UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Data WareHouse – Logical modeling

Prof. A. Peron

Slides from M. Golfarelli, S. Rizzi, Datawarehouse Design, Modern Principles and methodologies, McGrawHill.

(Slightly modified by Dario Della Monica)

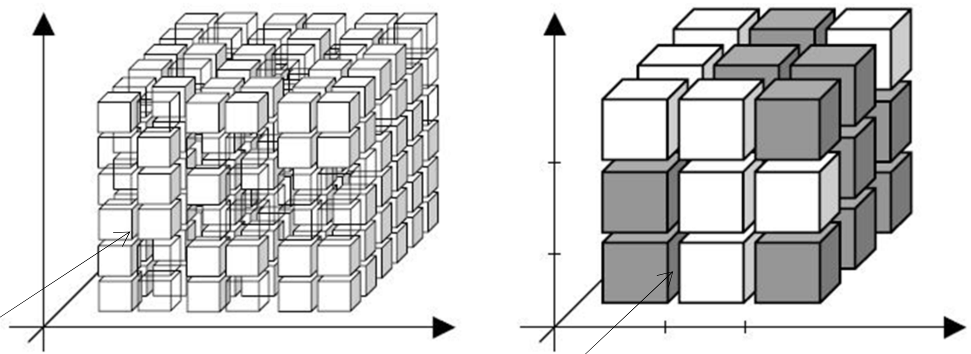
MOLAP

Multidimensional On-Line Analytical Processing

- They store data using a multidimensional model (e.g. multidimensional vectors); each element of the vector is associated with a set of coordinates in the space of values.
- **Criticism: Sparsity of data**
- In a multidimensional DBMS all the cells of the cube should be represented;
- Usually less than 20% of the cells in a cube has data.
- Criticism: lack of widely accepted standards
- The systems have common basics, such as multidimensional data structures and handling of sparsity
- The implementations are often based on poorly documented proprietary data structures
- The systems are hard to be replaced and accessed by third-party tools.
- No query standard playing the same role of SQL standard query language.

MOLAP: dealing with sparsity

- **Partition cubes: A** dimensional cube is split into various sub-cubes called **chunks.**
- This strategy leads to small sized blocks of data that can be quickly loaded into memory;
- Cube partitioning can also manage a sparse chunks and dense chunks in different ways
- A chunk is dense if most of its cells include data otherwise a chunk is sparse



Data

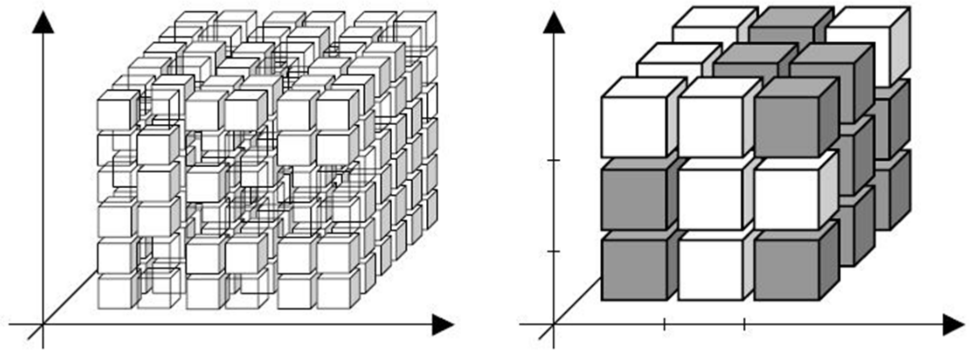
Dense Chunk

MOLAP: sparsity

Handling sparsity

▶ **Compress chunk:** starting a sparse chunk directly to memory implies a waste of a free space because of the representation of the cells with no information

▶ An index that lists only the chunk cells containing information is normally used to create a compressed representation of sparse chunks



HOLAP: Hybrid OLAP

- **In HOLAP systems a crucial design factor is the definition of the policies to apply to select which data should be stored in ROLAP mode and which data should be stored in MOLAP mode.**
- **Possible strategies:**
- Store dense chunks in MOLAP mode and the sparse chunks in ROLAP mode
- Store primary cubes in ROLAP mode and secondary cubes in MOLAP mode
- Store frequentely accessed data in MOLAP mode and remaining data in ROLAP mode

ROLAP

Relational On-Line Analytical Processing

- ROLAP systems adopt the well known relational model to represent multidimensional data
- It uses a model based on a bidimensional element: relations have rows and columns for modeling multidimensional data.
- Pro
    - The relational model is the standard de facto for DBMS.
    - It has no problems of sparsity
    - ROLAP systems are more scalable than MOLAP systems

ROLAP

- The multidimensional modeling on relational DBMS is based on the idea of STAR SCHEMA **and its variants.**

- **Star schema**: The star schema consists of

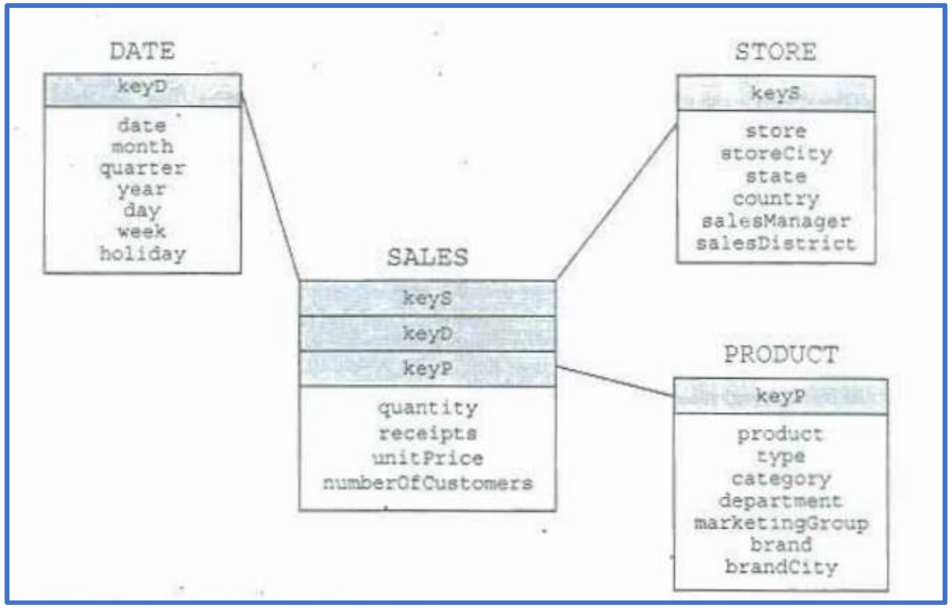
- A set of relations $\boldsymbol{DT_1}$**…,** $\boldsymbol{DT_n}$, named **dimension tables**, in one-to-one correspondence with the dimensions.

- Any relation $\boldsymbol{DT_i}$ has primary key **(usually surrogate)** k_i and a set of attributes describing the dimension at different aggregation level.

- A relation **FT**, called **fact table**, that
  - Includes the primary keys of all the dimension tables k_1 , …, k_n (k_1 , …, k_n is the primary key of FT)
  - If features an attribute per each measure.

