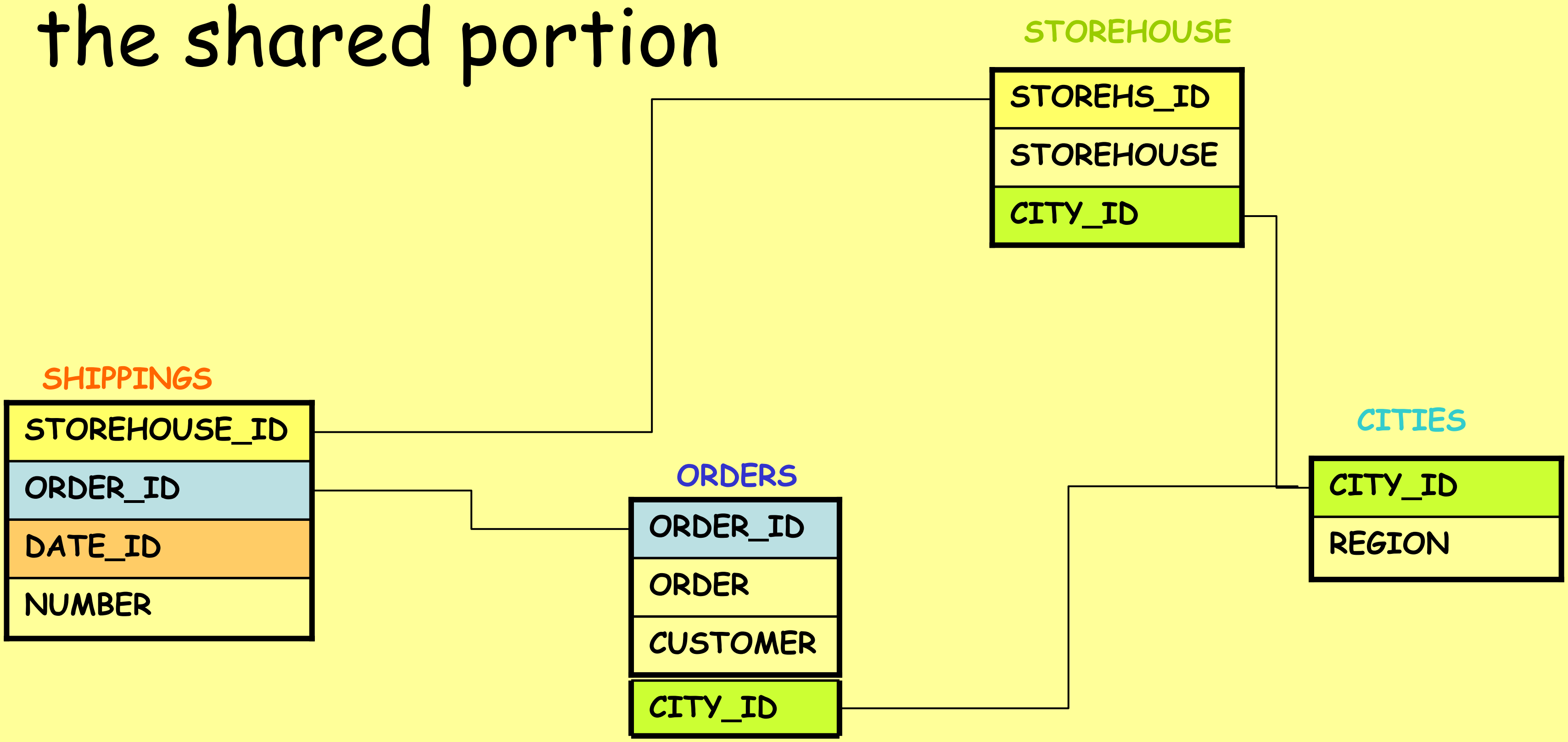
Shared hierarchies and convergence

- The two hierarchies contain the same attributes, but with different meanings (e.g. phone call → caller number, phone call → called number)



