Riassunto di Business intelligence: Datawarehouse

# Cos’è la business intelligence?

La business intelligence è una serie di strumenti utili per la raccolta e la trasformazione di dati in conoscenza e, sulla base di questa, prendere le decisioni.

Ma come si prendono questi dati? e come si effettuano queste decisioni?

Ebbene, quello che essenzialmente si fa è provare a “predire” il futuro sulla base dei dati a disposizione sul momento e/o quelli passati.

Cosa si può dire invece della quantità di dati a disposizione?

Essenzialmente avere tanti dati è utile tanto quanto non averne, infatti per sfruttarli appieno servirebbero tecnologie molto specifiche che non sempre si hanno a disposizione.

Quindi, dal momento che i normali database non bastano, servono nuove e strumenti.

## Esempio

Immaginando di avere una catena di supermercati diffusa in ogni provincia e regione, quello che si vuole fare è distribuire i prodotti al meglio in base all’andamento delle vendite.

Come si fa a fare tutto questo?

Si possono utilizzare le normali query, queste però non bastano dato che si focalizzano solo su una piccola parte della conoscenza, inoltre costano troppo.

Un secondo problema riguarda i database stessi: essi immagazzinano i dati in modo sparso, bisogna quindi trovare un modo eterogeneo per gestirli e integrarli con altri in modo indipendente dalla forma, che sia un file Excel, un file di testo, eccetera.

Come soluzione a questi problemi, si utilizzano le query OLAP (OnLine Analytical Processes), esse però sono complesse da realizzare e utilizzano molta potenza computazionale, quindi ci vuole tempo per ottenere un risultato, anche in termini di giorni.

In un caso reale si effettuano più query di questo tipo, la prima di queste comincia l’indagine mentre le altre, sulla base del risultato di quella precedente, servono per avere informazioni più specifiche e quindi prendere decisioni in modo sensato!

Nel caso dell’esempio precedente, si può iniziare l’indagine facendo una query su tutti i supermercati e con le altre raffinare la conoscenza già a disposizione andando a vedere per distretti, per mese, per regione, eccetera.

Di conseguenza, perdere tanto tempo per fare queste query non ha senso, bisogna fare in modo che tutto avvenga in modo più interattivo e utilizzabile anche dai meno esperto dato che non tutti sono informatici

Proprio per questi motivi, l’utilizzo diretto del linguaggio SQL non va bene.

# Cos’è il data warehousing?

Il data warehousing è un insieme di metodi, tecnologie e strumenti utili per chi lavora con la conoscenza, tra questi ci sono anche strumenti per l’analisi dei dati per attuare e migliorare tutto il patrimonio informativo a disposizione.

Cosa si vuole ottenere?

Quel che si vuole essenzialmente ottenere è un astrazione sui dati, ciò serve per ottenere:

* flessibilità, in quanto non è sempre possibile sapere quali informazioni bisogna estrarre;
* integrazione, quindi i dati devono essere uniformati in base al modello;
* accessibilità agli utenti che non sono informatici;
* consistenza, cioè che i dati non devono contraddirsi;
* multidimensionalità in modo da sviluppare query OLAP per offrire all’utente una visione user-friendly e facilmente manipolabile;
* bisogna inoltre rappresentare l’evoluzione temporale, cosa che i database normali non possono fare.

## Aspetti importanti per i data warehouse

* Un datawarehouse non crea nuovi dati, bensì riorganizza e sintetizza quelli già a disposizione;
* In questo contesto la prospettiva storia (a volte non presente nei dati operazionali) è importante dato che da essa si prendono le decisioni;
* i dati del datawarehouse sono read-only, la scrittura dei nuovi dati non avviene infatti per mano di un utente ma di un software ETL a partire dai dati sorgente. Quello che essenzialmente cambia riguarda la consistenza.

## Cosa si può dire riguardo alla consistenza?

La consistenza nei database è garantita dalla terza regola della forma normale: ogni attributo non-chiave deve dipendere solo e solamente dalla chiave.

Ciò permette l’inserimento manuale da parte degli utenti senza rendere inconsistente il database, ma cosa si fa quando si ha bisogno di ridondanza?

In tal caso l’inserimento manuale viene meno, quindi si utilizzano software che modificano tutto il necessario nel modo corretto.

Coi dati read-only, la ridondanza permette maggior consistenza e ottimizzazioni per quanto riguarda la raccolta dei dati.

## Architetture di un data warehouse

### Architettura a un livello

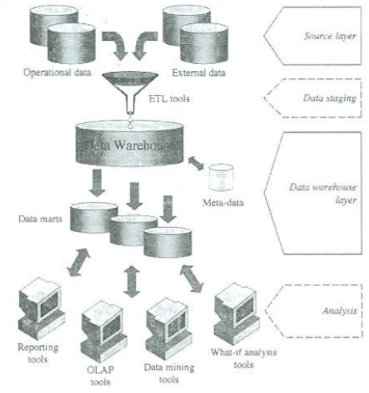
Questa non è un’architettura vera e propria per i datawarehouse dato che non soddisfa certi principii.

Alla base ci sono le sorgenti di dati operazionali mentre in mezzo c’è il warehouse, una visione virtuale dei dati fatta attraverso un software detto middleware, su di esso si applicano tutti gli strumenti in base alle esigenze.

Nonostante la semplicità e la ridondanza minima, in questa architettura non è presente una separazione tra l’elaborazione OLAP analitica e quella OLTP transazionale, generando un degrado delle performance.

Inoltre non è possibile inserire un livello di storicizzazione superiore a quello dei dati operazionali.

## Architettura a due livelli

L’architettura a due livelli migliora la precedente aggiungendo una separazione tra i dati utilizzati dalle query OLTP e OLAP.

Precisamente sono presenti quattro livelli:

* Il livello delle sorgenti indica i dati generati dall’ambiente di produzione;
* Il livello di alimentazione utilizza gli strumenti ETL per ripulire i dati dalle inconsistenze e completarli nelle parti mancanti. A questo livello i dati vengono inoltre arricchiti con dei metadati.
* Il livello di warehouse inserisce i dati nel datawarehouse e dei data mart, quest’ ultimi sono sottoinsiemi dei dati del datawarehouse. Da ciò è possibile generare delle viste sul datawarehouse.
* A livello di analisi, i dati dei datawarehouse e dei data mart vengono consultati attraverso degli strumenti OLAP o di reportistica.

Il vantaggio di questa architettura riguarda la separazione tra le query analitiche e transazionali, permettendo maggior efficienza e la storicizzazione.

L’unico svantaggio invece è la ridondanza dai dati, questi infatti vengono memorizzati sia nelle sorgenti operazionali, sia nel datawarehouse.

## Architettura a tre livelli

L’architettura a tre livelli è essenzialmente un’architettura a due livelli che ci ha creduto fino in fondo! Qui viene aggiunto il livello dei dati riconciliati in cui vengono vengono inseriti i dati risultanti dal processo di integrazione e pulitura.

La fase di integrazione non è esclusiva di questa architettura, infatti questa viene anche eseguita nell’architettura a due livelli, solo che non viene memorizzato il risultato.

Al costo di maggior ridondanza, questa architettura permette una maggior separazione, estraendo e integrando i dati operazionali e poi memorizzandoli.

# Come si consultano i dati?

La consultazione dei dati avviene attraverso i fatti, essi sono il nucleo della rappresentazione e la loro aggregazione è importante per l’analisi dell’utente.

Essenzialmente un fatto è qualcosa che accade nel dominio e ciò, come già detto, è utile per l’analisi.

Un evento è l’avvenimento di un fatto, descrivibile in modo quantitativo da una misura.

Eventi importanti sono quelli primari, da questi si possono ottenere quelli secondari sintetizzando i dati e perdendo alcuni dettagli.

Una dimensione è una prospettiva di analisi su cui è possibile raggruppare e identificare gli eventi, studiandoli solo negli aspetti che interessano.

Il tempo in questo contesto è sempre presente mentre altri variano in base al caso d’uso.

Ogni dimensione può essere vista in diversi livelli di dettaglio, da ciò deriva la cosiddetta gerarchia di dimensioni,la quale permette di definire i vari livelli di aggregazione degli eventi.

## Cos’è la visione multidimensionale?

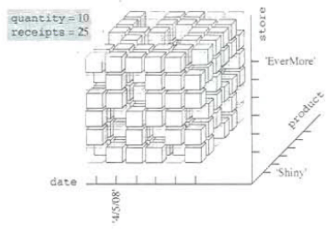
La visione multidimensionale è l’aggregazione dei dati lungo le gerarchie delle dimensioni.

Prendendo un ipercubo come esempio, si considerano tre dimensioni (data, negozio e prodotto), ogni fatto è una vendita ottenuta incrociando i valori e ogni evento primario contiene le sue misure.

Su questo esempio si fanno delle query analitiche aggregando i dati lungo le gerarchie, creando così la visione multidimensionale.

Dalle vendite per giorno, si passa a quelle per mese e dai prodotte alle tipologie, il risultato ottenuto è una serie di eventi secondari i cui valori sono aggregazione di eventi primari facendo la somma, la media, eccetera delle singole misure.

