Progettazione Logica

- Per Progettazione Logica si intende la trasformazione di **uno o più schemi di fatto** in **uno schema logico** (schema logico del DataMart)
  - ✓ Schema Logico da implementare su calcolatore elettronico
- La struttura multidimensionale dei dati può essere rappresentata utilizzando **due distinti modelli logici**:
  - ✓ MOLAP (*Multidimensional On-Line Analytical Processing*) memorizzano i dati tramite strutture multidimensionali (es. vettori multidimensionali).
  - ✓ ROLAP (*Relational On-Line Analytical Processing*) utilizza il modello relazionale per rappresentare i dati multidimensionali.

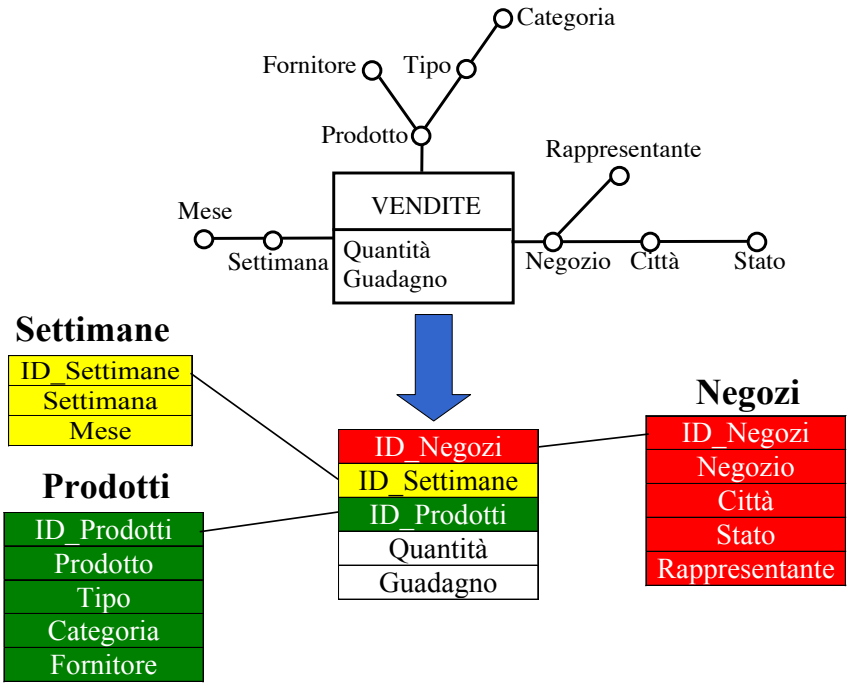
- Noi useremo il modello ROLAP:
  - ✓ Lo schema logico del DataMArt è uno schema relazionale
  - ✓ Il DataMart è un RDB (Relation DataBase)

ROLAP: lo schema a stella

- La modellazione multidimensionale su sistemi relazionali è basata sul cosiddetto *schema a stella* (*star schema*) e sulle sue varianti.
- Uno schema a stella è composto da:
  - ✓ Un insieme di relazioni DT_1…, DT_n, chiamate *dimension table*, ciascuna corrispondente a una dimensione. Ogni DT_i è caratterizzata da una chiave primaria (tipicamente surrogata) d_i e da un insieme di attributi che descrivono le dimensioni di analisi a diversi livelli di aggregazione.
  - ✓ Una relazione *FT*, chiamata *fact table*, che importa le chiavi di tutte le dimension table. La chiave primaria di *FT* è data dall'insieme delle chiavi esterne dalle dimension table, d_1…, d_n; *FT* contiene inoltre un attributo per ogni misura.

- Le Dimension Table sono completamente denormalizzate
  - 👍 Basta un join per recuperare tutti i dati relativi a una dimensione
  - 👎 La denormalizzazione introduce una forte ridondanza nei dati

Lo schema a stella



Lo schema a stella

ID_Negozi	Negozio	Città	Stato	Rappresentante
1	N1	RM	I	R1
2	N2	RM	I	R1
3	N3	MI	I	R2
4	N4	MI	I	R2

Dimension Table

ID_Negozi	ID_Sett	ID_Prodotti	Quantità	Guadagno
1	1	1	100	100
1	2	1	150	150
3	3	4	350	350
4	4	4	200	200

Fact Table

ID_Sett.	Settimana	Mese
1	Gen1	Gen.
2	Gen2	Gen.
3	Feb1	Feb.
4	Feb2	Feb.