Shared hierarchies and convergence

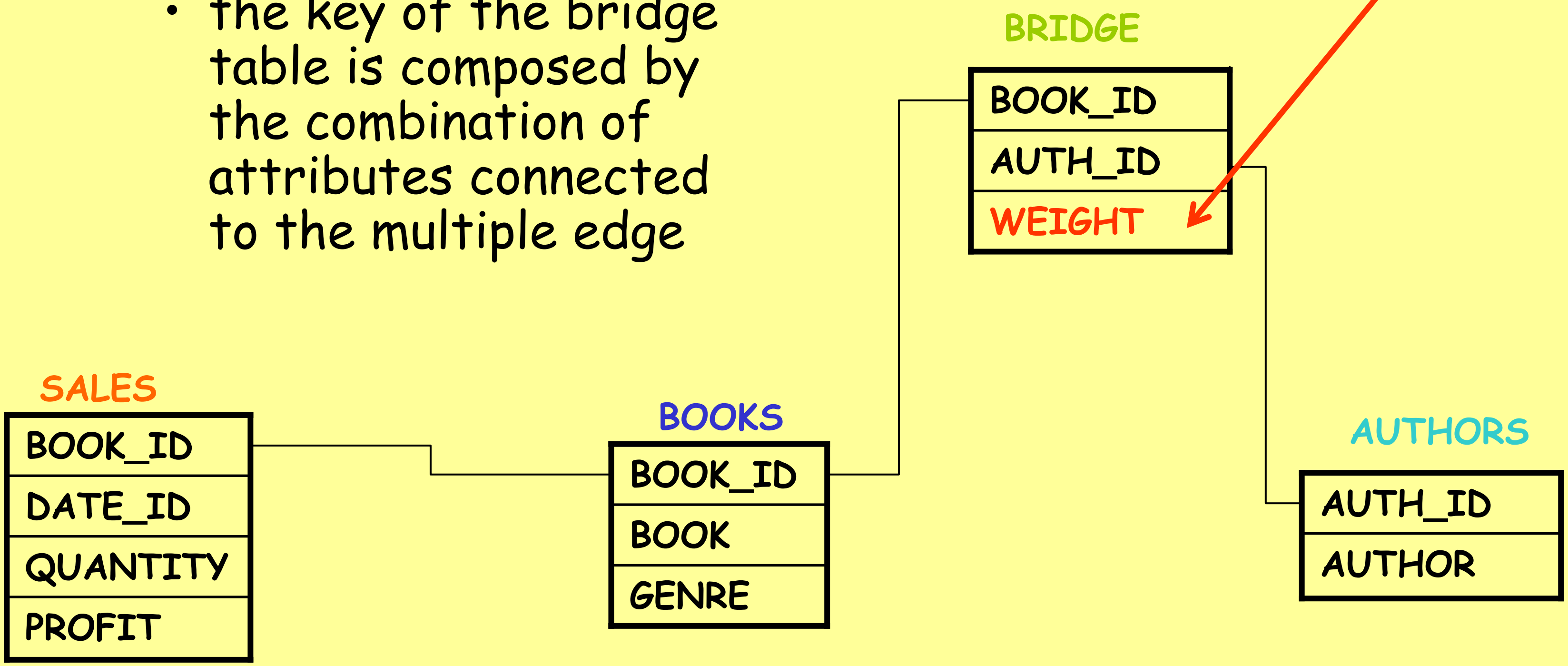
- The two hierarchies contain the same attributes only for part of the trees. Here we could also decide to replicate the shared portion



Guidelines

- Multiple edges
  - A bridge table models the multiple edge
    - the key of the bridge table is composed by the combination of attributes connected to the multiple edge

The weight of the edge is the contribution of each edge to the cumulative relationship



Guidelines

- Multiple edges: bridge table
  - Weighed queries take into account the weight of the edge

Profit for each author

```
SELECT AUTHORS.Author,SUM(SALES.Profit * BRIDGE.Weight)
FROM AUTHORS, BRIDGE, BOOKS, SALES
WHERE AUTHORS.Author_id=BRIDGE.Author_id
AND BRIDGE.Book_id=BOOKS.Book_id
AND BOOKS.Book_id=SALES.Book_id
GROUP BY AUTHORS.Author
```