Star Schema

Star Schema for sales



Multidimensional view

```
SELECT *
FROM   SALES AS FT, PRODUCT AS DT1,
       STORE AS DT2, DATA AS DT3
WHERE  FT.keyP = DT1.keyP
AND    FT.keyS = DT2.keyS
AND    FT.keyD = DT3.keyD
```

Star schema

- We can relate many fact table to the same dimension tables (dimensions shared by cubes)
- The dimension tables are not normalized (due to the functional dependencies in the hierarchy of dimensional attributes).
- **Redundancy**: The redundancy due to denormalization is not a problem (problems for insertion, deletion and update) since the dimensions are typically static.
- The size of dimension tables is typically far lower than the fact table (waste of space due to redundancy is negligible)
- **Sparsity:** it is not an issue since the fact table only records occurrences of a fact (there is no placeholder for cells with empty information).

Use of surrogate keys

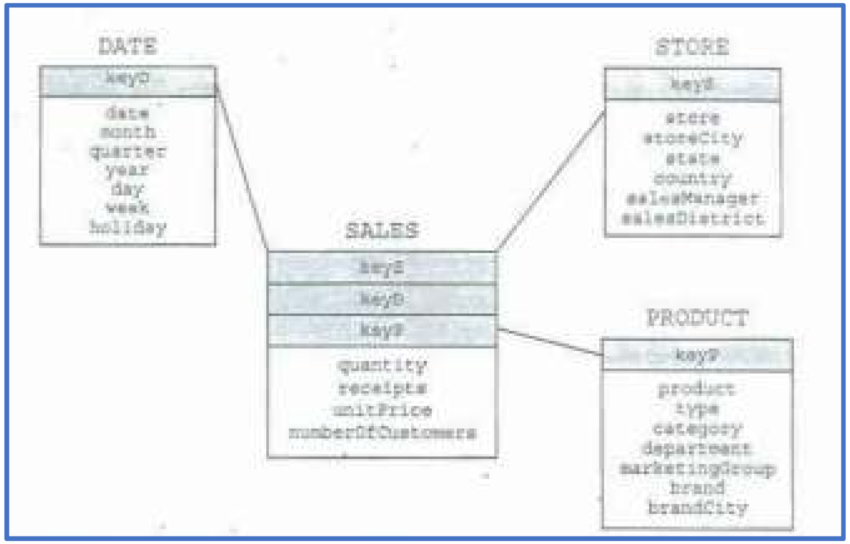
- It is suggested to use surrogate keys in the dimension tables:
- **Advantages**:
- They are usually more compact than semantic keys and reduce the size for the foreign keys in the fact table.
- They provide a quicker access to data because the query execution plans can use a simple index based on a single numeric attribute
- They offer independence of any changes of identifier values applied to operational sources
- They are able to represent many versions of an individual hierarchy in the case of dynamic hierarchies
- **DisAdvantages**:
- In the population phase they force you to transcode the natural keys included in the source schema.
- It causes and increase in size of dimension tables if the natural keys are also included in dimension tables.

Snowflake Schema

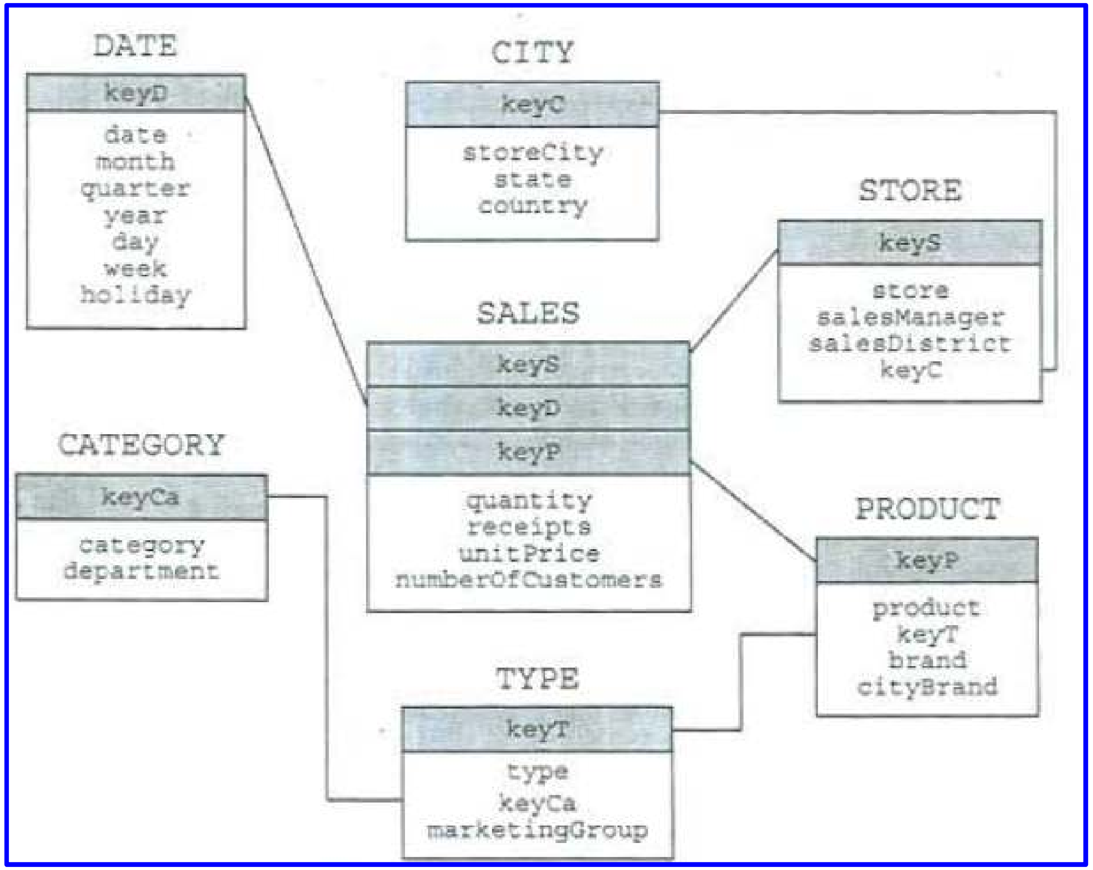
- Introduces partial normalization in dimension tables.
- **snowflake schema: It can be obtained from star schema by decomposing one or more dimension table $\boldsymbol{DT_i}$ into various smaller tables $DT_{i,1}$…$DT_{i,m}$ to remove some or all transitive functional dependencies.**
  - Every dimension table consists of the following:
  - One primary key (typically surrogate) $d_{i,j}$;
  - A subset of $\boldsymbol{DT_i}$ attributes functionally depending on $d_{i,j}$;
  - Sone foreign keys, each referencing another $\boldsymbol{DT_{i,k}}$ table necessary for any $\boldsymbol{DT_i}$ to be properly reconstructed.
- Dimension tables whose keys are referenced in the fact tables are called primary.
- The remaining tables are called secondary dimension tables.

Snowflake

Star schema for sales



Derived Snowflake.