### Operazioni

* Roll up: è l’operazione di aggregazione e analizza gli eventi cambiando i livelli di aggregazione, precisamente raggruppando gli eventi primari in secondari.
* Drill down: è l’operazione inversa a quella di roll-up, infatti indica la disgregazione dei dati, andando nello specifico. Questa operazione ha però dei vincoli, infatti non può essere utilizzata con gli eventi primari;
* Selezione: è un’operazione che fissa un valore per ogni dimensione andando a prendere un evento primario specifico.
* Dice: è un’operazione simile alla selezione in cui si va a selezionare un sottoinsieme di eventi.
* Slice: operazione che sopprime una dimensione fissando un valore,andando quando a prendere solo un dato;
* Drill-Across: Analogo del join, permette di unire più fatti differenti, tuttavia non è molto utilizzata se non in casi specifici;
* drill-through: consiste nel passare dai dati aggregati multidimensionali del datawarehouse a quelli operazionali a livello di sorgenti o riconciliato;
* pivoting: cambia le modalità di presentazione per analizzare le informazioni su punti di vista differenti.

Tutte queste operazioni sono basate su OLAP e sono più intuitive delle normali query SQL, inoltre è possibile inserire supporti grafici, eccetera.

## Come realizzare l’approccio?

Esistono due modi per realizzare questo approccio:

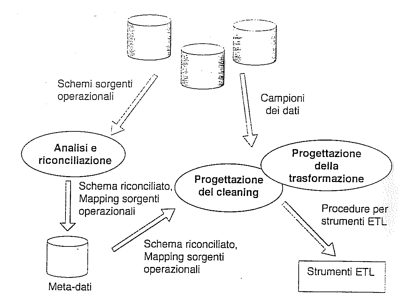
* Relational OLAP (ROLAP): un metodo in cui l’utente vede i dati multidimensionali, il tutto però viene memorizzato utilizzato database relazionali. Ci sarà quindi un middleware che permetta questa conversione. ROLAP ha il vantaggio di essere ben documentato grazie alla sua base relazionale, tuttavia ciò è anche uno svantaggio dato che possono capitare dei join di tabelle di grandi dimensioni. A fronte di questo problema, è possibile denormalizzare i dati per migliorare le prestazioni, inserendo della ridondanza;
* Multidimensional OLAP (MOLAP): il datawarehouse è già costruito utilizzando strutture dati multidimensionali, la comunicazione con l’utente avviene di conseguenza in modo diretto. Questo approccio è più efficiente ma allo stesso tempo problematico con certe dimensioni dei dati.
* Approcci ibridi.

## Realizzazione di un datawarehouse

Prima della realizzazione vera e propria, si parte facendo due fasi di bottom-up:

* una sul data mart di interesse, utile per ottenere un prototipo rapido;
* l’altra sulle sorgenti di dati operazionali.

Queste due fasi producono lo schema riconciliato, da questo si parte per la realizzazione vera e propria:

* la progettazione concettuale prende questo schema e i requisiti utente e produce uno schema di fatto;
* Dando lo schema di fatto in pasto alla progettazione logica, si ottiene una rappresentazione logica dello schema;
* Da quest’ultima si fa la progettazione fisica.

# Riconciliazione

La riconciliazione è la fase in cui appunto si riconciliano i dati. Questa fase è in ogni caso da fare, tanto che nelle architetture a tre livelli è pure esplicita.

La riconciliazione si fa nel seguente modo:

* si analizzano le sorgenti, da qui viene ricavato lo schema riconciliano e un mapping delle sorgenti operazionali, i cosiddetti metadati. Questa fase viene effettuata sugli schemi dei dati e non sul loro contenuto dal momento che non si ha sempre a disposizione modelli strutturati come quello ER.
* I metadati vengono poi utilizzati nella progettazione del cleaning e della trasformazione, due procedure utili per gli strumenti ETL in modo da trattarli a nostra discrezione.

## Strumenti ETL

Come si può vedere dalla foto a destra, in alto ci sono i dati operazionali, il loro problema riguarda l’eterogeneità dato che possono avere strutture differenti.

Attraverso gli strumenti del datawarehouse, si estraggono i dati, questi vengono messi in aree di lavoro in cui vengono puliti e trasformati.

### Estrazione

La fase di estrazione può essere fatta in due modi:

* con l’estrazione statica, si popola il datawarehouse la prima volta inserendo i dati operazionali rilevanti;
* con l’estrazione incrementale, si aggiorna il datawarehouse con le sole modifiche effettuate sulle sorgenti, per farlo si sfruttano dei log e dei trigger nel DBMS.

### Pulitura

La fase di pulitura corregge e/o cambia i dati cercando di eliminare problemi ed errori, ciò è molto importante in quanto la qualità dei dati è un fattore che conta nei datawarehouse.

Possibili errori che si possono trovare nella fase di pulitura sono duplicazioni, inconsistenza, mancanza di dati, utilizzo sbagliato di un campo, valori impossibili errati.

Molte aziende investono molto sulla pulitura dei dati utilizzando regole specifiche e vocabolari per gestire errori di battitura, sinonimi e omonimi, con le regole è inoltre possible capire quali valori derivano da altri e quindi verificarne la correttezza.

### Trasformazione

La trasformazione invece è un cambiamento vero e proprio del formato dei dati, precisamente è presente:

* una fase di conversione e normalizzazione che convertono i dati operando a livello di memorizzazione e di unificazione sulle unità di misura;
* la fase di matching stabilisce quali campi sono tra loro equivalenti tra le varie sorgenti.

Ultima ma non meno importante è la fase di caricamento dei dati nel database riconciliato, questo può essere:

* Update: Vengono solamente inseriti i dati che hanno subito dei cambiamenti senza alterare gli altri. Ciò avviene di solito insieme a un’estrazione incrementale dalle sorgenti;
* Refresh: I dati nel data warehouse vengono riscritti in modo integrale (anche quelli modificati), ciò avviene solitamente insieme a un’estrazione statica nel primo caricamento.

## Come si riconciliano i dati?

Per effettuare questa fase bisogna decidere a monte come riconciliare i dati e quindi come organizzarli.

Dalle sorgenti bisogna costruire lo schema logico di riconciliazione, qui devono esserci regole ben definite per descrivere ogni corrispondenza.

### Ricognizione e normalizzazione

La fase di ricognizione effettua un esame approfondito degli schemi locali per comprendere il dominio applicativo e il formato dei dati, da ciò deriva uno schema di integrazione, utile per ottenere lo schema finale da inserire nei metadati.

E’ importante in questa fase vedere lo schema e le varie informazioni, nei casi reali infatti queste possono essere strutturate diversamente.

La fase di normalizzazione ha il compito di correggere lo schema ricavato in precedenza per poter modellare il dominio applicativo in modo più accurato.

Queste due fasi vengono eseguite per ogni sorgente operazionale, inoltre è importante ascoltare l’esperto del domino per verificare la completezza degli schemi ottenuti.

Una volta ricavati tutti gli schemi, si uniscono per ottenere lo schema finale, qui bisogna stare attenti dato che potrebbero esserci dati impliciti come le dipendenze funzionali.

### Come integrare tutto questo?

Prima di tutto bisogna capire i concetti comuni delle sorgenti, non basta però controllare il nome dal momento che ci possono essere sinonimi oppure omonimi, ciò non è banale e per questo motivo questa fase viene fatta a mano o sfruttando intelligenze artificiali.

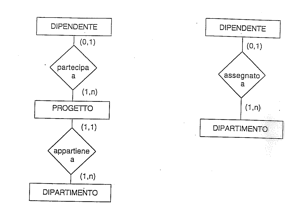
Dopo aver stabilito cos’è comune, si va a vedere come questi concetti sono stati descritti, anche qui le descrizioni sono differenti e possono essere:

* identiche: i due concetti sono uguali, quindi si utilizza lo stesso schema;
* equivalenti: i due concetti hanno descrizioni differenti ma c’è corrispondenza uno a uno tra i valori;
* Comparabili: i due concetti hanno descrizioni non equivalenti ma nemmeno in contrasto;
* Incomparabili: è presente incoerenza tra le descrizioni, non si può fare niente.

Ogni volta che due concetti non sono identici, si verifica un conflitto.

### Esempi di conflitti

Nella foto a destra c’è una caso di concetti equivalenti, dipartimento ed edificio sono infatti sinonimi.

La seconda foto presenta un caso molto simile al precedente: anche qui i concetti sono equivalenti solo che, al posto dei sinonimi, ci sono degli omonimi, in questo caso cliente con acquirente e credito con ordine.

Nel terzo e ultimo esempio c’è un caso di compatibilità: tra i due modelli ER è infatti presente un’entità in più (Progetti) che non permette un mapping uno a uno tra i dati.

## Fasi dell’integrazione

### Preintegrazione

Prima dell’integrazione vera e propria, si fa una fase di preintegrazione, lo scopo è quello di analizzare gli schemi per capire quali porzioni integrare e quale strategia utilizzare.

Le possibili strategia sono le seguenti:

* la strategia binaria considera due sorgenti alla volta, qui si può decidere se effettuare il processo a scala o bilanciato, come si vede in figura;
* la strategia ennaria considera n>2 sorgenti alla volta, questa permette di risolvere i conflitti fin da subito. Questa strategia può essere fatta in un solo passo oppure iterativamente.

Non esiste una strategia corretta, si sceglie in base a vari fattori come il numero di sorgenti e le loro correlazioni.