Guidelines

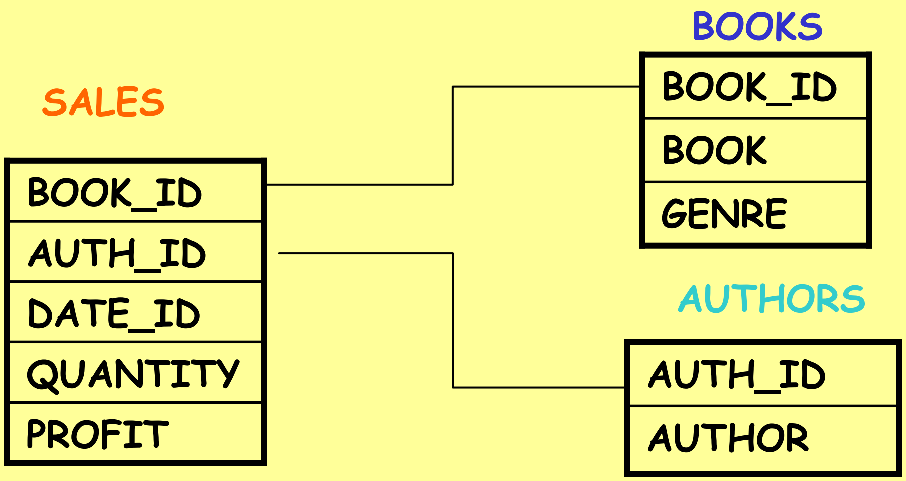
- Multiple edges: bridge table
  - Impact queries do not take into account the weight of the edge

Sold copies for each author

```
SELECT AUTHORS.Author, SUM(SALES.Quantity)
FROM AUTHORS, BRIDGE, BOOKS, SALES
WHERE AUTHORS.Author_id=BRIDGE.Author_id
AND BRIDGE.Book_id=BOOKS.Book_id
AND BOOKS.Book_id=SALES.Book_id
GROUP BY AUTHORS.Author
```