Snowflake schema

- A snowflake is obtained by progressively deleting some transitive functional dependency from the dimension table.
- Each normalization step is related to an arc in the DFM and marks a sub-hierachy that should be stored separately.
- **Consequences (+/-):**
- (+) The disk space required for data storage decreases due to the removal of duplicated data.
- (-) It is necessary to add new surrogate keys in order to express the relationship between primary and secondary dimension tables.
- (+) Processing the queries that involve only fact table attributes and the primary dimension table attributes is less costly because their joints involve smaller tables.
- (-) The time needed for queries of secondary dimension table attributes is longer because of a larger number of necessary joins

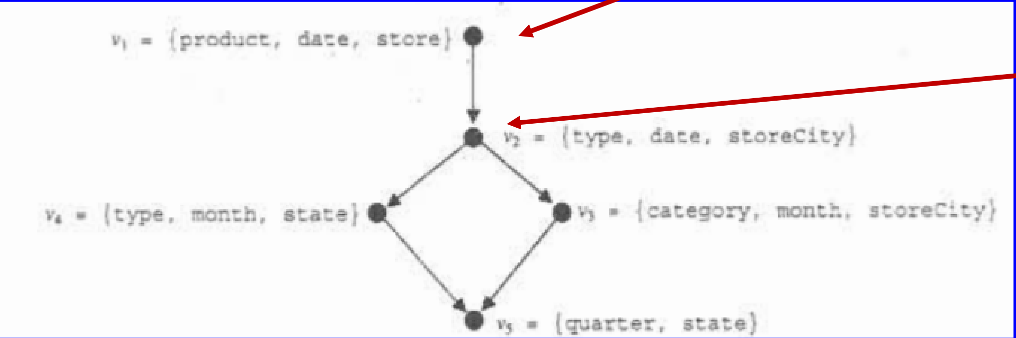
Materialized views.

- The huge amount of data stored in a data warehouse makes users analysis difficult.
- Users tend to apply selection and aggregation to decrease the parts of data they examine
- If one calculates in advance the most frequently used aggregate data this can result in a significant increase in performance
- **Views: The fact table containing aggregate data are called views.** (identified by their aggregation pattern).
- A view can be identified by its group-by set.
- **Primary views**: the fact table defined by the primary events (the most detailed one)
- **Secondary views:** correspond to secondary group-by sets (aggregated).
- A relevant aspect of secondary views is that they can be populated from other views in the datawarehouse (and not directly from operational data).
- If secondary views are populated from other views attention must be payed to additivity of dimensions and distributivity of aggregation operations.

Viste materializzate

Primary view

Secondary views obtained by aggregation



type	product	quantity	unitPrice	receipts
Dairy product	Slurp Milk	5	1.0	5.0
Dairy product	Fresh Milk	7	1.5	10.5
Drinks	Coky	9	0.8	7.2
			Total:	22.7

SUM AVG

type	quantity	unitPrice	quantity × unitPrice
Dairy product	12	1.25	15.0
Drinks	9	0.8	7.2
		Total:	22.2

FIGURE 8-8 Calculating aggregate values can result in errors if you do not carefully take operator features into account.

Receipts cannot be calculated from the secondary view

Secondary views (2)

Attention have to be paid in the usage of non distributive aggregation operations.

4-MonthPeriod	date	stockLevel
I'09	1/1/2009	100
I'09	2/10/2009	200
I'09	4/31/2009	60
II'09	6/5/2009	85
II'09	7/18/2009	125
III'09	12/31/2009	110
	Average:	113.33

AVG

4-MonthPeriod	stockLevel	count	stockLevel × count
I'09	120	3	360
II'09	105	2	210
III'09	110	1	110
Average:	111.66	*Weighted Avg:*	113.33

FIGURE 8-9 Calculating aggregate values can result in errors if you do not carefully take operator features into account.

Additional information required for the correct computation of AVG (algebraic operator)

Schemata with aggregate data

▶ If materialized views are present you can use different variants of the standard star schema

▶ **Single fact table**: primary view data and secondary view data are stored in the same fact table.

▶ The aggregation level of individual tuples in fact tables can be specified by the corresponding tuples in dimension tables.

▶ The dimension table related to aggregated data will have NULL values in all the attributes whose aggregation level is finer

SALES

keyS	keyD	keyP	quantity	receipts	.....
1	1	1	170	85	.....
2	1	1	300	150	.....
3	1	1	1700	850	.....
.....	.....	.....	.....	.....	.....

STORE

keyS	store	storeCity	state	.....
1	COOP1	Columbus	Ohio	.....
2	–	Austin	Texas	.....
3	–	–	Texas	.....
.....	.....	.....	.....	.....

Schemata with aggregate data

- **Single fact table**
- **(+)** the same fact table can be used to solve all the queries
- **(-)** performance becomes poorer because of the huge size of the one and only fact table.

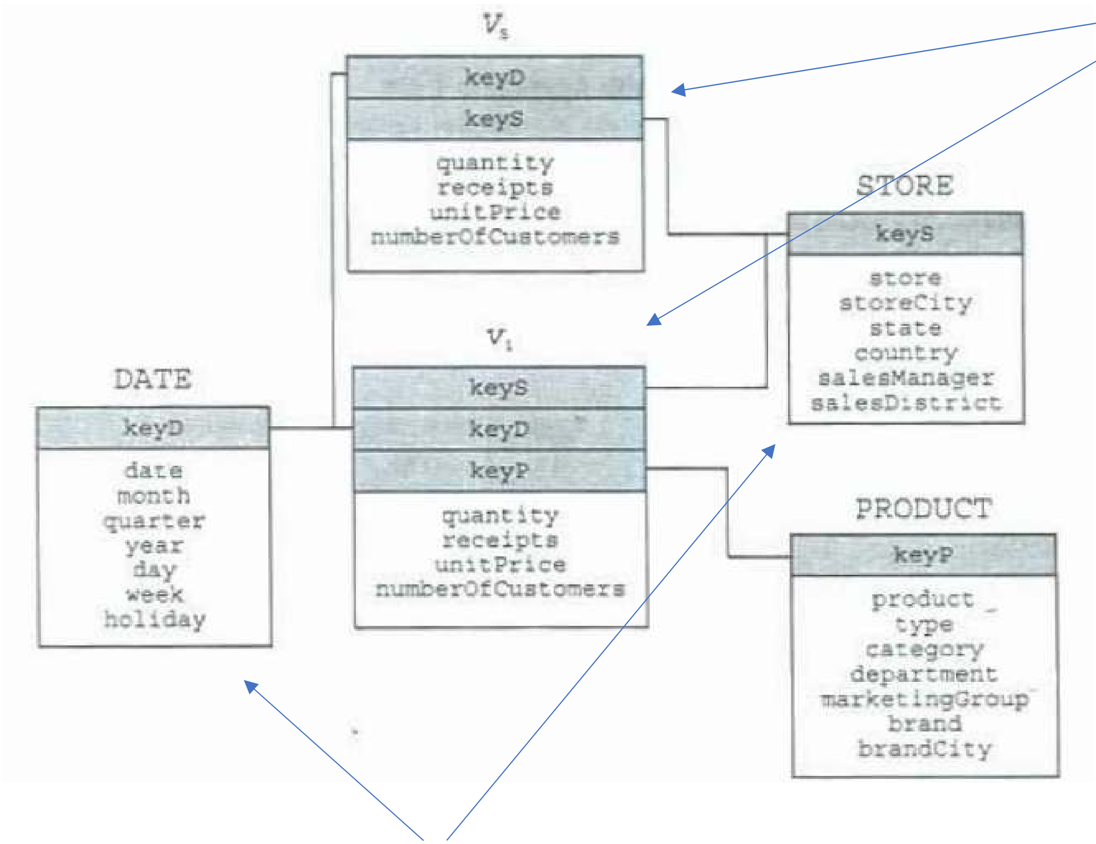
SALES	keyS	keyD	keyP	quantity	receipts	.....
	1	1	1	170	85	.....
	2	1	1	300	150	.....
	3	1	1	1700	850	.....
	.....	.....	.....	.....	.....	.....