### Comparazione degli schemi

La fase di comparazione degli schemi identifica e analizza gli schemi per trovare correlazioni tra concetti. Precisamente, questa fase ha il compito di trovare:

* i conflitti di eterogeneità dei modelli;
* i conflitti dei nomi, quindi sinonimi e omonimi);
* i conflitti semantici, in questo caso gli schemi modello lo stesso concetto ma a livelli di dettaglio differenti;
* i conflitti strutturali, quindi il tipo, le dipendenze e tutto ciò riguardante il come è stato modellato un dato concetto.

### Allineamento degli schemi

Nella fase di allineamento degli schemi, si cercano di risolvere i conflitti trovati in precedenza, come si fa? Precisamente si utilizzano delle trasformazioni, modificando i nomi e/o i tipo degli attributi oppure agendo sulle dipendenze funzionali avvantaggiando gli schemi ritenuti più importanti.

In questa fase bisogna inoltre creare un mapping tra gli schemi sorgente e quello riconciliato in cui specificare le corrispondenze.

### Fusione e ristrutturazione degli schemi

La fase di fusione e ristrutturazione degli schemi ha lo scopo di mettere insieme tutti gli schemi per formarne uno unico, per fare ciò si uniscono tutti i concetti comuni, questi infatti sono legati a tutti gli altri concetti rimanenti.

Dopodichè si effettuano delle modifiche per migliorare le seguenti proprietà:

* completezza, cioè evidenziare le proprietà inter-schema;
* minimalità, quindi evitare la ridondanza tra i concetti;
* leggibilità, ovvero migliorare lo schema per facilitare le successive fasi di progettazione.

Non esiste una tecnica univoco per effettuare questa fase, la più frequente è Global As View

## Global as View

L’idea di questa tecnica consiste nel considerare il database riconciliato come una vista delle varie sorgenti da integrare. Precisamente, ogni concetto dello schema globale deve essere associato a una vista il cui significato è definito dallo schema sorgente.

Esempio: data la foto a destra, CL1 e CL2 indicano i database locali mentre CLIENTE e ORDINI sono il database riconciliato.

Ogni database locale è strutturato in mondo differente, quindi possono essere o meno informazioni comuni.

Quello che si fa è creare delle viste facendo dei join sulle chiavi e prendere gli attributi che interessano.

L’approccio Global as View permette di sostituire ogni concetto dello schema globale con la rispettiva vista.

Un approccio duale è Local as View, in essa non vengono definite delle viste ma si esprime tutto in termini di concetti degli schemi locali.

# Come si modella un datawarehouse?

Per progettare un datawarehouse c’è bisogno di un’astrazione dei dati ad alto livello per poter gestire i dati con più semplicità, per farlo però servono strumenti dello stesso tipo. Discutere con gli esperti del dominio è inoltre importante per ottenere una visione dei dati il più user-friendly possibile per la progettazione concettuale.

Si può utilizzare il modello ER? Sì, tuttavia la visione user-friendly e multidimensionale non sono supportate, cosa che si cerca di mantenere nei datawarehouse.

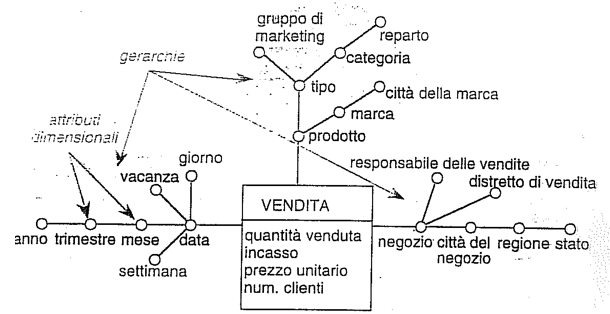
La progettazione di un datawarehouse non ha uno standard, uno dei tanti modi per farlo è Il Dimensional Fact Model (DFM).

Questa tecnica è molto utilizzata per i seguenti motivi:

* fornisce un supporto user-friendly tra progettisti e utenti;
* supporta una metodologia semiautomatica per passare dal modello delle sorgenti a quello concettuale;
* passaggio quasi automatico da modello concettuale a logico;
* rende l’ambiente intuitivo per le interrogazioni.

## Dimensional Fact Model (DFM)

Il DFM si basa sui seguenti concetti:

* i fatti, cioè i concetti di interesse per la decisione, essi sono dipendenti dal dominio e rappresentano aspetti statici e dinamici;
* le misure, numeri che descrivono una quantità relativa a un fatto;
* le dimensioni, ovvero le proprietà con dominio finito di un fatto che ne descrive la coordinata di analisi, esse indicano la granularità minima di rappresentazione. Una dimensione importante è il tempo ed è utile per la storicizzazione;
* gli eventi primari, delle occorrenze dei fatti individuati da un valore per ogni dimensione. A ogni evento primario è associata una misura e indica la granularità dei fatti, ovvero il loro grado di precisione.

Un DFM può essere visto come un’associazione N a N tra le dimensioni, ogni fatto è rappresentato con una tabella aventi le misure che servono mentre ogni attributo è una dimensione.

## Dimensioni, attributi e gerarchie

Ogni dimensione ha differenti livelli ed è descritta da attributi dimensionali organizzati in una gerarchia.

La gerarchia dimensionale è un albero direzionato i cui nodi sono gli attributi dimensionali, ogni arco indica una corrispondenza N a 1 tra attributi, le dipendenze funzionali nel modello ER, inoltre vale la proprietà transitiva tra attributi.

L’importanza delle gerarchie è data dal fatto che definiscono il livello di granularità e quindi come aggregare e disaggregare gli eventi primari, ciò è molto utile nelle query OLAP.

## Eventi secondari

Oltre agli eventi primari, esistono anche gli eventi secondari, in questi ultimi è presente meno dettaglio e sono utili per avere una visione dei dati più generica. Per ottenere un evento secondario, si annidano più eventi secondari, aggregando opportunamente i valori.

Il modo in cui gli eventi primari possono essere aggregati è dettato dalle gerarchie dimensionali, gli attributi infatti corrispondono a granularità crescenti.

DFM è superiore ai modelli ER in quanto permette la rappresentazione di eventi secondari in modo semplice e comprensibile, ER infatti non enfatizza e i livelli di rappresentazione, quindi non è user-friendly.

## Costrutti avanzati

### Attributi descrittivi

Gli attributi descrittivi sono particolari attributi che contengono informazioni aggiuntive riguardanti gli attributi dimensionali.

Proprio per questo motivo, questi attributi non vengono utilizzati nell’aggregazione (anche perchè il loro dominio è continuo).

Gli attributi descrittivi indicano una dipendenza 1 a 1 con l’attributo dimensionale a cui fanno riferimento.

Gli attributi descrittivi sono foglie delle gerarchie dimensionali e possono anche essere connesse direttamente a un fatto, descrivendone gli eventi primari.

In DFM questi attributi vengono indicato con il nome sottolineato.

### Archi opzionali

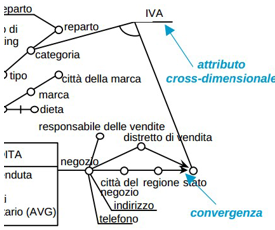
Gli archi opzionali indicano dipendenze opzionali e corrispondono ad associazioni 0-N, essi vengono utilizzati quando l’associazione nello schema non è definita per un dato sottoinsieme di eventi.

Proprio per questo motivo, qui entra in gioco il concetto di copertura, cioè quanti valori sono associati a ogni attributo, essa può essere:

* totale quando ogni valore di un attributo è almeno associato ad almeno un attributo dei figli;
* parziale nel caso in cui la precedente condizione non valga.
* esclusiva se ogni valore di un attributo è associato a un solo figlio;
* sovrapposta se non è esclusiva.

In DFM questi archi si indicano con un trattino.

### Convergenze tra archi

Le convergenze tra archi indicano che un attributo di una dimensione è raggiungibile da più attributi della stessa dimensione a granularità più fine, ciò è dovuto al fatto che sono presenti degli archi che permettono ciò.

In questo contesto è importante evitare i cicli, di conseguenza occorre direzionare le frecce.

### Attributi cross-dimensionali

Un attributo cross-dimensionale è un attributo particolare avente dipendenze multiple su differenti dimensioni. un esempio è l’IVA che dipende dal prodotto, e dallo stato.

* In DFM questi attributi si indicano intersecando delle linea che passano per gli attributi a cui fanno riferimento.

### Gerarchie condivise

Le gerarchie condivise sono particolari gerarchie in cui, da un dato attributo in avanti, tutti i valori vengono condivisi, essi si identificano in DFM utilizzando un doppio cerchio.

### Archi multipli

Gli archi multipli sono archi utilizzati per esprimere dipendenze non funzionali, cioè le associazioni N a N tra attributi.

### Gerarchie incomplete

Le gerarchie incomplete sono gerarchie in cui sono assenti un o più valori nei vari livelli di aggregazione, esse si differenziano dalle normali gerarchie in quanto utilizzano anche archi opzionali.

### Gerarchie ricorsive

Una gerarchia è detta ricorsiva quando è presente un arco che parte da un attributi e rientra su sé stesso o un suo antenato.

## Dinamicità degli attributi

Anche gli attributi possono essere dinamici (e di conseguenza anche le gerarchie dimensionali).

Cosa succede se si cambia un valore a un attributo?