Dimension Table

ID_Prodotti	Prodotto	Tipo	Categoria	Fornitore
1	P1	A	X	F1
2	P2	A	X	F1
3	P3	B	X	F2
4	P4	B	X	F2

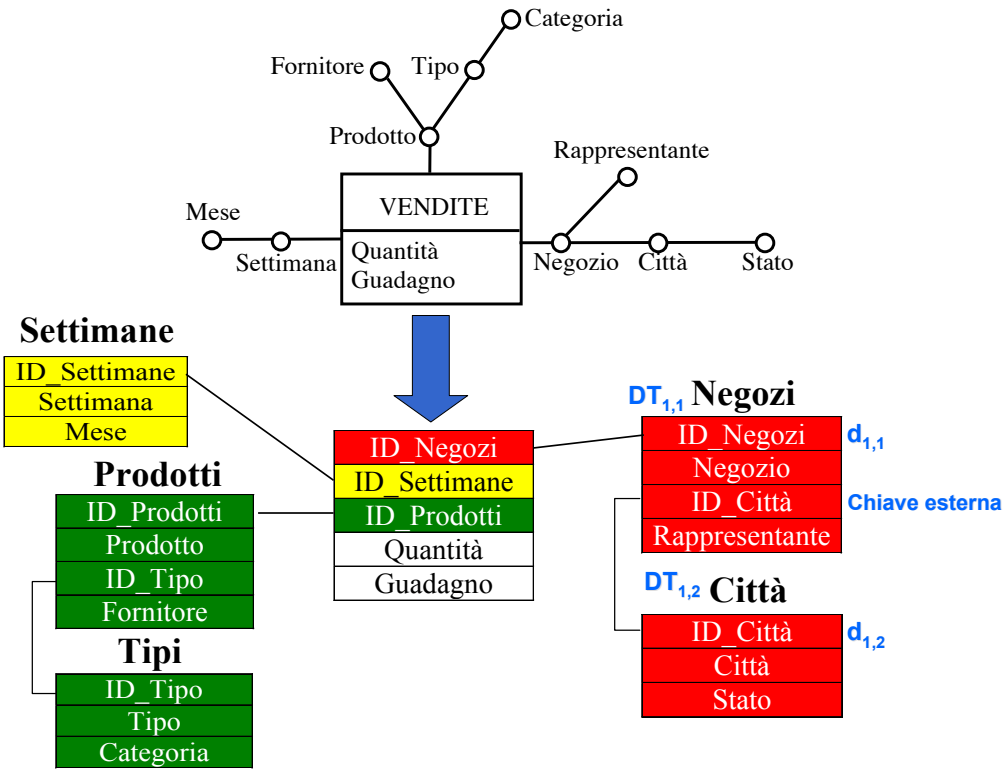
Lo snowflake schema

- Lo schema a fiocco di neve (*snowflake schema*) riduce la denormalizzazione delle dimension table DT_i degli schemi a stella eliminando alcune delle dipendenze transitive.