STORE	keyS	store	storeCity	state	.....
	1	COOP1	Columbus	Ohio	.....
	2	-	Austin	Texas	.....
	3	-	-	Texas	.....
	.....	.....	.....	.....	.....

Schemata with aggregate data

- storing data related to two different group-by sets into separate factor tables is another option.
- **Having multiple fact tables available requires an additional decision on dimension tables:**
- **costellation schema**: dimension tables are merged and shared by all the fact tables.
- NULL values are required for the attribute not suitable for a given aggregation level (the same solution adopted in the single fact table).
- **the solution optimizes only the access to the fact tables which contains only data at a** particular aggregation level.
- **multiple star schemata**: dimension tables are replicated and customized to the aggregation level of the secondary views.
- Any dimension table includes only the attributes meaningful for the aggregation level for which is used.
- **The solution optimizes both the access to the fact table and the access to the dimension tables.**

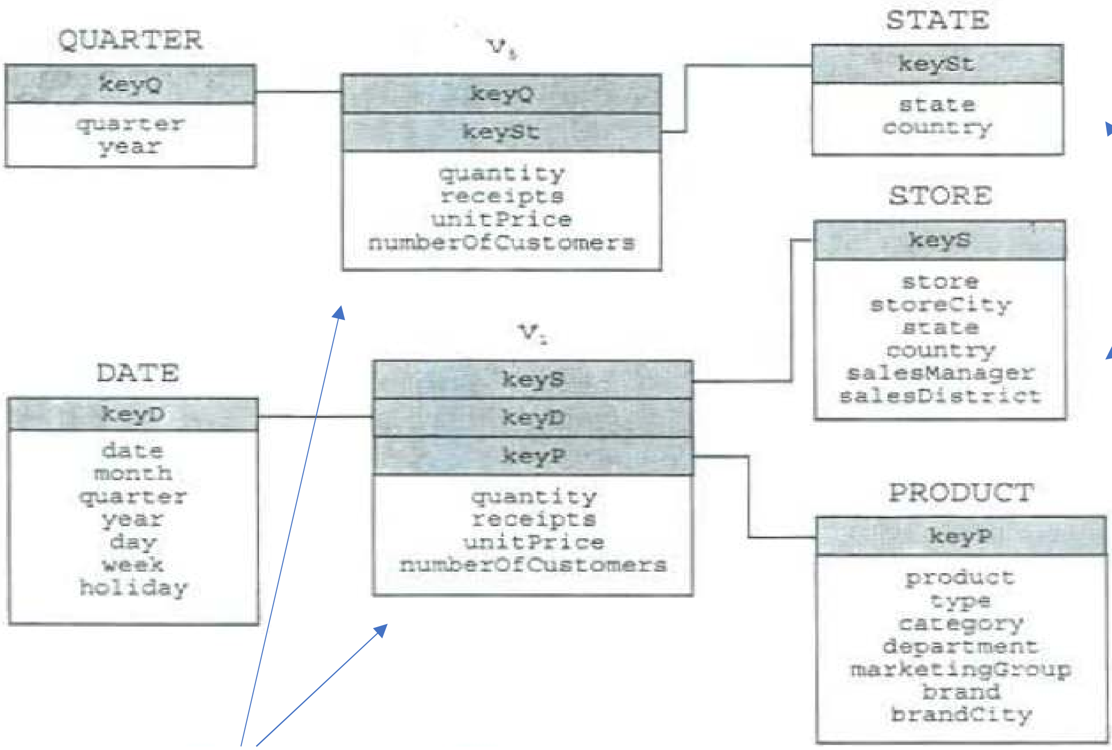
Constellation schema



Fact tables. The fact table V_2 has a dimension which is completely aggregated. It has not a foreign key referencing that dimension.

Shared Dimension tables: in a constellation schema the dimension tables are shared by the fact tables.

Multiple star schemata

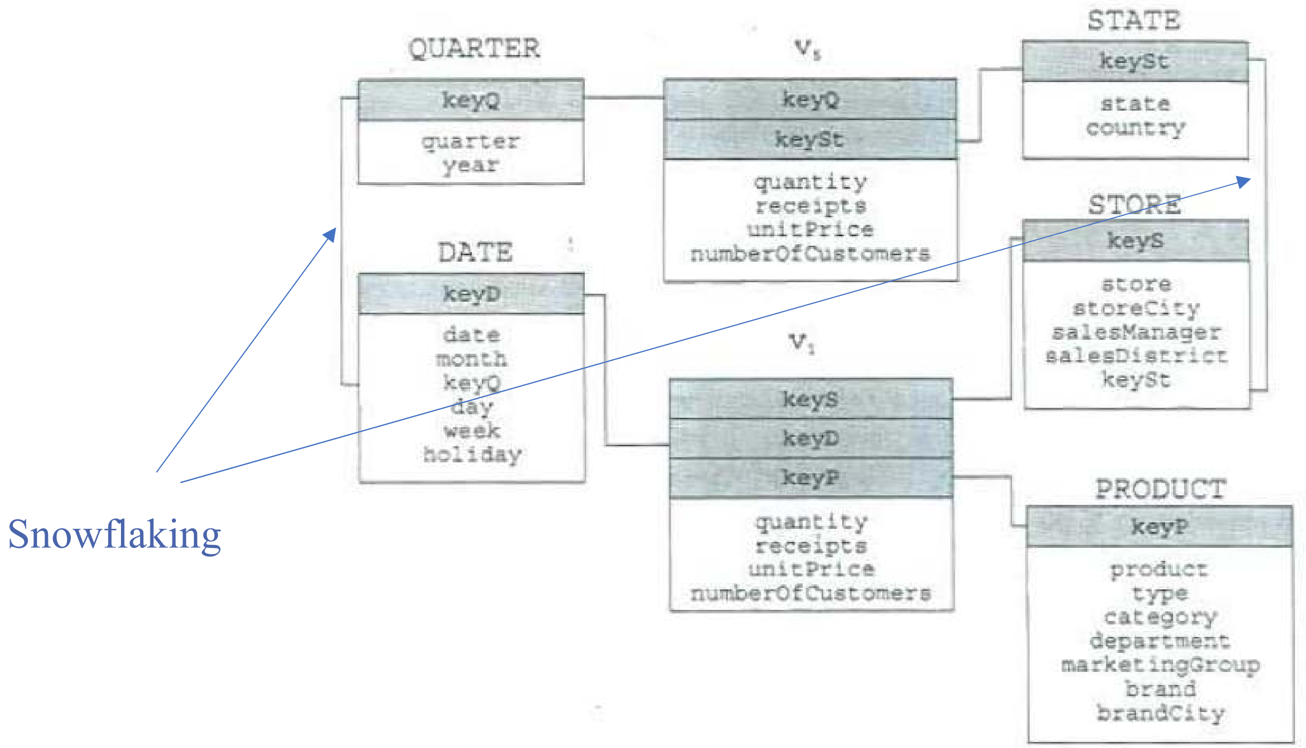


Star schemata

Fact tables

Hybrid solution (snowfalking)

- The snowflaking on dimension is applied on dimension tables in correspondence with the aggregation levels of the secondary fact views.
- It reduces the need of replication in the dimension tables.