If we want to keep the star model

Multiple edges with a star schema: add authors to the fact schema

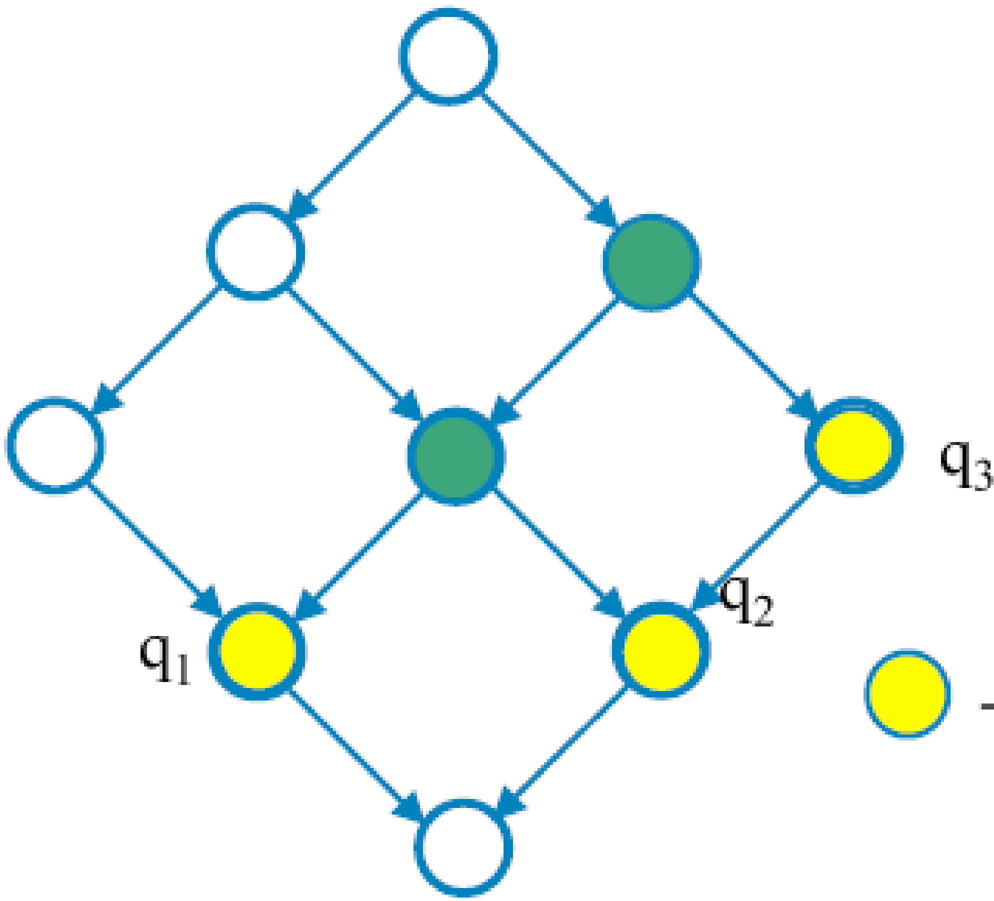


Here we don't need the weight because the fact table records quantity and profit per book and per author

Secondary-view precomputation

- The choice about views that have to be materialized takes into account contrasting requirements:
  - Cost functions minimization
    - Workload cost
    - View maintenance cost
  - System constraints
    - Disk space
    - Time for data update
  - Users constraints
    - Max answer time
    - Data freshness

Materialized views (MD lattice)