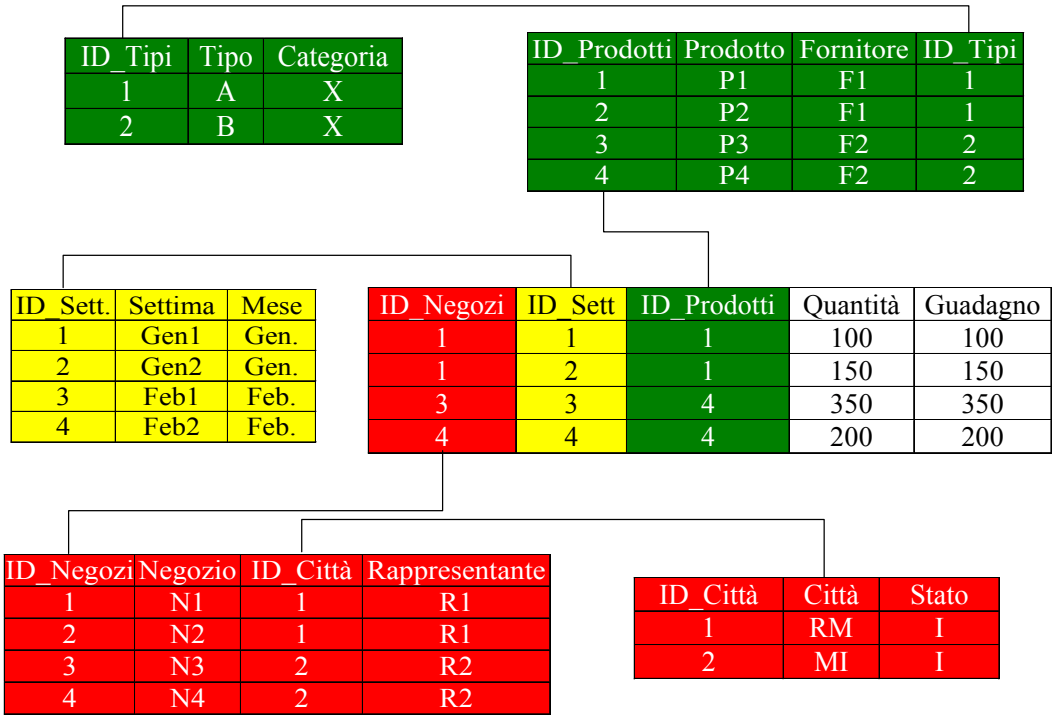
- Le dimension table $DT_{i,j}$ di questo schema sono caratterizzate da:
  - ✓ una chiave primaria (tipicamente surrogata) $d_{i,j}$
  - ✓ il sottoinsieme degli attributi di DT_i che dipendono funzionalmente da $d_{i,j}$.
  - ✓ zero o più chiavi esterne a importate da altre $DT_{i,k}$ necessarie a garantire la ricostruibilità del contenuto informativo di DT_i.

- Denominiamo *primarie* le dimension table le cui chiavi sono importate nella fact table, *secondarie* le rimanenti.

Lo snowflake schema



Lo snowflake schema



ID_Tipi	Tipo	Categoria
1	A	X
2	B	X

ID_Prodotti	Prodotto	Fornitore	ID_Tipi
1	P1	F1	1
2	P2	F1	1
3	P3	F2	2
4	P4	F2	2

ID_Sett.	Settima	Mese
1	Gen1	Gen.
2	Gen2	Gen.
3	Feb1	Feb.
4	Feb2	Feb.

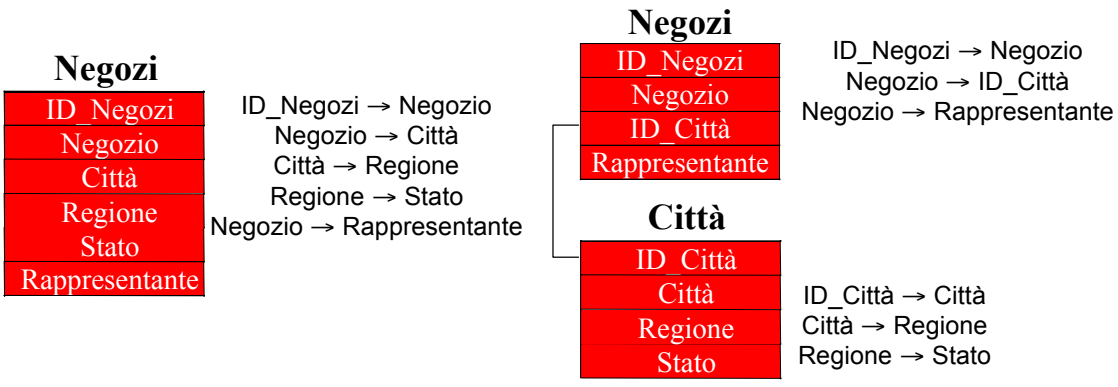
ID_Negozi	ID_Sett	ID_Prodotti	Quantità	Guadagno
1	1	1	100	100
1	2	1	150	150
3	3	4	350	350
4	4	4	200	200

ID_Negozi	Negozio	ID_Città	Rappresentante
1	N1	1	R1
2	N2	1	R1
3	N3	2	R2
4	N4	2	R2

ID_Città	Città	Stato
1	RM	I
2	MI	I

Normalizzazione con lo snowflake schema

- Le specifiche caratteristiche degli schemi a stella richiedono particolare attenzione affinché nella nuova relazione sia spostato il corretto insieme di attributi
- La presenza di più dipendenze funzionali transitive in cascata fa sì che, affinché la decomposizione sia efficace, tutti gli attributi che dipendono (transitivamente e non) dall'attributo che ha determinato lo snowflaking siano posti nella nuova relazione



Progettazione logica

- Include l'insieme dei passi che, a partire dallo schema concettuale, permettono di determinare lo schema logico del data mart

- Le principali operazioni da svolgere durante la progettazione logica sono:
  1. Scelta dello schema logico da utilizzare (es. star/snowflake schema)
  2. Traduzione degli schemi di fatto

Star VS Snowflake

- Esistono pareri contrastanti sull'utilità dello snowflaking, in quanto esso contrasta con la filosofia del data warehousing di avere tabelle denormalizzate.