Handling dynamic hierachies

- Dimension tables can have values changing in time.
- Interesting temporal scenarios: ***Yesterday for today, today for yesterday, today or yestarday, today and yesterday.***
- In the design phase the designer has to choose the scenario to be modelled.
- Handling the dynamic aspect of hierarchies implies extra cost in terms of disk space and may have bad effects on performance.

Status on 1/1/2008

store	salesManager
EverMore	Smith
ProFitsOnly	Johnson
SmartMart	Johnson

Status on 1/1/2009

store	salesManager
EverMore	Johnson
AllEvenMore	Smith
ProFitsOnly	Johnson
SmartMar	Johnson

Sales Events

store	date	quantity
EverMore	2/8/2008	100
ProFitsOnly	10/18/2008	100
SmartMart	12/25/2008	100
EverMore	2/8/2009	100
AllEvenMore	7/5/2009	100
ProFitsOnly	10/18/2009	100
SmartMart	12/25/2009	100

FIGURE 8-14 Evolution of the store hierarchy and sales events from 2008 to 2009

Handling dynamic hierarchies: type 1

- It only supports the scenario ***today for yesterday***
- The pure star schema sufficies
- When the value of a dimensional attribute changes it is simply required to overwrite (update) the past value. All the events in the fact table previously associate to the old value are now associate with the fresh.

Status on 1/1/2008

store	salesManager
EverMore	Smith
ProFitsOnly	Johnson
SmartMart	Johnson

Status on 1/1/2009

store	salesManager
EverMore	Johnson
AllEvenMore	Smith
ProFitsOnly	Johnson
SmartMar	Johnson

Sales Events

store	date	quantity
EverMore	2/8/2008	100
ProFitsOnly	10/18/2008	100
SmartMart	12/25/2008	100
EverMore	2/8/2009	100
AllEvenMore	7/5/2009	100
ProFitsOnly	10/18/2009	100
SmartMart	12/25/2009	100

STORE

Status on 1/1/2008

keyS	store	salesManager	.....
1	EverMore	Smith	.....
2	ProFitsOnly	Johnson	.....
3	SmartMart	Johnson	.....
.....	.....	.....	.....

STORE

Status on 1/1/2009

keyS	store	salesManager	.....
1	EverMore	Johnson	.....
2	ProFitsOnly	Johnson	.....
3	SmartMart	Johnson	.....
4	AllEvenMore	Smith	.....
.....	.....	.....	.....

Handling dynamic hierarchies: type 2

- It supports the scenario ***today or yesterday***
- The standard star schema sufficies
- An event stored into a fact table has to be associated with the hierarchy instance that it was valide when the event took place.
- The update of a hierachy implies the insertion of a new record for the new attributes in the dimension table.
- New events are now associated only to the newly inserted dimensional record.
- The type 2 allows to partition the events with respect to the time of the change without using additional temporal marks.
- In case of high dynamicity the dimension table can increase its size quickly.

Handling dynamic hierarchies: type 2

STORE

Status on 1/1/2008

keyS	store	salesManager	.....
1	EverMore	Smith	.....
2	ProFitsOnly	Johnson	.....
3	SmartMart	Johnson	.....
.....	.....	.....	.....

STORE

Status on 1/1/2009

keyS	store	salesManager	.....
1	EverMore	Smith	.....
2	ProFitsOnly	Johnson	.....
3	SmartMart	Johnson	.....
4	AllEvenMore	Smith	.....
5	EverMore	Johnson	.....
.....	.....	.....	.....

Sales for sale managers

salesManager	year 2008
Johnson	200
Smith	100

salesManager	year 2008	2009
Johnson	200	300
Smith	100	100

Handling dynamic hierarchies: type 3

- **It supports all the temporal scenarios**
- dimension tables should include one or more attributes that track previous version of attributes being changed and attributes modification data.
- **Requirements:**
- A **pair of time-stamps** giving the time validity of a dimensional record;
- A **master attribute re**porting, for every changed record, the key value of the dimensional record from which each previous version record stems. (If a dimensional record has been changed many times the refence is to the first original record)
- It is a modification of the pure star schema.
- For each modification of the hierarchy, a new record in the dimensional table is added and the values of the end **time-stamp and the master attribute are updaded.**

Handling dynamic hierarchies: type 3

STORE

Status on 1/1/2008

keyS	store	salesManager	.....	from	to	master
1	EverMore	Smith	.....	1/1/2008	–	1
2	ProFitsOnly	Johnson	.....	1/1/2008	–	2
3	SmartMart	Johnson	.....	1/1/2008	–	3
.....	.....	.....	.....	.....	.....	.....

STORE

Status on 1/1/2009

keyS	store	salesManager	.....	from	to	master
1	EverMore	Smith	.....	1/1/2008	12/31/2008	1
2	ProFitsOnly	Johnson	.....	1/1/2008	–	2
3	SmartMart	Johnson	.....	1/1/2008	6/30/2008	3
4	BigMart	Johnson	.....	7/1/2008	10/31/2008	3
5	HyperMart	Johnson	.....	11/1/2008	–	3
6	AllEvenMore	Smith	.....	1/1/2009	–	6
7	EverMore	Johnson	.....	1/1/2009	–	1
.....	.....	.....	.....	.....	.....	.....

Handling dynamic hierarchies: type 3

By grouping by the master attribute it is possible to obtain all the dimensional records obtained by updating a particular record.

- **Implementation of temporal scenarios:**
- **Today for yesterday**: **1) find the tuples of the currently valid records in dimension table** (NULL value in the field **To**); 2) find the records from which the current are derived (using the master attribute); 3) make access to the fact table.
- **Yesterday for today:** 1) fixed a date, find the records valid at that date (using timestamps); proceed as in the previous case.
- **Today or yesterday:** it sufficies to consider each dimensional record without considering timestamps o master attribute (exactly as in type 2).
- **Today and yesterday:** one has to consider only the records which have not been modified during the interval of time of interest (using the timestamp pairs).

Logical modelling

Logical modelling

In the logical modelling step one has to design the structure of the datamarts using the chosen logical model. The design adopt suitable optimization choices.

- **Activities:**
- The dimension fact model are encoded into logical schemata: star schemata, snowflake schemata, constellation schemata, etc
- **Design of the materialization strategy.**
- **Design of the fragmentation strategy.**

From the conceptual design to the logical design

A design specified by a Dimensional Fact Model can be easily translated into a ROLAP model.

- The basic aspects (fact, dimensions, hierarchies) can be easily modelled by a STAR schema:
- The fact table includes the measures and the descriptive attributes directly associated with the facts.
- There is a dimension table for each hierarchy including all the dimensional and descriptive attributes.
- Specific encodings can be used for the advanced features of the DFM

Descriptive Attributes

The descriptive attributes are not used for aggregation.

▶ A descriptive attribute associated with a dimensional attribute is included in the dimensional table where the dimensional attribute occurs.