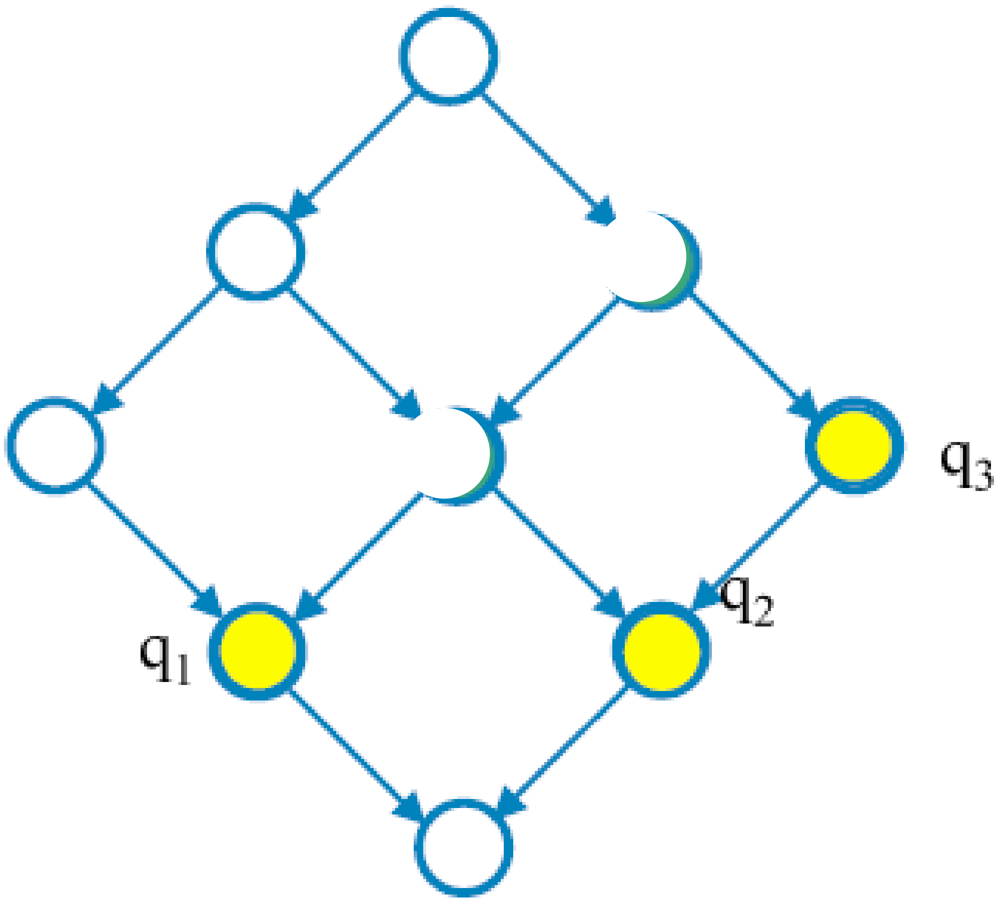


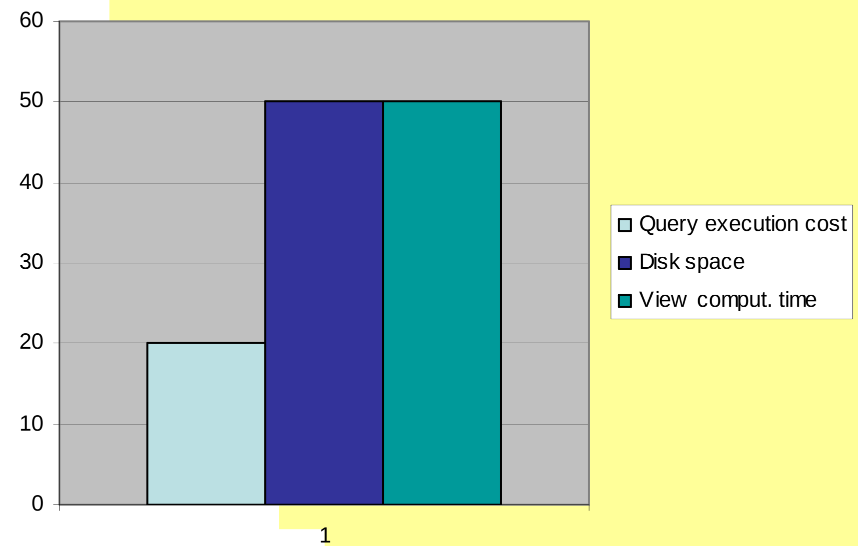
= **exact views:**
They solve exactly the queries

= **less aggregate views:**
They solve more than one query

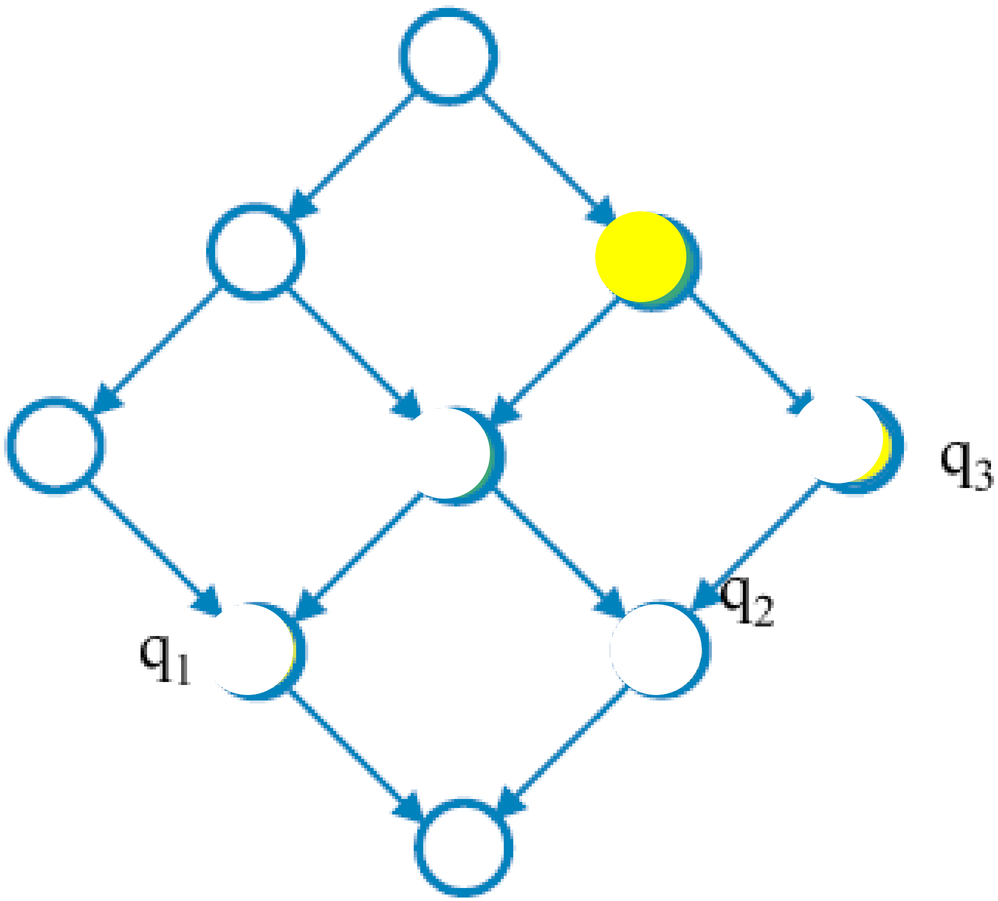
+ = **candidate views**
They could reduce elaboration costs