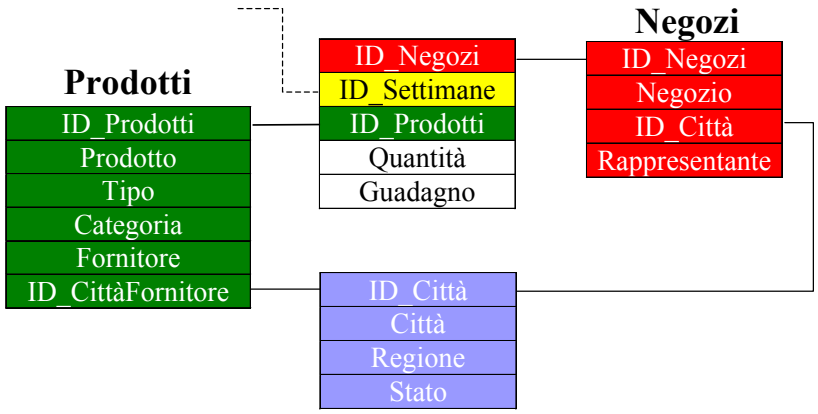
- Nello **star schema** le Dimension Table sono **denormalizzate**
  - 👍 Basta un join per recuperare tutti i dati relativi a una dimensione
  - 👎 La denormalizzazione introduce una forte ridondanza nei dati

- Nello **Snowflake** schema le Dimension Table sono **normalizzate** (eventualmente solo alcune di esse)
  - 👍 La normalizzazione elimina la ridondanza nei dati e riduce quindi lo spazio richiesto per la memorizzazione
  - 👎 E' necessario un join tra tutte le tabelle secondarie per recuperare tutti i dati relativi a una dimensione

- Nella nostra progettazione logica non faremo considerazioni sull'**efficienza** dello schema logico ottenuto con le due soluzioni (spazio occupato, velocità nelle interrogazioni …)

Star VS Snowflake

- Oltre all'efficienza delle due soluzioni, si può considerare anche la *semplicità* dello schema logico: uno snowflak ha sì più tabelle, ma esse possono essere usate in più schemi
- Lo SnowFlake può essere utile quando una parte di una gerarchia è comune a più dimensioni (dello stesso schema o di schemi diversi) . Nell'esempio la dimension table secondaria è riutilizzata per più gerarchie



Dagli schemi di fatto agli star schema

- La regola di base per la traduzione di uno schema di fatto in schema a stella prevede di:

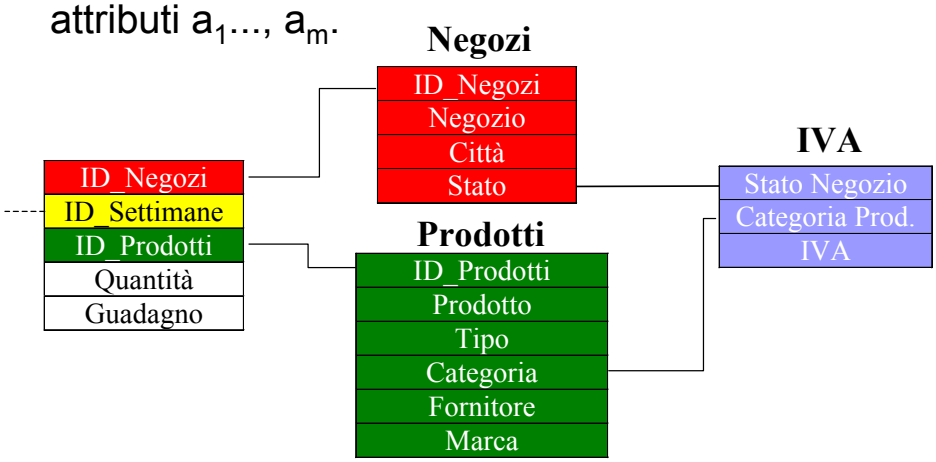
  Creare una fact table contenente tutte le misure e gli attributi descrittivi direttamente collegati con il fatto e, per ogni gerarchia, creare una dimension table che ne contiene tutti gli attributi.