Ci sono diversi modi per gestire questa situazione, ognuno coi propri costi e dimensioni:

* Oggi per ieri: si vede la realtà con gli occhi di oggi, quindi tutte le tuple passate valgono coi dati odierni;
* Ieri per oggi: il ragionamento è uguale a quello precedente solo che i dati che valgono sono quelli passati;
* Oggi e ieri: indica tutto ciò che non è cambiato, qui infatti non bisogna gestire i cambiamenti;
* Oggi o ieri: si considerano i fatti tenendo conto della storia dei valori e attribuendo misure a quelli temporaneamente corrispondenti, di conseguenza le cose giuste valgono al momento giusto.

# Progettazione concettuale

Il processo di definizione del database riconciliato con DFT è la progettazione concettuale, in questa fase:

* vengono definiti i fatti di interesse;
* per ogni fatto, si costruisce un albero di attributi e su di esso si effettuano eventuali potature e innesti. In questa fase vengono anche definite le dimensioni e le misure, oltre che la creazione degli schemi di fatto.

Alcune sottofasi sono semi-automatiche e richiedono gli esperti del dominio, altre invece vogliono gli schemi delle sorgenti.

L’albero degli attributi è una struttura i cui nodi sono gli attributi, la radice è l’identificatore del fatto e gli archi indicano le varie dipendenze, ciò significa che l’albero rappresenta le dipendenze funzionali degli attributi rispetto al fatto.

## Come si definisce un fatto?

Considerando il modello relazionale, un fatto è un’entità oppure una relazione N-aria tra entità, in quest’ultimo caso è necessario reificarla e convertirla in un’entità, sostituendo tutti i rami con associazioni 1 a N binarie.

## Come si costruisce un albero?

All’inizio si rappresenta ogni fatto con un’entità, facendo reificazioni in presenza di relazioni.

Dopodichè, si prende un’entità e il vertice corrente v e:

* Si genera la radice;
* Per ogni attributo diverso da quello attuale, si aggiunge l’attributo come figlio;
* Per ogni entità G connessa a E da una relazione R con cardinalità massima a 1, si aggiunge ogni attributo di R come figlio;
* Si genera un nodo vertice dall’entità G e si aggiunge come figlio al nodo corrente;
* Si fa una chiamata ricorsiva su G e il nodo appena generato.

Questo algoritmo è tuttavia approssimato in quanto alcuni casi si possono risolvere soltanto manualmente. Un esempio sono i cicli di dipendenze funzionali, queste creano un albero infinito e, per risolverlo, si considera un numero finito di iterazioni.

Questa non è l’unica parte non gestita dall’algoritmo, quindi:

* le associazioni N a N vengono convertite in associazioni binarie;
* l’opzionalità viene segnata sugli archi;
* la convergenza viene risolta considerando i soli attributi cross-dimensionali;
* le gerarchie di specializzazione vengono trattate come associazioni 1 a 1 opzionali, volendo però si può gestire aggiungendo un attributo discriminatore e tutti gli attributi della sottoentità come figli diretti della chiave;
* gli attributi composti può essere gestito come figlio diretto di una chiave a cui sono associati gli attributi semplici, in pratica viene trattato quasi come un’entità.

## Come si gestiscono le potature e gli innesti?

Potature ed innesti sono operazioni che modificano l’albero degli attributi, per effettuarle è opportuno consultare l’esperto del dominio per rimuovere gli attributi che non interessano.

La rimozione degli attributi comporta una diminuzione della granularità degli eventi primari del datawarehouse:

* L’operazione di potatura ha lo scopo di rimuovere un intero sottoalbero di attributi;
* L’innesto invece elimina un nodo e connette eventuali figli al “nonno”, in quest’ultimo caso le dipendenze funzionali possono cambiare.

Un fattore importante riguarda l’innesto sugli attributi opzionali, qui infatti tutti i figli ereditano l’opzionalità. Se invece si innesta un attributo sulla radice, si rende più grossolana la granularità del fatto.

Eccezionalmente sono anche possibili altre operazionazioni di modifica dell’albero come ad esempio l’aggiunta/rimozione di dipendenze funzionali.

Da quest’albero si determinano:

* le dimensioni, considerando gli attributi discreti che sono figli diretti della radice, la loro scelta è cruciale e determina la granularità degli eventi primari;
* le misure considerando gli attributi numerici che sono figli diretti della radice.

Lo schema di fatto viene generato prendendo principalmente conto di misure e dimensioni, nei casi avanzati si tiene anche conto di attributi cross-dimensionali, dipendenze non funzionali, ecc.

## Analisi dell’additività delle misure

In alcuni casi occorre verificare se la misura è additiva o meno rispetto alla dimensione a cui fa riferimento.

Con additività si intende la possibilità di utilizzare la somma come operazione di aggregazione, per capire se una misura lo è o meno, data una dimensione di e una misura m:

* per ogni dimensione diversa da di, si fissa un valore;
* si fissano k valori della misura m per la dimensione di;
* identificare com’è possibile aggregare le k misure in modo da avere una misura aggregata per di.

In generale esistono diversi tipi di misure associabili a differenti operatori di aggregazione:

* le misure di flusso si riferiscono a periodi al cui termine vengono valutate in modo cumulativo;
* le misure di livello vengono valutate in particolari istanti temporali;
* le misure unitarie vengono valutate come le precedenti solo che sono espresse in termini relativi.

# Analisi del volume dei dati e del carico di lavoro

Quest’analisi serve a ridimensionare il database e indicare quali query sono più importanti al fine di utilizzarle più spesso, ciò si fa sia a livello fisico, sia a livello logico. L’analisi del carico di lavoro nei data warehouse non si fa allo stesso modo dei database dato che i primi servono per il supporto alle decisioni.

La stima del carico di lavoro nei data warehouse è quindi più complessa e deve essere continuamente aggiornata perchè non vale una volta per tutte.

Ciò che si può fare per stimare l’analisi del carico è salvare i livelli di aggregazione più interessanti utilizzando delle viste.

## Linguaggi OLAP

I linguaggi OLAP permettono l’interrogazione dei dati in modo multidimensionale, qui le operazioni più importanti sono roll up e drill down. In queste query le informazioni di interesse non sono sparse, i fatti vengono appunto rappresentati con un’unica entità ciascuno e a livello logico sono già raggruppati.

Ciò che si fa è quindi raggruppare/ dividere i dati focalizzandosi sul livello di aggregazione.

L'interrogazione è un’espressione seguita da una misura o un attributo, un’espressione dimensionale è il nome di un fatto seguito da una o più clausole di aggregazione, ognuna di essere:

* un pattern di aggregazione;
* un pattern seguito da un predicato di selezione, equivalente al “where” nelle query SQL.

Il pattern indica il livello di aggregazione e può essere:

* primario: è richiesta la granularità di base;
* secondario: si utilizza la granularità degli eventi secondari.

| <query> ::= <espressione>,<nome\_misura> | <espressione>,<nome\_attributo>  <espressione> ::= <nome\_fatto> <clausola\_di\_aggregazione>\*  <clausola\_di\_aggregazione> ::= <pattern> | <pattern> ; <predicato\_selezione>  <pattern> ::= <nome\_attributo> | <pattern> , <nome\_attributo> |
| --- |

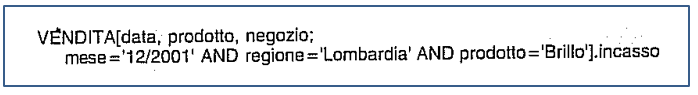
Nel template sopra:

* espressione indica le clausole di aggregazione e il fatto che si vuole considerare;
* la clausola di aggregazione è formata da un pattern, indicante il livello di aggregazione, e un predicato di selezione, quest’ultimo seleziona i dati che interessano nella query.

Un predicato di selezione si dice interno quando gli attributi hanno un aggregazione inferiore rispetto al livello specificato, se invece l’aggregazione è maggiore allore il predicato è esterno.

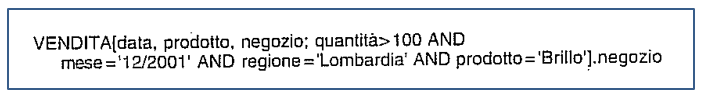
Per quanto riguarda i pattern, essi vengono detti primari quando il livello di aggregazione utilizzato è pari a quello base, altrimenti è detto secondario.

## Query primarie



Considerando la query sopra, si fissa la granularità di base, quindi si considerano le vendite giorno per giorno, prodotto per prodotto e negozio per negozio.

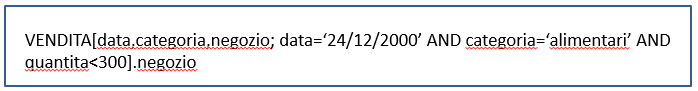
Tra tutti i dati aggregati, si prende l’incasso del prodotto Brillo in Lombardia nel mese di Dicembre 2001.



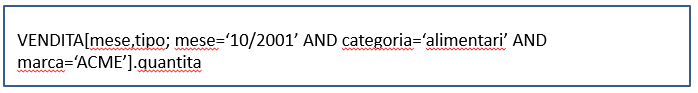
Prendendo ora in considerazione la query sopra, si aggrega allo stesso modo ma c’è una condizione in più: si prendono in considerazione tutti i negozi che hanno venduto più di 100 unità del prodotto specificato.

## Query secondarie

Le query secondarie non sono tanto diverse dalle precedenti, sono infatti aggregazioni di query primarie.



Nella query sopra, si considerano granularità che non sono quelle base.