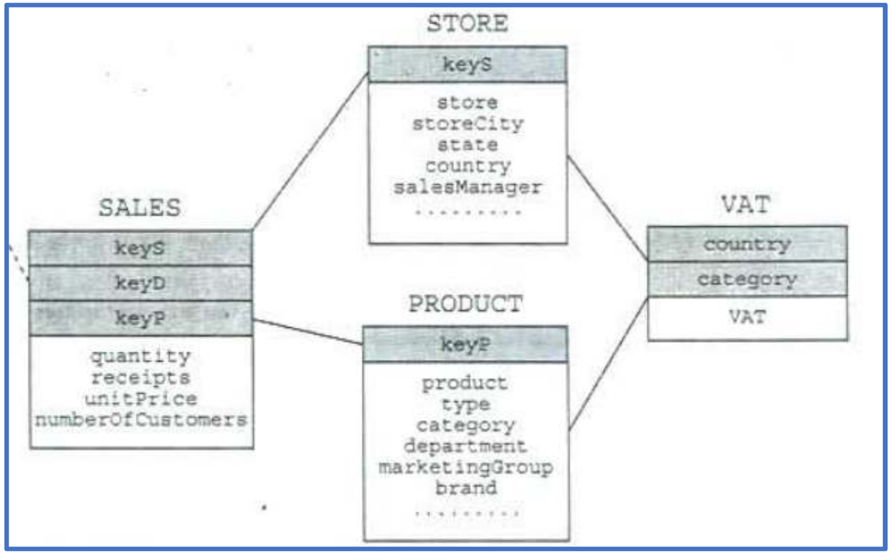
▶ A descriptive attribute associated with a fact is included in the fact table.

▶ A descriptive attribute associated with a fact does not occur in the secondary fact tables obtain by aggregating the fact table.

Cross-dimensional attributes

They conceptually define a many-to many association between two or more dimensional attributes

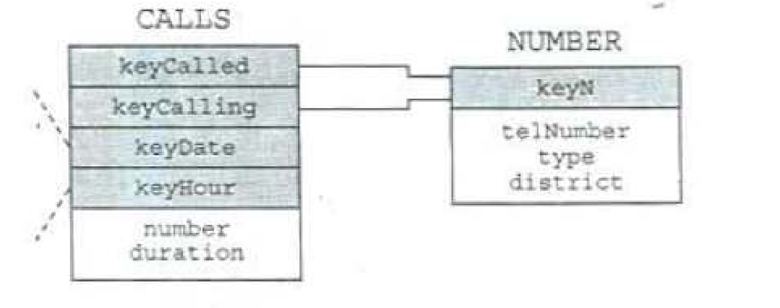
The translation at the logical level requires a bridge table including the involved dimensional attributes and the cross-dimensional attributes.



Shared hierarchies

Case 1: two hierarchies have exactly the same attributes which are used with different meanings

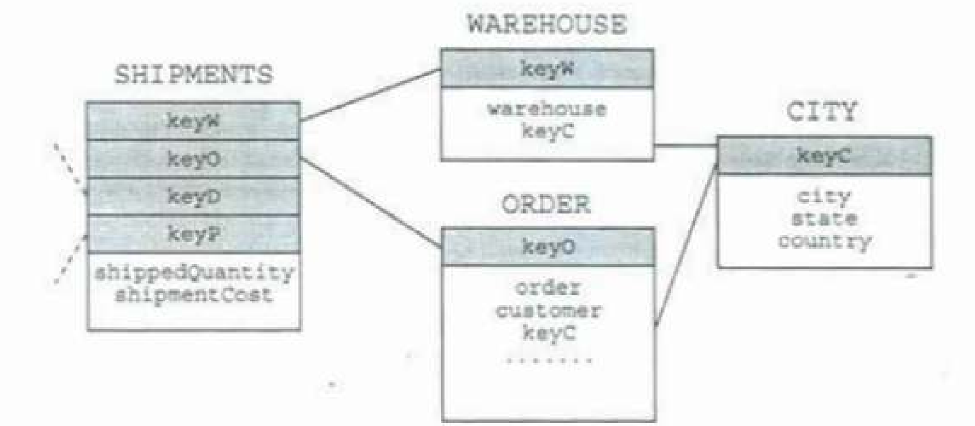
The two hierarchies are modelled by the same dimension table.



Caso 2: Two hierarchies share only a subset of the attributes. Implementation options:

▶ Two separate tables with duplication of attributes.

▶Snowflaking



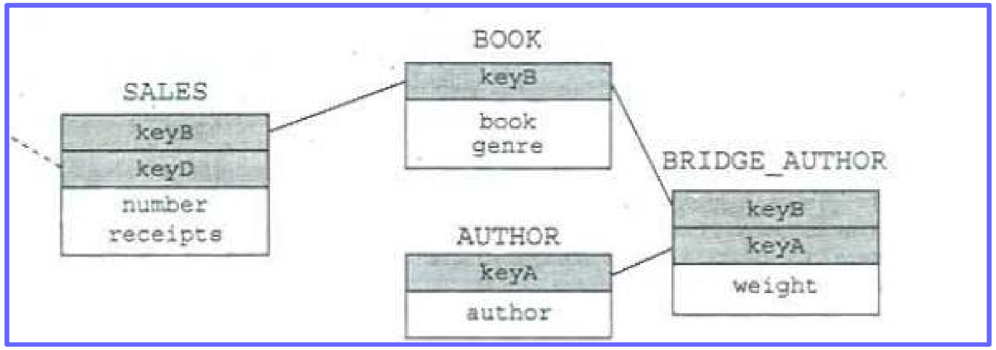
Multipli arcs (1)

Multiple arcs conceptually represent many-to-many associations

▶**Solution 1**. (it is not neither a star nor a snowflake schema)

▶ Use a bridge table as in the standard relational setting for encoding the association.

▶ Include a normalized weighting attribute in the bridge table to allow a weighted aggregation (the sum of the weights of elements in the same association is 1)



Multiple arcs (2)

Weighted queries.

```
SELECT A.author, SUM(S.Receipts*B.weight)
FROM Author A, Bridge_Author BA, Book B, Sales S
WHERE S.keyB = B.keyB AND A.keyA = BA.keyA AND
        B.keyB=BA.KeyB
GROUP BY A.author
```

Impact queries (do not use the weight)

```
SELECT A.author, SUM(S.number)
FROM Author A, Bridge_Author BA, Book B, Sales S
WHERE S.keyB = B.keyB AND A.keyA = BA.keyA AND
        B.keyB=BA.KeyB
GROUP BY A.author
```

Multiple arcs (3)

A way to avoid the bridge table and respect the star schema is to make the granularity finer. The association is modelled directly in the fact table (**pushdown**).

- Add to the fact table a new dimension for the attribute A in the many to many association (multiple arc).
- Possible descendents of A will be included in the hierarchy for A and the relative dimension table.
- Some measures can be weighted.

BOOK

keyB	book	genre
1	Facts & Crimes	Technical
2	Sounds Logical	Technical
3	The Right Measure	Current affairs
4	Facts: How and Why	Current affairs
5	The 4th Dimension	Science fiction

AUTHOR

keyA	author
1	Matteo Golfarelli
2	Stefano Rizzi

SALES

keyB	keyD	number	receipts
1	1	3	150
2	1	5	250
3	1	10	300
4	1	4	80
5	1	8	400

Pushdown of author on the fact table

Number of sold copies normalized

SALES

keyB	keyA	keyD	number	receipts
1	1	1	1.5	75
1	2	1	1.5	75
2	1	1	5	250
3	2	1	10	300
4	1	1	2	40
4	2	1	2	40
5	1	1	8	400

Bridge vs pushdown

- **The two techniques convey the same information.**
- Pushdown
- The **pushdown approach introduces redundancy in the** fact table
- The records in the fact table are replicated a number of times corresponding to the multiplicity of the arc.
- Naturally supports weighted queries and less naturally impact queries.
- The weighted query does not need a join with the bridge table (+) but works with a bigger fact table (-).
- Bridge table
- La bridge table stores the weights without any redundancy.
- The weight can be easily and efficiently modified (if needed).
- Supports both impact and weighted queries.
- The weighted query does need a join with the bridge table (-) but works with a smaller fact table (+).