- In aggiunta a questa semplice regola, la corretta traduzione di uno schema di fatto richiede una trattazione approfondita dei costrutti avanzati del DFM

- Attributi descrittivi
  - Se collegato a un attributo dimensionale, va incluso nella dimension table che contiene l'attributo.
  - Se collegato direttamente al fatto deve essere incluso nella fact table.

Attributi cross-dimensionali

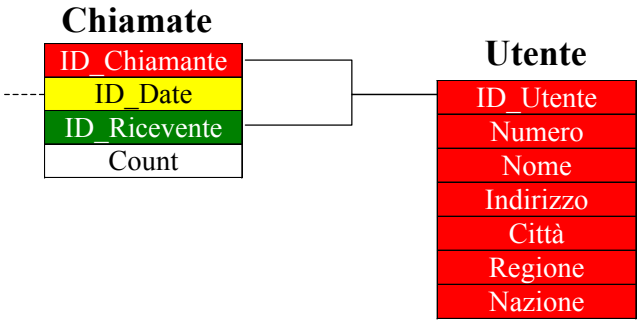
- Dal punto di vista concettuale, un attributo cross-dimensionale b definisce un'associazione molti-a-molti tra due o più attributi dimensionali a_1..., a_m.
- La sua traduzione a livello logico richiede l'inserimento di una nuova tabella che includa b e abbia come chiave gli attributi a_1..., a_m.



Gerarchie condivise

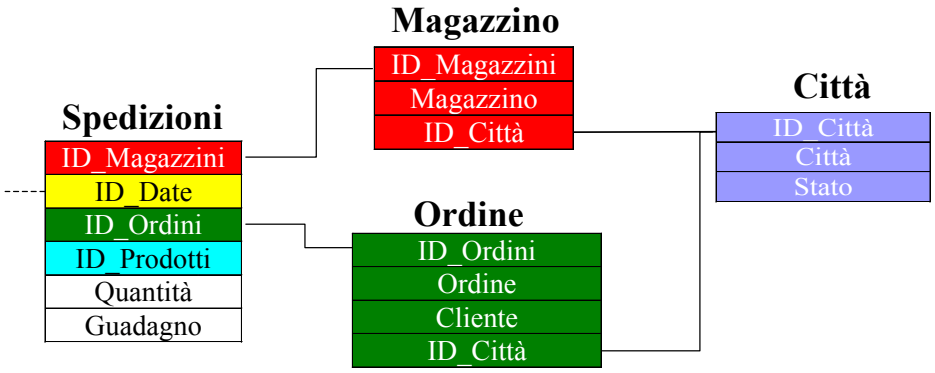
- Se una gerarchia si presenta più volte nello stesso fatto (o in due schemi di fatto diversi) non conviene introdurre copie ridondanti delle relative dimension table.

- Se le due gerarchie contengono esattamente gli stessi attributi sarà sufficiente importare due volte la chiave della medesima dimesion table



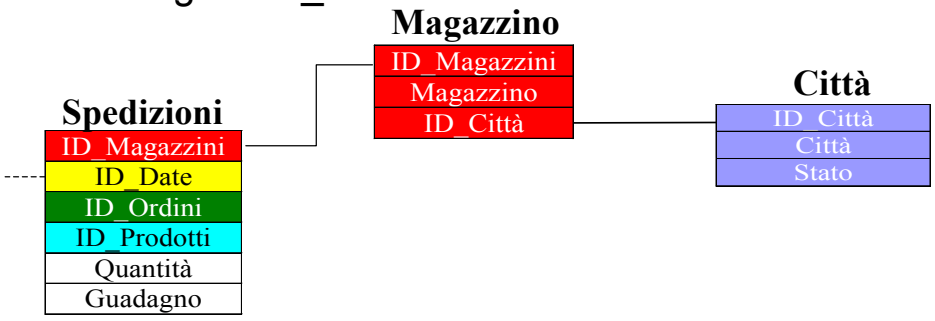
Gerarchie condivise

- Se le due gerarchie condividono solo una parte degli attributi è necessario decidere se:
  I. Introdurre ulteriore ridondanza nello schema duplicando le gerarchie e replicando i campi comuni.
  II. Eseguire uno snowflake sul primo attributo condiviso introducendo una terza tabella comune a entrambe le dimension table.