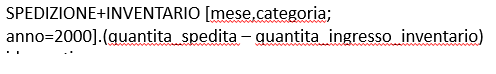


In questo secondo esempio invece sono presenti annidamenti, di tutto il risultato si prende il livello dell’inventario.

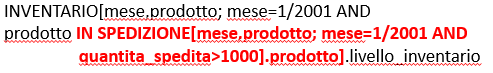
E’ anche possibile rappresentare queste query OLAP in modo grafico sviluppando interfacce interattive per facilitarne l’utilizzo anche ai non informatici.

## Come si esprime il drill across?

Le operazioni drill across sono quelle che mettono in relazioni due schemi di fatto,esse si possono esprimere come una query in cui si specificano più livelli di aggregazione:



Un altro metodo è il flow-across e si esprime attraversando in sequenza i vari schemi di fatto:



In entrambi i casi, gli schemi di fatto da confrontare devono essere tra loro compatibili, ciò ne permette la sovrapposizione.

Tutti questi metodo sono equivalenti, la differenza è che il secondo è più astratto.

## Interrogazioni composte

Le interrogazioni composte sono quelle in cui si specificano più livelli di aggregazione, esse sono nella seguente forma:

<fatti>[<aggregazione>;<where>].<select>



Nell’esempio sopra, si aggregano tutte le vendite per data, negozio e prodotto considerando tutto quelle che hanno venduto più di 50 unità, il risultato ottenuto viene aggregato per regioni e tipo e da essi si prende l’incasso.

Una volta ottenute le query più importanti, occorre verificarle andando a eseguirle al fine di capire se il risultato è quello che si vuole ottenere.

Nella fase di progettazione potrebbero esserci misure esplicite, implicite o derivate che vengono fuori dal carico di lavoro, altre misure invece potrebbero essere non previste.

Ciò che si vuole ottenere è un modello che soddisfa il carico di lavoro.

Purtroppo esistono anche query che non sono possibili per il datawarehouse, in tal caso non vengono considerate.

Per quanto riguarda il volume, nel caso degli eventi primari bisogna considerare la cardinalità delle dimensioni (cosa facilmente calcolabile). Negli eventi secondari invece questo calcolo è più complesso, di conseguenza si utilizzano approcci probabilistici.

# Rolap e Molap

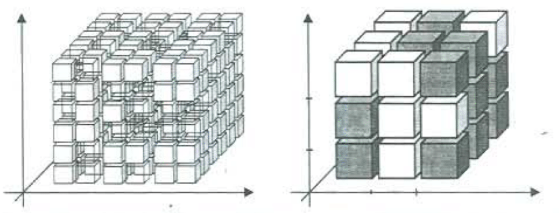
## Molap

L’approccio Molap permette di realizzare datawarehouse con la visione multidimensionale già implementata, per farlo si utilizzano vettori n-dimensionali.

Questa rappresentazione risulta più naturale per i dati, inoltre permette prestazione ottimali dato che le query OLAP vengono eseguite direttamente sulle strutture dati senza fare conversioni in SQL.

Questo approccio non è tuttavia dominante dal momento che non è uno standard, inoltre la maggior parte dei database in circolazione è relazionale, l’idea infatti è quella di cercare di mantenere questa caratteristica.

Un altro problema riguarda la sparsità dei dati, infatti solamente il 20% dei dati è effettivamente presente, ciò causa uno spreco di memoria.

A fronte di questo problema, esistono delle tecniche per arginarlo come ad esempio la compressione con appositi indici. La compressione è solitamente abbinata a tecniche di chunk in cui si dividono le “regioni” di dati in chunk e su di essi si effettuano tutte le operazioni per ricavare la sparsità.

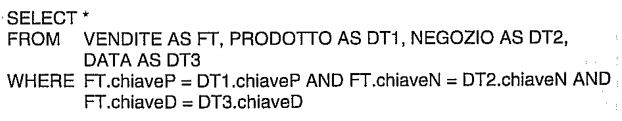
## Rolap

L’approccio Rolap, a differenza del precedente, non presenta buchi dato che ogni tupla ricopre solo una certa informazione, di conseguenza la sparsità dei dati viene gestita “automaticamente”, permettendo una miglior scalabilità.

Oltre a questo, esistono anche approccio intermedi per conciliare tutti i vantaggi.

## Star schema

La modellazione di un datawarehouse in Rolap utilizza uno star schema, una collezione di tabelle che descrivono un fatto sulle dimensioni, esso è composto da:

* una dimension table per ogni dimensione, ognuna descrive la dimensione con un insieme di attributi, inoltre sono caratterizzate da una chiave primaria, di solito surrogata per ridurre lo spazio nella fact table;
* una fact table contenente le chiavi di ogni dimensioni table, tutte queste formano la chiave primaria. La fact table contiene inoltre altri attributi, questi indicano le misure. Questa tabella contiene tutti gli eventi primari.

Il datawarehouse contiene uno star schema per ogni fatto. La visione multidimensionale si ottiene utilizzando una query che raggruppa tutte le informazioni, in questo contesto lo schema è coerente dato che si specifica un valore per ogni dimensione.

## Come si leggono le informazioni?

Dal momento che la fact table contiene gli eventi primari, considerandone uno si può notare tanta ridondanza di dati nelle dimension table, tuttavia ciò non sorprende dato che in queste tabelle c’è un attributo per ogni attributo della dimensione.

Le dimension table contengono inoltre le tutte le dipendenze funzionali rispetto alla chiave, ciò va contro la terza forma normale.

Questa violazione è un bene o no? Dipende dal contesto, nei database relazionali la terza forma normale garantisce la consistenza dei dati, una qualità essenziali dato che il database è accessibile agli utenti in scrittura.

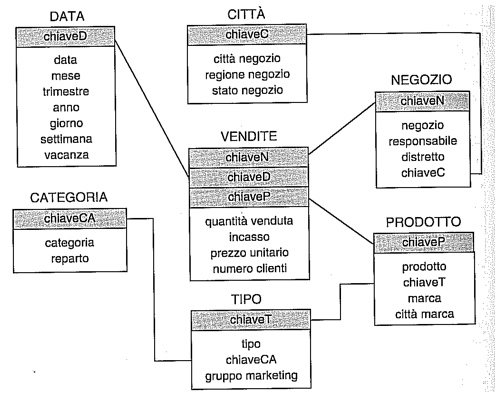
Nel contesto dei datawarehouse, l’utente non può modificare i dati, di conseguenza si può gestire la consistenza in automatico, inoltre risulta più semplice aggregare i dati.

La terza forma normale infatti ha il difetto di creare tantissime tabelle, l’aggregazione ha quindi bisogna di una query più complessa.

Lo star schema ha quindi il vantaggio di ridurre il numero join per il recupero delle informazioni, tutto ciò al costo di più spazio per la memorizzazione a causa delle dipendenze funzionali transitive.

## Schema snowflake

Lo schema snowflake è una generalizzazione di quello a stella in cui parte della ridondanza viene gestita, ciò permette di ottenere soluzioni migliori sugli eventi secondari dato che alcune dipendenze transitive vengono eliminate.

In questo schema, le dimension table possono essere:

* primarie quando la chiave è importata direttamente nella fact table;
* secondarie invece quando non è così e riferisce a un’altra dimension table.

Lo schema snowflake è ottenibile da uno star schema andando a spezzare le tabelle su alcune dipendenze, ciò non viene fatto casualmente ma in un certo punto, solitamente dove c’è molta ridondanza.

Lo schema snowflake ha come vantaggio quello di avere meno duplicazione e dimension table più piccole, permettendo inoltre l’aggregazione parziale negli eventi primari.

A fronte di ciò, lo schema snowflake è più complesso da aggregare completamente, ciò è dovuto a un numero maggiore di join da effettuare.

## Modellazione logica

Ci sono casi in cui è bene materializzare le viste per rendere tutto più efficiente, come si fa? Supponendo l’esempio a destra, la vista v1 è quella primaria e deve essere sempre materializzata, tutte le altre invece sono quelle secondarie.

Le frecce indicano quale vista è più dettagliata di un’altra, v1→ v2 ad esempio indica che v1 è più dettagliata di v2. Questa relazione non è tuttavia completa dal momento che due viste potrebbero non essere comparabili secondo la relazione.

Le viste secondarie servono a rispondere alle query in modo più efficiente senza il bisogno di aggregare i dati ogni volta, nel caso non si ha a disposizione una data vista, si può utilizzare quella più vicina alla richiesta.

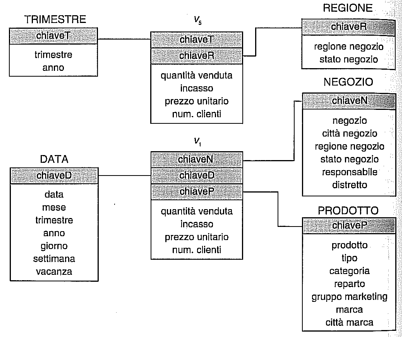
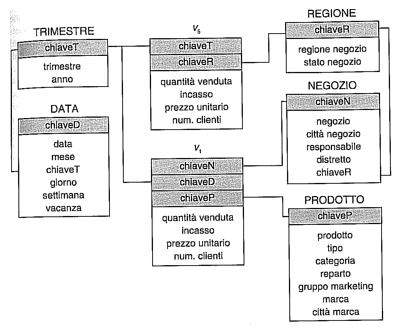
## Come si materializza una vista?