Opzional arcs

- **The feature does not affect the logical structure and can be handled by suitably assigning NULL or special values to the attributes.**
- **The absence of a value for an attribute can be witnessed either by the NULL value o by a special value.**

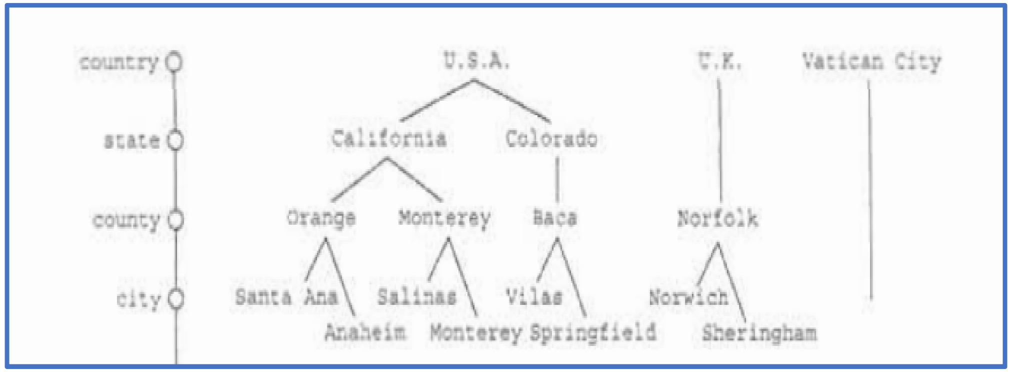
Optional hierachy:

- it cannot be handled by inserting a NULL value in the corresponding foreign key in the fact table
- it requires the insertion of a special record in the dimensional table witnessing the lack of values.
- the fact record references the special record.

Incomplete hierarchies

- **The feature does not affect the logical structure and can be handled by suitably assigning special values (placeholders) to the attributes.**
- The possible solutions differ for the choice of the placeholder:
- **Balancing by esclusion**:
- all the missing attribute values are associated with a same generic placeholder.
- it is a good option if many records have missing attribute values.
- it breaks regular roll-up semantics (using the same value, different hierarchial levels can be aggregated)
- **Downward Balancing**:
- **the missing value in the dimensional record is filled with the value of the attribute immediately preceeding in the hierarchy.**
- **It does not break the roll-up semantics.**
- **Upward Balancing**:
- **the missing value in the dimensional record is filled with the value of the attribute immediately following in the hierarchy.**

Incomplete hierarchies



USA	Vatican City	UK	UK
California	Other	Other	Other
Orange	Other	Norfolk	Essex
Santa Ana	Other	Norwich	Epping

Roll-up from state to country

USA	Vatican City	UK

California	Other

Figure 9-9 Balancing by exclusion for the incomplete hierarchy of Figure 5-18

USA	Vatican City	UK	UK
California	Vatican City	Norfolk	Essex
Orange	Vatican City	Norfolk	Essex
Santa Ana	Vatican City	Norwich	Epping

Roll-up from state to country

USA	Vatican City	UK

California	Vatican City	Norfolk	Essex

Figure 9-10 Top-down balancing for the incomplete hierarchy of Figure 5-18

USA	Vatican City	UK	UK
California	Vatican City	UK	UK
Orange	Vatican City	Norfolk	Essex
Santa Ana	Vatican City	Norwich	Epping

Roll-up from state to country

USA	Vatican City	UK

California	Vatican City	UK

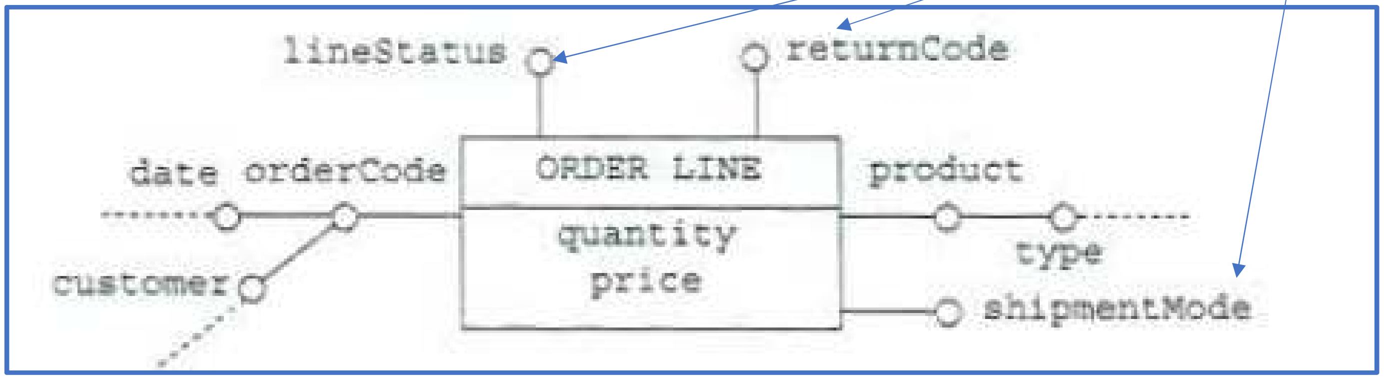
Figure 9-11 Bottom-up balancing for the incomplete hierarchy of Figure 5-18

Degenerate Dimensions

A dimension is said degenerate if it consists of only one attribute (the hierarchy is missing). Options:

- **Define a dimension table (in the standard way)**
- (a good solution when the length of the attribute is much more than the length of the surrogate key)
- **Include the dimensional attribute directly in the fact table.**
- **Create a unique dimension table for all (or a subset) of the degenerate dimensions.**

Degenerate dimensions



Materialized views

- View Materialization is the selection of a set of secondary views obtained from the data stored in the primary view.
- The choice of the set of views to be materialized it depends on project goals. Possible goals:
    - Minimization of a cost function
    - Meeting a system-oriented constraint.
    - Meeting a user-oriented constraint.
- **Minimization of a cost-function**:
- Some selection techniques consider the workload the datamart has to cope with.
- The total cost of the workload is given by the weighted sum of the cost of the query to be performed.
- The weight of each query is related to the frequency of the query or the importance of the query for the user.

Materialized views

- View maintenance cost
- The materialized views need a periodic update to replicate changes in the operational data.
- The maintenance cost is the cost of the queries necessary to transmit those updates from operational sources to views.
- The cost calculation is quite complex because of the many different solutions that can be adopted:
  - A simple way is to issue update queries directly accessing the operational database.
  - Other techniques are based on incremental view updates from already updated views.
  - Other technique replicate operational data source tables in that amount to reduce the number of remote queries.