Chiavi Surrogate

- Chiave surrogata ID_CITTA nella tabella secondaria Città



- Senza chiave surrogata si usa direttamente la chiave *semantica* Citta (riportandola quindi in Magazzino)
- Anche se si ha un attributo in più, la chiave surrogata può ridurre lo spazio occupato in quanto è un codice *corto* quindi risparmio spazio quando si usa come foreign key (in Magazzino)
- Nella nostra progettazione logica non faremo considerazioni sull'**efficienza** dello schema logico : non useremo chiavi surrogate

Scenari temporali

- Il modello multidimensionale assume che gli eventi che istanziano un fatto siano ***dinamici***, e che i valori degli attributi che popolano le gerarchie siano ***statici***.
- Questa visione non è realistica poiché anche i valori presenti nelle gerarchie variano nel tempo dando vita alle gerarchie dinamiche (*slowly changing dimension*).
- L'adozione di gerarchie dinamiche implica un sovraccosto in termini di spazio e può comportare una forte riduzione delle prestazioni.

Scenari temporali

…. Sono possibili diverse soluzioni

- Oggi per ieri
  - ✓ I dati vengono interpretati in base all'attuale configurazione della gerarchia
  - ✓ Implementabile sullo schema a stella (Tipo I)
- Oggi o ieri
  - ✓ I dati vengono interpretati in base alla configurazione valida al momento in cui sono stati registrati
  - ✓ Implementabile sullo schema a stella (Tipo II)
- Ieri per oggi
  - ✓ I dati vengono interpretati in base alla configurazione della gerarchia valida in un particolare istante
  - ✓ Richiede la storicizzazione dei dati (Tipo III)
- Oggi e ieri
  - ✓ Come nello scenario oggi o ieri ma limitatamente ai dati che non hanno subito modifiche
  - ✓ Richiede la storicizzazione dei dati (Tipo III)

Un esempio

Situazione al 1/1/2001

negozio	responsabile
DiTutto	Rossi
NonSoloPile	Bianchi
NonSoloPane	Bianchi

Situazione al 1/11/2001

negozio	responsabile
DiTutto	Rossi
NonSoloPile	Bianchi
PaneEPizza	Rossi

Situazione al 1/7/2001

negozio	responsabile
DiTutto	Rossi
NonSoloPile	Bianchi
NonSoloPane	Rossi

Situazione al 1/1/2002

negozio	responsabile
DiTutto	Bianchi
NonSoloPile	Bianchi
PaneEPizza	Rossi
DiTuttoDiPiù	Rossi

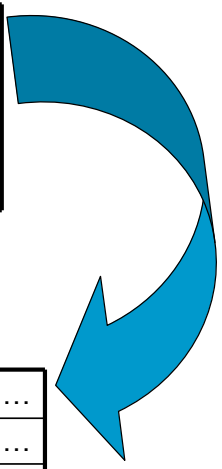
Gerarchie dinamiche: tipo I

- Supportano solo lo scenario oggi per ieri, pertanto tutti gli eventi, anche quelli passati, vengono interpretati in base all'attuale configurazione delle gerarchie senza tenere traccia del passato.
- Questa soluzione è realizzabile sullo schema a stella sovrascrivendo il vecchio valore con quello nuovo ogni volta che si verifica un cambiamento.

Gerarchie dinamiche: tipo I

Situazione al 1/1/2001

chiaveN	negozio	responsabile	…
1	DiTutto	Rossi	…
2	NonSoloPile	Bianchi	…
3	NonSoloPane	Bianchi	…



Tutte le vendite di NonSoloPane vengono attribuite a Rossi anche se erano state effettuate durante la gestione di Bianchi

Situazione al 1/7/2001

chiaveN	negozio	responsabile	…
1	DiTutto	Rossi	…
2	NonSoloPile	Bianchi	…
3	NonSoloPane	Rossi	…

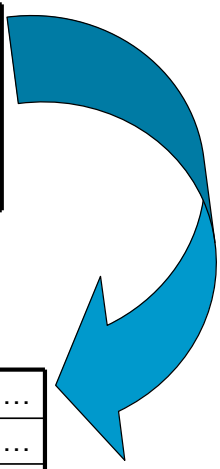
Gerarchie dinamiche: tipo II

- Supportano solo lo scenario oggi o ieri, e consentono di registrare la verità storica.
- Gli eventi memorizzati nella fact table vengono associati ai dati dimensionali che erano validi quando si è verificato l'evento.
- Questa soluzione è realizzabile sullo schema a stella: ogni modifica a una gerarchia comporta l'inserimento di un nuovo record che codifichi le nuove caratteristiche nella dimension table corrispondente.
- È possibile adottare strategie diverse per attributi appartenenti alla stessa gerarchia.