Una prima soluzione è memorizzare tutte le viste in un’unica fact table, utilizzando il NULL per gestire gli attributi non valorizzati nelle dimension table. Questo approccio è il più semplice ma aumenta la dimensione della fact table, inoltre le query con dati aggregati non hanno buone performance.

La seconda soluzione è quella di utilizzare una fact table per ogni vista, in tal caso bisogna scegliere come gestire le dimension table.

Quest’ultima soluzione permette di separare i dati per livelli di aggregazione, permettendo diverse possibilità per la memorizzazione delle dimension table.

## Gestione delle dimension table

* schema a costellazione: le dimension table rimangono quelle dello star schema e sono connesse alle fact table in base ai livelli di aggregazione. Questo approccio permette un accesso veloce sia alle fact table, tuttavia ciò viene fatto sia in quelle generali, sia nelle secondarie, risultando non rilevanti su alcuni attributi;
* Star schema multipli: Da ogni vista si materializzano le dimension table (oltre che le fact table) contenenti i soli attributi utili alla vista. Questo approccio ottimizza l’accesso sia alle fact table, sia alle dimension table al costo di una duplicazione parziale di queste ultime;
* snowflake delle dimension table: viene effettuato uno snowflake delle dimension table per ottenerne altrettante corrispondenti al livello di aggregazione delle viste. Ciò permette un accesso efficiente alle viste secondarie e uno meno efficiente per quelle primarie (facendo più join).

## Come si calcolano le aggregazioni?

Bisogna stare attenti al calcolo delle aggregazione dato che può portare a errori se non vengono considerate le caratteristiche degli operatori di aggregazione, questi possono essere:

* Distributivi: calcolano i dati aggregati da quelli parzialmente aggregati, tra questi ci sono la somma, il minimo e il massimo;
* Algebrici: richiedono misure di supporto per il calcolo di dati aggregati da altri aggregati in modo parziale, un esempio è la media che ha bisogno del conteggio;
* Olistici: non permettono il calcolo dei dati aggregati da quelli parziali se si hanno a disposizione un numero finito di misure di supporto, moda e mediana sono due esempi appartenenti a questa famiglia.

# Realizzazione di scenari temporali

Uno scenario temporale è una gerarchia dinamica con valori dinamici sugli attributi multidimensionali.

Considerando l’esempio nella foto, a destra ci sono i singoli eventi di vendita mentre a sinistra c’è l’evoluzione di ogni attributo multidimensionale.

Nella seconda situazione, il signor Rossi lascia la gestione del negozio DiTutto al signor Bianchi per diventare responsabile di DiTuttoDiPiù.

L’utilizzo delle gerarchie dinamiche ha come conseguenza l’utilizzo di più spazio, è quindi necessario valutare i singoli casi in cui impiegarle.

## Gerarchie dinamiche di tipo 1

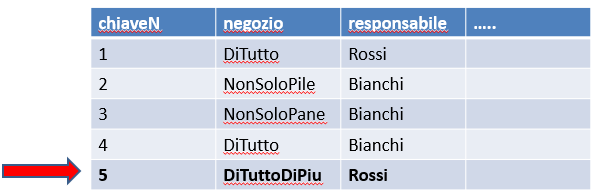
La gerarchia di tipo 1 permette di mantenere lo schema, che sia a stella o snowflake, ciò che cambia sono i valori nelle dimension table.

Considerando il precedente esempio, si modifica la tupla del negozio DiTutto mettendo Bianchi come responsabile e si aggiunge la tupla di DiTuttoDiPiù, di cui il signor Rossi è il gestore.

Questo tipo di gerarchia supporta solamente lo scenario “Oggi per ieri”, di conseguenza tutte le vendite del negozio DiTutto verranno attribuite al signor Bianchi, anche quelle precedenti al cambio di gestione.

Come si può risolvere questo problema?

## Gerarchie dinamiche di tipo 2

Le gerarchie dinamiche di tipo 2 permettono anche qui il mantenimento dello schema, la differenza rispetto a prima sta nell’aggiornamento della tupla, qui infatti ogni modifica comporta la creazione di una nuova tupla. Inoltre si fa in modo che solo la nuova tupla sia raggiungibile dall’ETL attraverso i metadati.

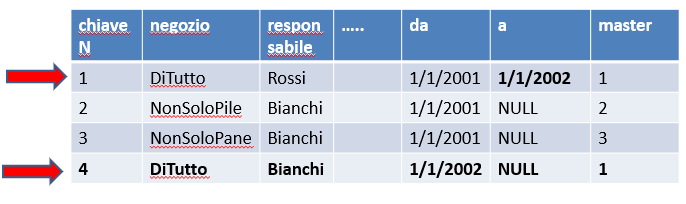
Nell’esempio dei negozi, viene creata una nuova tupla del negozio DiTutto avente il signor Bianchi come responsabile, infine si aggiunge la tuple del negozio DiTuttoDIPiù gestito dal signor Rossi.

Questa gerarchia dinamica permette il partizionamento degli eventi rispetto al tempo, senza però aggiungere nessuna marca temporale.

La gerarchia di tipo 2 supporta lo scenario “Oggi o ieri”, purtroppo non ne può supportare altri, “Ieri per oggi” ad esempio non è supportabile a causa della mancanza delle date.

## Gerarchie dinamiche di tipo 3

Nelle gerarchie dinamiche di tipo 3 c’è piena storicizzazione, com’è possibile? Ebbene, si modifica lo schema aggiungendo tre attributi:

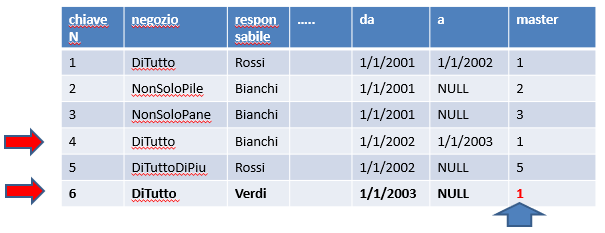
* from indica la data di inizio della validità della tupla;
* to invece è la data di fine, inizialmente a NULL;
* master contiene la chiave della tupla originale, quella che non è stata modificata. Nel caso in cui le tuple non hanno avuto modifiche, l’attributo master avrà la chiave della propria tupla come valore.

Nell’esempio delle vendite, come si gestisce il cambio di gestione?

* Si modifica la tuple del negozio DiTutto inserendo la data di fine nell’attributo to;
* Si aggiunge la nuova tupla del negozio DiTutto con Bianchi come responsabile, il suo campo master corrisponde all’id di quella vecchia.

L’aggiunta di nuove tuple viene gestita normalmente, senza fare ulteriori modifiche.

### Cosa succede se la gestione del negozio DiTutto passa a Verdi?

Come nel caso precedente, si aggiunge la data di fine alla vecchia tuple e si aggiunge quella nuova, quest’ultima però cos’ha nell’attributo master?

Ci può essere sia l’id della tupla appena modificata, sia quello originale.

Quest’ultimo però permette più efficienza senza perdere l’informazione dal momento che nel primo caso bisogna percorrere tutta la lista di cambiamenti.

La gerarchia di tipo 3 permette la gestione di ogni possibile scenario nei modi seguenti:

* Con “Oggi per ieri”, si individuano tutte le tuple senza data di fine si si propagano in valori su quelle con lo stesso master;
* Per lo scenario “Ieri per oggi”, si fissa una data e si fa betweening per trovare tutte le tuple in cui ricade, dopodichè si propagano i valori per tutte le tuple con lo stesso master;
* Con lo scenario “Oggi o ieri” si prendono tutte le informazioni su un dato, gestendo tutto come nelle gerarchie di tipo 2;
* “Oggi e Ieri” infine viene gestito prendendo tutte le tuple senza modifiche (quelle con master unico).

# Progettazione logica in Rolap

La progettazione logica è quella fase che permette di ricavare lo schema logico del data mart a partire da quello concettuale.

Una buona rappresentazione dei dati non può non essere sensibile al contesto, infatti ciò è di grande aiuto nelle diverse fasi della progettazione di un datawarehouse.

La transazione è inoltre importante dal momento che tutte le scritture sono concorrenti.

Il contesto dei datawarehouse è differente, le query infatti prendono grosse fette di dati, il caricamento avviene tramite l’ETL e di conseguenza non c’è bisogno di gestire la consistenza dell’utente.

Detto questo, è possibile concentrarsi su un recupero dati veloce al costo di una denormalizzazione degli stessi.

### Come si procede per creare lo schema logico da quello di fatto?

Lo schema di fatto può essere modellato in ER utilizzando uno star schema: le fact table contengono le misure e gli attributi descrittivi, inoltre per ogni gerarchia si genera una dimension table che contiene gli attributi.

Da questo schema è possibile passare allo snowflake per eliminare alcune ridondanze.

## Come vengono gestiti i costrutti avanzati?

### Attributi descrittivi

Gli attributi descrittivi sono quelli che vengono mantenuti nella gerarchia, non servono per l’aggregazione e hanno senso solo nel livello in cui vengono posti, un attributo di questo tipo di un evento primario non ha senso in un evento secondario e viceversa.

Nel caso in cui un attributo descrittivo è relativo a un evento primario, questo deve essere inserito nella fact table.