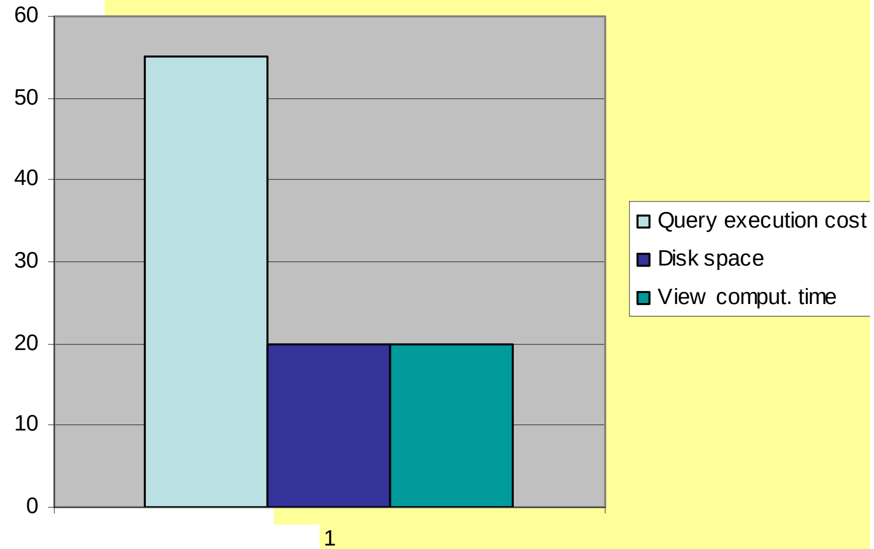
Materialized views (MD lattice)