Gerarchie dinamiche: tipo II

Situazione al 1/1/2001

chiaveN	negozio	responsabile	…
1	DiTutto	Rossi	…
2	NonSoloPile	Bianchi	…
3	NonSoloPane	Bianchi	…



Dopo l'1/7 i record della fact table relativi a NonSoloPane importeranno il valore di chiaveN = 4

Situazione al 1/7/2001

chiaveN	negozio	responsabile	…
1	DiTutto	Rossi	…
2	NonSoloPile	Bianchi	…
3	NonSoloPane	Bianchi	…
4	NonSoloPane	Rossi	…

N.B. Solo le selezioni su campi che hanno subito modifiche sono sensibili alle modifiche stesse!!

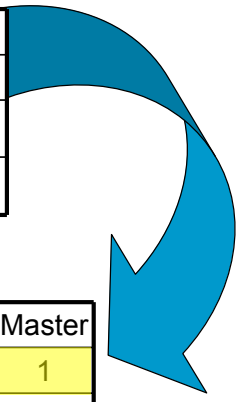
Gerarchie dinamiche: tipo III

- Supportano tutti gli scenari temporali. La loro adozione richiede la storicizzazione dell'attributo e non può pertanto essere basata sul classico schema a stella.
- Gli elementi necessari per la gestione di una gerarchia di tipo 3 sono:
  - ✓ Una coppia di marche temporali (*time-stamp*) che indichino l'intervallo di validità di una tupla;
  - ✓ Un meccanismo per individuare le tuple coinvolte in una serie di modifiche (tramite per esempio un attributo *master*).
- In uno schema così modificato la dinamicità viene gestita aggiungendo, per ogni modifica, un nuovo record nella dimension table e aggiornando di conseguenza i valori dei time-stamp e dell'attributo master.

Gerarchie dinamiche: tipo III

Situazione al 1/1/2001

chiaveN	negozio	responsabile	…	da	a	Master
1	DiTutto	Rossi	…	1/1/2001	—	1
2	NonSoloPile	Bianchi	…	1/1/2001	—	2
3	NonSoloPane	Bianchi	…	1/1/2001	—	3



Situazione al 1/1/2002

chiaveN	negozio	responsabile	…	da	a	Master
1	DiTutto	Rossi	…	1/1/2001	31/12/2001	1
2	NonSoloPile	Bianchi	…	1/1/2001	—	2
3	NonSoloPane	Bianchi	…	1/1/2001	30/6/2001	3
4	NonSoloPane	Rossi	…	1/7/2001	31/10/2001	3
5	PaneEPizza	Rossi	…	1/11/2001	—	3
6	DiTuttoDiPiù	Rossi	…	1/1/2002	—	6
7	DiTutto	Bianchi	…	1/1/2002	—	1

Gerarchie dinamiche: tipo III

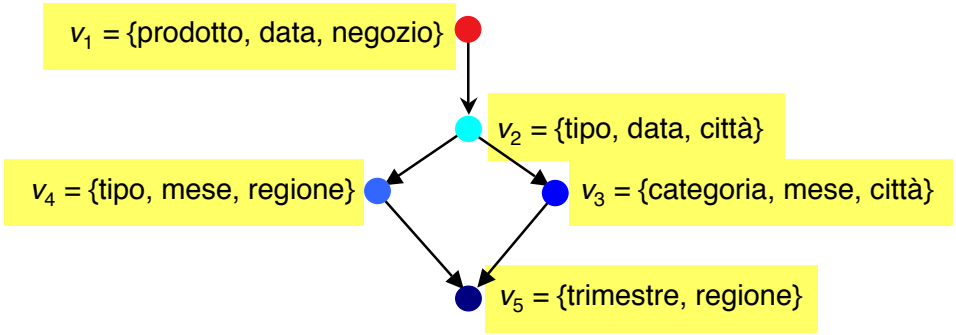
- Avendo a disposizione lo schema descritto in precedenza è facile realizzare i differenti scenari temporali:
  - ✓ **Oggi per ieri:** si identificano dapprima le tuple della dimension table attualmente valide (in base ai time-stamp) e per ciascuna si individuano eventuali altre tuple da cui esse hanno avuto origine
  - ✓ **Ieri per oggi:** fissata una particolare data si individuano le tuple valide in quel particolare momento, quindi si procede come nel caso precedente
  - ✓ **Oggi o ieri:** non richiede l'analisi delle marche temporali poiché l'aggiornamento delle tuple nelle dimension table avviene come per le gerarchie di tipo 2
  - ✓ **Oggi e ieri:** tramite i time-stamp si individuano le tuple della dimension table che non hanno subito modifiche durante l'intervallo di tempo di interesse e, limitandosi a queste, si procede come per lo scenario oggi o ieri.