La gestione di questo tipo di attributo viene fatta inserendolo:

* nella dimension table al livello di aggregazione adeguato se è collegato a un attributo dimensionale;
* nella fact table se invece è connesso direttamente al fatto, questo non deve essere riportato nelle viste secondarie se materializzate.

### Attributi cross-dimensionali

Gli attributi cross-dimensionali nella modellazione concettuale sono rappresentati con due archi che convergono su un nodo, un’ attributo di questo tipo può essere ad esempio l’IVA, essa dipende sia dal prodotto, sia dallo stato.

Come si modellano? Ebbene, si arricchisce lo star schema inserendo una tabella per gestirlo, di solito si importano i valori direttamente dalle altre tabelle ma è anche possibile utilizzare dei surrogati, soluzione più adatta in caso di attributi dimensionali troppo spaziosi.

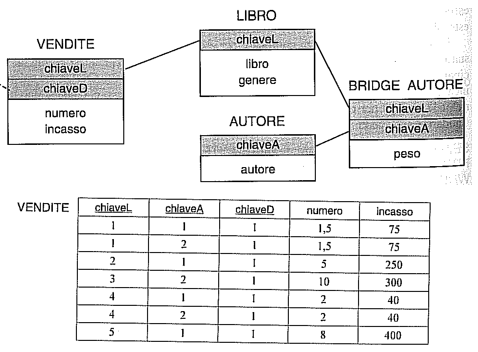
### Gerarchie condivise

Per quanto riguarda le gerarchie condivise, ci possono essere due casi:

* in caso di condivisione completa, si crea una nuova tabella e si connettono i vari attributi;
* se invece la condivisione è solo parziale, si creano tante tabelle finchè sono presenti differenze tra le gerarchie, alla fine si unisce tutto con una tabella. Un’altra soluzione è fare snowflake sul primo attributo condiviso e introdurre una nuova tabella comune alle dimension table.

Le chiavi di ogni tabella nuova corrispondono a quelle corrispondenti nelle dimension table primarie.

### Archi multipli

Gli archi multipli sono quelli che evidenziano le rappresentazioni molti a molti, causa principale della complessità nelle aggregazioni.

Un modo per trattare gli archi multipli consiste nell’utilizzo della bridge table, una tabella avente la combinazione degli attributi connessi all’arco come chiave primaria.

In queste tabelle è presente anche un attributo peso per effettuare una corretta aggregazione, le query possono quindi essere:

* Pesate se considerano questo attributo;
* di impatto quando non lo considerano esse permettono una quantità di risultati più grande;

Un altro metodo è il push down, esso consiste nel modellare l’arco multiplo creando una nuova dimension table in cui viene rappresentato insieme ai suoi discendenti.

Queste soluzione è necessaria nel caso in cui l’attributo multidimensionale non è direttamente connesso, è inoltre consigliata se la rappresentazione aumenta enormemente la cardinalità.

Da una parte questa soluzione ha il vantaggio di mantenere lo star schema e quindi ridurre il numero di join, dall’altra però si introduce una ridondanza nella fact table la quale ne aumenta la cardinalità e ne diminuisce le prestazioni.

Inoltre le query di impatto sono molto più complesse.

### Opzionalità

La gestione dell’opzionalità avviene inserendo un valore fittizio negli attributo vuoti. Questi sono supplementari dal momento che valori come il NULL hanno differenti significati, inoltre una chiave esterna non può esserlo.

### Gerarchie incomplete

La gestione delle gerarchie incomplete avviene inserendo valori fittizi negli attributi in cui non c’è un valore, ciò però è complicato da gestire dato che la mancanza di un valore non implica la mancanza degli altri nei livelli di aggregazione successivi.

Quindi che si fa? Ebbene, esistono tre soluzioni:

* Col bilanciamento per esclusione, si utilizza un segnaposto generico in ogni tupla di ogni dimension table;
* Si rimpiazza il valore mancante con quello dell’attributo precedente nella gerarchia dimensionale, questo metodo è detto bilanciamento verso il basso;
* Si applica il bilanciamento verso l’alto, uguale al precedente con la differenza che utilizza l’attributo successivo nella gerarchia.

### Gerarchie ricorsive

Dal momento che il numero di livelli non è definito a priori, le gerarchie risorsive non sono codificabili nello schema, come si gestiscono?

Una soluzione è modellarle attraverso un autoanello, permettendo la rappresentazioni di livelli illimitati, tuttavia molti DBMS non supportano le query ricorsive.

Un’altra soluzione è utilizzare una tabella di navigazione che modella le relazioni N a N tra una fact table e una dimension table, la sua cardinalità aumenta in modo esponenziale rispetto ai livelli nell’albero degli attributi.

### Dimensioni degeneri

Le dimensioni degeneri sono gerarchie dimensionali aventi un solo attributo dimensionale, la loro gestione avviene nei seguenti modi:

* si utilizza una dimension table per ogni dimensione degenere, cosa che ha senso se la cardinalità è limitata e le dimensioni degli attributi sono molto più grandi di quella della chiave;
* si memorizzano nelle fact table includendole nella chiave composta, soluzione consigliata se la cardinalità è elevata e i valori occupano poco spazio;
* si utilizza una junk table, un’unica tabella per tutte le dimensioni degeneri, ognuno col proprio id surrogato. Questa soluzione viene utilizzata solo quando la cardinalità dei valori non è troppo alta. Tutto ciò permette di evitare una fact table troppo grossa al costo di un join in più nelle query, inoltre non sono presenti dipendenze funzionali.

## Materializzazione delle viste secondarie

Per materializzare le viste secondarie, occorre sapere quali livelli di aggregazione sono più utili per ottimizzare il carico di lavoro. Ciò non è l’unico concetto da considerare, esistono infatti obiettivi specifici e tra loro anche in contrasto, l’obiettivo è trovare il miglior compromesso.

### Costi

I costi di valutazione del carico di lavoro sono quelli che riguardano perlopiù le interrogazioni, essi possono essere stimati come la somma pesata delle query in base all’importanza data dall’utente o alla frequenza.

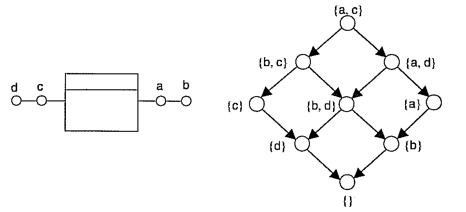
I costi di manutenzione delle viste indicano quelli riguardanti l’aggiornamento di queste ultime in modo da riflettere le modifiche sulle sorgenti operazionali, essi si determinano dal costo delle query utili per la propagazione degli aggiornamenti.

### Vincoli

Oltre agli obiettivi riguardanti il carico di lavoro, bisogna anche considerare i vincoli di sistema come ad esempio la memoria occupata da un data mart e il tempo di aggiornamento delle viste.

I vincoli dell’utente riguardano il tempo di risposta delle query e la frequenza di aggiornamento dei dati.

Per fare un esempio: considerando lo schema a destra, come si determina il miglior bilanciamento?

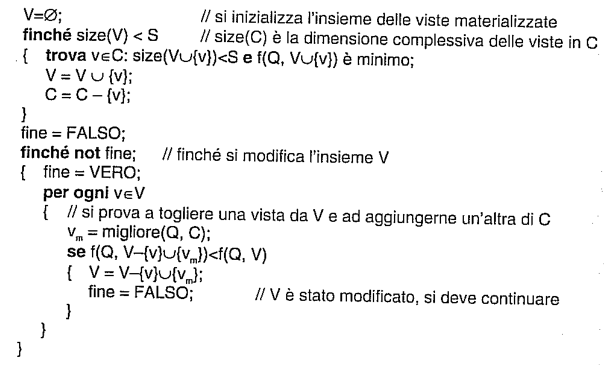
In base agli attributi, si determina una gerarchia di coppie, (a,c) in questo caso è la coppia più fine e si trova più in alto, al livello più basso si aggregano tutti i dati per ogni dimensione.

Dal grafo ottenuto si trova il miglior bilanciamento, per farlo si prende il carico di lavoro e si considerano i suoi pattern di aggregazione, da ciò si materializzano le viste.

Precisamente, data una query q, una vista v può risolvere q se e solo se v ha un pattern di aggregazione più fine di q, di conseguenza dalla vista v possono essere ricavate le misure di q.

Materializzando tutte le viste utili, il costo è minore ma si ha troppo spazio occupato e si perde troppo tempo, il ragionamento duale si ottiene memorizzando solo una vista.

## Algoritmo

L’algoritmo di materializzazione di viste definisce lo spazio dei livelli di aggregazione per ogni combinazione e per ogni gerarchia, nel caso della foto sopra, il livello più fine è dato da (a,c), il più generale invece è l'aggregazione totale senza scegliere attributi.

L’idea è che, dati due nodi, il massimo comune divisore è il nodo meno fine del primo ma più fine del secondo, come si ottiene? Essenzialmente si prendono tutti i nodi più fini e di questi si prende il meno fine, ciò permette di gestire le query in un unico livello di aggregazione.

L’algoritmo utilizza il concetto di vista candidata, cioè una vista per cui, dato uno schema F un carico Q:

* esiste una query q in Q avente lo stesso pattern di aggregazione;
* esistono due viste vi e vj tali che il massimo comune divisore dei loro pattern sia uguale al pattern della vista.