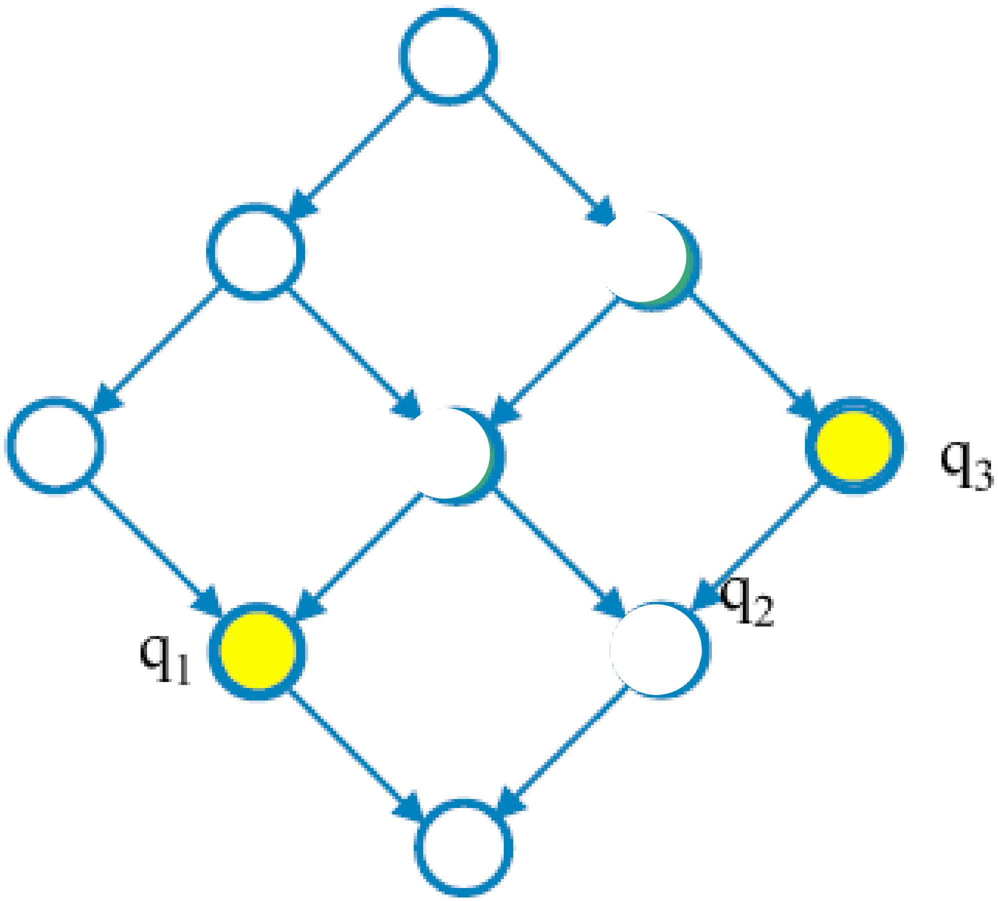


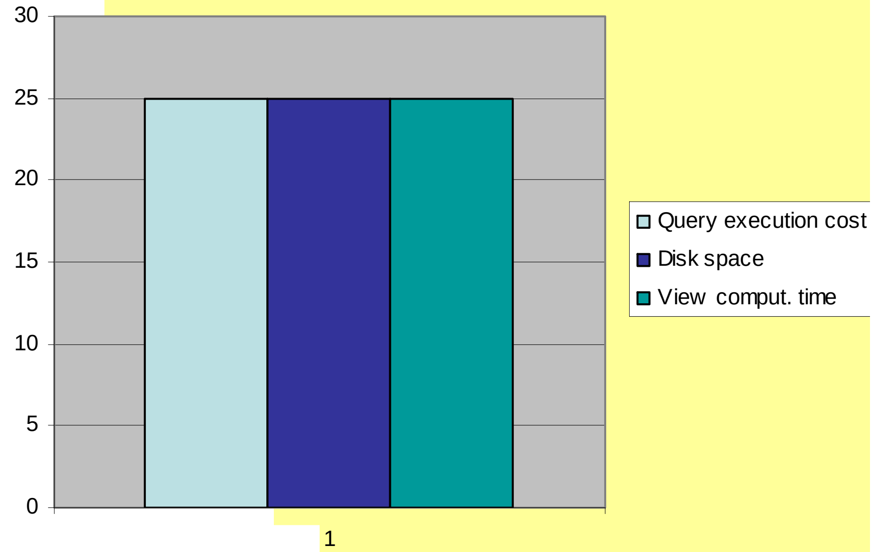
Materialized views (MD lattice)





Materialized views (MD lattice)





Materialized Views

- It is useful to materialize a view when:
  - It directly solves a frequent query
  - It reduce the costs of some queries
- It is not useful to materialize a view when:
  - Its aggregation pattern is the same as another materialized view
  - Its materialization does not reduce the cost

References

- <u>M. Golfarelli, S. Rizzi:</u> Data Warehouse: teoria e pratica della progettazione McGraw-Hill, 2002.
- <u>Ralph Kimball:</u> The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses John Wiley 1996.