Progetto Fisico e Viste

- Un significativo aumento delle prestazioni può essere ottenuto precalcolando i dati aggregati di uso più comune
- Misure definite con operatori Distributivi ed Algebrici permettono di calcolare dati aggregati a partire direttamente da dati parzialmente aggregati
- L'ottimizzazione usa il concetto di *vista materializzata:*
  - ✓ Ogni pattern secondario corrisponde ad una vista sul pattern primario
  - ✓ Vengono **materializzate le viste** (ovvero pre-calcolate e memorizzate in tabelle o altre strutture dati) corrispondenti ad alcuni pattern secondari, ovvero a quelli maggiormente utilizzati (**carico di lavoro**)
  - ✓ La scelta delle viste da materializzare è basata sul compromesso tra diversi vincoli, i principali dei quali sono
    - Tempo di costruzione ed aggiornamento delle viste materializzate, ovvero tempo di calcolo al momento del caricamento/refresh del DW
    - Spazio aggiuntivo richiesto

Le viste

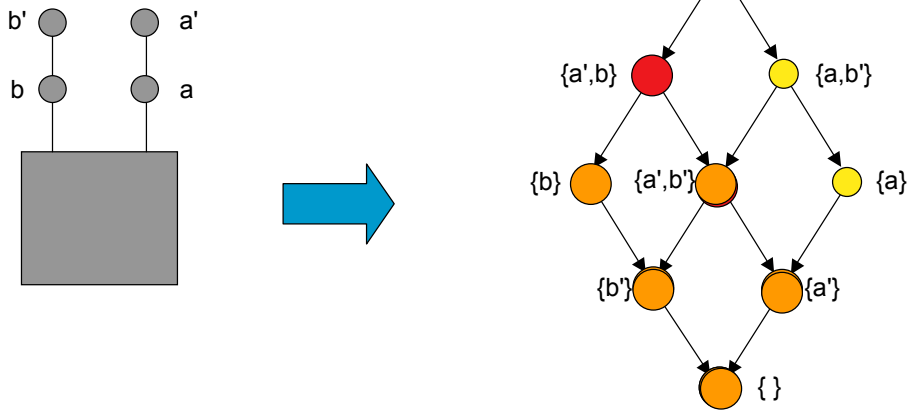
- Le viste possono essere identificate in base al livello (*pattern*) di aggregazione che le caratterizza



  - **Viste primarie:** corrispondono al pattern di aggregazione primario (non aggregato)
  - **Viste secondarie:** corrispondono ai pattern di aggregazione secondari (aggregati)

Risolvibilità delle interrogazioni

- Una vista v sul pattern p non serve solo per le interrogazioni con pattern di aggregazione p ma anche per tutte quelle che richiedono i dati a pattern p' più aggregati di p (p → p')



Reticolo di roll-up

Scelta delle viste

- È utile materializzare una vista quando:
  - ✓ Risolve direttamente una interrogazione frequente
  - ✓ Permette di ridurre il costo di esecuzione di molte interrogazioni
- Non è consigliabile materializzare una vista quando:
  - ✓ Il suo pattern di aggregazione è molto simile a quello di una vista già materializzata
  - ✓ Il suo pattern di aggregazione è molto fine
  - ✓ La materializzazione non riduce di almeno un ordine di grandezza il costo delle interrogazioni
- ➢ Tecniche di scelta delle viste da materializzare sono generalmente già implementate nei sistemi OLAP e all'utente viene solo chiesto di configurare alcuni parametri, quali lo spazio a disposizione per la memorizzazione delle viste da materializzare