Materialized views: systems constraints

- Limitations on available resources.
- **Disk space:**
- Space made available to a Data Mart is normally the main constraint on view materialization
- The available disk space must be shared with other optimization structures (e.g. indexes).
- Normally indexes uses a very high percentage of the available disk space.
- The choice of how to distribute free disk space becomes a major design decision.
- **Update time**
- A data mart is normally updated when the data warehouse system is offline.
- The time for maintenance is limited and it is shared with other regular operations like backup, synchronization, and so on.
- It is not possible to materialize more views than the number of views that can be updated in the available time.

Materialized views: user constraints

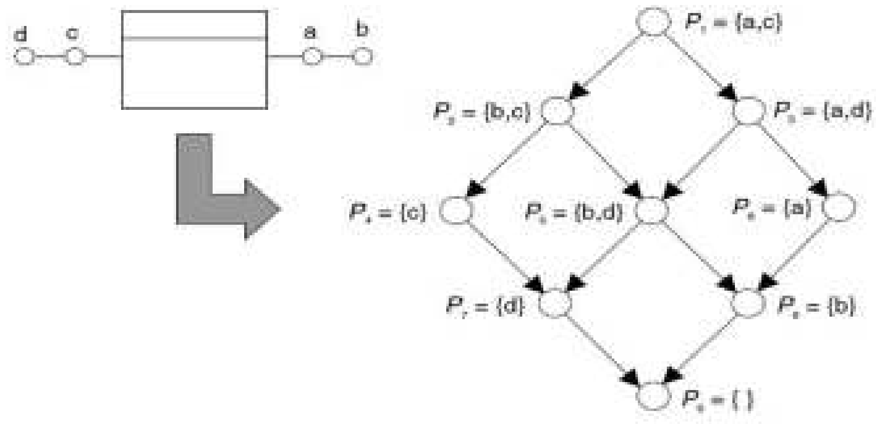
- **Query response time**:
- The greatest admitted time in between issuing a query and the response time.
- User may specify that limit for each query, thus showing how urgently each query should be answered.
- **Data freshness**.
- Maximum limit on time since the last update overview used to execute a query.
- For each query it is possible to define the “freshness” of data that can be used to answer the query.
- The goals are clear in conflict with one another. If constraints are too restrictive the problem of view materialization may not offer any solution.

View materialization problem

- The view materialization problem is a problem with minimizing workload response time and complying with the system constraints (disk space and update time)
- The search of solutions exponentially grows with the number of dimension attributes which determine the aggregations patterns.
- Each combination of dimension attributes (one for each dimension) determine a possible pattern of data aggregation.
- Even neglecting hierarchies, a fact related with N dimensions has 2^N possible aggregation patterns.
- **The approaches to solve the problem usually act in two steps**:
- Select among the possible materializable views, the subset of those which can effectively be useful for a given workload;
- by using euristic algorithms determine the subset of useful queries that minimizes the cost function fulfilling all the system and user constraints.

Materialization of views: the multimensional lattice

- **A view is uniquely determined by its aggregation pattern (the list of dimensional attributes)**
- The patter include a dimensional attribute for each dimension.
- The pattern does not fix explicitly the measures and the support information needed by algebraic operators to calculate measures from aggregate data.
- A multidimensional lattice can be used to model the partial order of roll-up of patterns.
- The oriented edges represents the partial ordering
- Intuitive meaning: if $P_i < P_j$ data in P_i allows to compute those in P_j



Materialization of views: the multimensional lattice

- The dimension of the multidimensional lattice exponentially grows with respect to the number of attributes.
- It is impractical the materialization of all the possible views.
- It is reasonable to consider only the patterns (views) which effectively optimize the execution cost of a specific workload (**candidate views**)
- **The candidate** views:
- Give the exact result of a frequent query
- Can be used to solve more than one query
- The data required by two or more queries can obtained by aggregating from the data in a candidate view
- Given a relevant frequently required queries, the materialization of all the queries optimizes the query performance but usually violates
  - space constraints
  - time for updating constraints

Materialization of views

- **Rules for materialization**
- **One should consider the opportunity of materializing a view when**
- It solves a very frequent query
- It can be used to solve many querries.
- **A view should not be materialized when**
- the pattern of the view is very similar to that of an already materialized view.
- the pattern is very fine (close to that of the fact table)
- the materialization does not reduce the workload by a relevant rate.

Partitioning

- Is the operation of fragmenting a table in parts called fragments in order to increase the performance of the system.
- Partitioning is a technique used both by centralized and distributed systems
- Specific data warehouse properties such as major data redundancy and existing multiple multidimensional cubes correlated by drill-across queries add new interest to fragmentation techniques
- **The advantages of fragmentation are visible if the** DW is implemented in a distributed architetcture.

Partitioning Tecniques:

- **Horizontal partitioning:**
- A relation is fragmented in parts each of them contains a subset of the records of the relation (each record has all the attributes of the original relation).

Vertical Partitioning:

- a table is partitioned in fragments containing a projection of a subset of all the records.
- the projection includes all the attributes in the primary key.

Vertical fragmentation

- The term vertical fragmentation or multi cubing stands for a set of views created to contain a subset of the measures defined in one or more fact schemata.
- The result of vertical fragmentation process must enjoy the following properties:
- **Consistency**
- Fragmented group by sets must be chosen among candidate view group by sets.
- **Completeness.**
- Every measure must be included in a primary fragment (a fragment of the primary view).
- **Non redundancy**
- A measure cannot be inserted in two or more fragments having the same aggregation pattern.

Vertical fragmentation: motivations

- **Ottimized workload cost**.
- Useful whenever only a subset of the measures in a cube are required by queries.
- Merging can be cost-effective if the number of drill across queries is large.
- **Saved space**.
- all the measures of the fact schemata must be included in the primary fragments in order to avoid information loss.
- Since the previous requirement is not necessary in the secondary fragments, some measures can be neglected in the secondary fragments resulting in a space saving.
- Reduced key replication
- The fragments are usually created for aggregated pattern, where one or more dimensions are completely aggregated (the foreign keys for the collapsed dimensions are note reported).

Frammentazione verticale (2)

- **The vertical fragmentation can be seen as a generalization of the view materialization.**
- The elements that make you select specific fragments are the measures requested by queries at different aggregation level
- To the terminal those sets you must evaluate:
  - the number of times that two measures are required at the same time
  - the number of times those measures are requested separately
- The non-fragmented solution should be preferred when almost all the measures are simultaneously required by the fixed workload.

Horizontal fragmentation

- The term horizontal fragmentation refers to a set of views created to contain all the measures of a specific factor schema but only the subset of tuples that meet specific boolean predicates

- The result of horizontal fragmentation process must enjoy the following properties:

- **Consistency.**

- The pattern of the fragments must be chosen among those of the candidate views (meaningful fragmentation).

- **Completeness.**

- Every record of the primary view must be included in a primary fragment (lossless fragmentation).

- **Non-redundancy.**

- A record cannot be included in two or more fragments having the same aggregation pattern.

- **Time is an attribute often used for horizontal fragmentation because it is often used in queries.**

- Time based fragmentation follows insertion orders. When the fact table updates new records can be appended to the most recent fragment.

Horizontal fragmentation

- In contrast to vertical fragmentation the horizontal fragmentation does not lead to any additional cost in terms of disk memory space used.
- The horizontal fermentation can also be used as a starting point for the parallel execution of queries.
- The reasons for using horizontal fragmentation are similar to those for using vertical fragmentation.
- In particular a reduction in query execution time is the result of the opportunity to access smaller fact tables that are free from those records that do not satisfy specific conditions.