### Procedimento

Dato un insieme di viste candidate C sulla base di un carico di lavoro Q:

* si inseriscono in C tutte le viste del grafo corrispondenti a pattern di query nel carico di lavoro;
* si calcolano le viste rimanenti applicando il massimo comune divisore per ogni coppia di viste, interrompendosi solamente quando vengono generate viste già presenti in C.
* Dall’insieme C, si ricava un sottoinsieme di viste V che minimizza il costo di esecuzione rispettando dei vincoli di spazio S, all’inizio si inserisco tutte le viste che minimizzano la funzione di costo e non sforano S;
* Dopodichè, si tenta di migliorare V cercando di scambiare alcune viste candidate con altre non selezionate utilizzando l’hill climbing.

## Frammentazione di viste

Con frammentazione si intende la divisione di una tabella in più tabelle dette frammenti, ciò serve ad aumentare le prestazioni.

La frammentazione può essere:

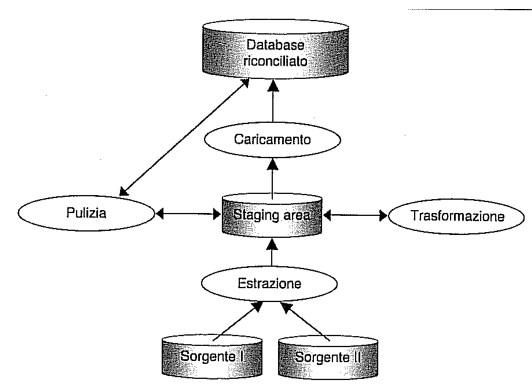
* orizzontale quando i frammenti contengono solo un sottoinsieme delle tuple, gli attributi di ogni frammento rimangono gli stessi. Questo metodo è utile per ridurre i tempi di esecuzione delle query, queste infatti operano su viste più piccole su cui sono già stati eliminate le tuple che non rispettano la selezione. Inoltre questo tipo di frammentazione non occupa spazio in più dal momento che le chiavi di ogni tupla vengono memorizzate una sola volta;
* verticale se i frammenti hanno un sottoinsieme degli attributi, chiave inclusa. Questo tipo di frammentazione è molto utile quando si conosce la quantità di lavoro in modo preciso, ciò infatti permette di avvantaggiare le query che utilizzano le misure di uno stesso frammento.

Entrambe le frammentazioni garantiscono le seguenti proprietà:

* consistenza: i frammenti devono avere pattern di aggregazioni presenti anche nella vista originale;
* completezza: ogni misura/tupla di ogni schema deve essere presente nello schema e viceversa;
* non ridondanza: non è possibile inserire una misura/tupla in più frammenti dello stesso pattern di aggregazione.

# Estrazione, trasformazione e caricamento

## Come si progettano gli ETL?

Prendendo come riferimento l’architettura a tre livelli, la progettazione consiste in tre passi:

* estrazione dei dati dalle sorgenti;
* trasformazione e pulizia;
* caricamento.

Prima di tutto però occorre alimentare il database riconciliato e da qui il datawarehouse.

### Alimentazione ed estrazione

Nella fase di alimentazione vengono definire le procedure per il caricamento dei dati operazionali nei data mart, in presenza di un livello riconciliato si distinguono due fasi:

* dalle sorgenti al livello riconciliato, la fase più complessa dal momento che le procedure agiscono sui dati nel modo definito in fase di integrazione, inoltre vengono definite le tecniche di pulizia dei dati;
* dal livello riconciliato al datawarehouse, qui le procedure definite servono a rendere conformi i dati riconciliati agli star schemi, inoltre si definiscono le tecniche di denormalizzazione, di inserimento delle chiavi surrogate e per il calcolo di dati derivati.

L’estrazione dei dati viene eseguita durante l'inizializzazione del livello riconciliato e ripetuta regolarmente per l’aggiornamento, essa può essere:

* statica, cioè quando viene effettuata in modo completo, utile per l’inizializzazione e deve essere implementata in ogni caso;
* incrementale, una scansione più immediata che riguarda solamente le novità.

L’estrazione incrementale può a sua volta essere:

* immediata, cioè quando i dati nuovi vengono messi subito nella staging area;
* ritardata, in questo caso i dati nuovi vengono messi in staging area in un tempo futuro.

#### Estrazione immediata

L’estrazione immediata può essere implementata tramite API, in questo caso il codice a disposizione si può modificare per effettuare ottimizzazioni, altrimenti potrebbe venir meno l’estrazione in sè.

L’estrazione basata sui trigger utilizza i trigger del DBMS per effettuare l’estrazione ogni volta che un dato viene modificato, questa tecnica non può essere sempre utilizzata dal momento che il DBMS dovrebbe controllare continuamente le transazione che innescano i trigger.

L’estrazione basata su log ha lo scopo di controllare i file log prodotti dal DBMS per verificare il caricamento di nuovi dati e/o la loro modifica, questa tecnica è consigliata quando quando l’estrazione è già implementata nel DBMS.

#### Estrazione ritardata

Per quanto riguarda l’estrazione ritardata, questa si può basare sulle marche temporale, cioè quando il caricamento dei dati nella staging area sfrutta la storicizzazione (sempre presente in questi casi), nei sistemi semi-storicizzati però potrebbe esserci perdita di dati intermedi.

La comparazione tra file è un metodo di estrazione che confronta due versioni successive dei dati per evidenziarne le differenze, quindi richiede il mantenimento di una versione precedente. Anche in questo caso potrebbero esserci perdite di dati intermedi, tuttavia si tratta dell’ultima spiaggia quando le tecniche precedenti non funzionano.

### Trasformazione e pulizia

Nella fase di trasformazione tutti i dati nella staging area vengono:

* convertiti nel formato;
* arricchiti con calcoli derivati;
* separati o concatenati per permettere la standardizzazione.

La fase di pulizia riguarda in generale la correzione degli errori, da quelli di battitura alle inconsistenze, alle informazioni mancanti o duplicate, la maggior parte di questi viene corretta attraverso l’utilizzo dei dizionari e/o delle regole di correzione ad-hoc.

Questa fase è molto importante dato che riguarda la veridicità dei dati nel data warehouse, proprio per questo motivo molti investono in questa fase.

### Loading nel database

Il caricamento dei dati nel database riconciliato dipende dal tipo di estrazione utilizzata:

* con l’estrazione statica, alla si alimenta il database da zero;
* con quella incrementale invece occorre guardare le varie corrispondenze, un esempio è la “Global as view”.

Esistono inoltre diversi casi sulla base della tipologia del database:

* nei database non storicizzati, conviene avere un campo che indica se il dato è nuovo o è stato cambiato, quindi si considerano solamente le tuple cambiate;
* nei database storicizzati invece, su utilizzano due attributi per indicare l’inizio e la fine, di conseguenza è facile capire cosa è nuovo.

### Loading del data warehouse

Per caricare i dati nel data warehouse, prima si alimentano le dimension table e poi le fact table per rispettare i vincoli di integrità referenziale.

Tutto ciò avviene in due fasi:

* identificazione dei dati da caricare nel database riconciliato attraverso le marche o i metadati;
* sostituzione delle chiavi con surrogati per le tuple nuove oppure mantenendo le vecchie corrispondenze con opportune modifiche. In questa fase si implementa la gerarchia dinamica con tutti i suoi aggiornamenti.

L’alimentazione delle fact table avviene solo nel caso in cui vengono inserite nuove tuple, tutte le corrispondenze si trovano nei metadati.

In caso di modifica, tutte le corrispondenze vengono mantenute con le vecchie chiavi utilizzando una tabella apposita nella staging area.

Se invece si vuole fare storicizzazione dei dati, è necessario gestire le manche temporali di inizio e fine di validità del record, mettendo a NULL il campo di fine quando la tupla è nuova o ancora valida. Nella fase di alimentazione successiva saranno considerati solamente i dati il cui campo di inizio è successivo all’ultimo aggiornamento del datawarehouse.

### Alimentazione delle viste

Cosa si può dire dell’alimentazione delle viste? Ebbene, l’ordine di caricamento viene deciso dal grafo per effettuare il minor numero possibile di caricamenti, lo stesso risultato si può ottenere anche senza grafo ma a un costo maggiore.

# Cenni di Big data

I big data hanno un ruolo fondamentale nella business intelligence, l’idea è quella di avere dati in più utili per orientarsi verso la miglior decisione.

Le sorgenti dei big data riguardano:

* l’interazione tra utenti come i social network e le email, il cosiddetto Web 2.0;
* i produttori automatici e logger come router e server;
* gli archivi multimediali;
* gli archivi documentali e operazionali, di solito i dati ERP.

## Caratteristiche

I big data hanno la caratteristica di essere in grande quantità e con alta eterogeneità, questi dati non hanno inoltre uno schema preciso per via della loro diversità (non strutturabilità). Ultima ma non meno importante è la loro velocità di produzione, le sorgenti producono dati in ogni momento.

Data la grande eterogeneità dei dati, questi possono avere volumi e complessità differenti tra loro:

* i dati prodotti dall’IoT ad esempio sono abbastanza semplici ma vengono prodotti in gran quantità, cosa che ne aumenta il volume;
* i server log e le informazioni di navigazione sono tra i dati nella media sia nel volume, sia nella complessità;
* I dati complessità e ad alto volume riguardano il Web 2.0 in generale, l’interazione tra utenti è una cosa che è infatti complessa da sintetizzare ed è una cosa che avviene continuamente nei social network.

## Sistemi per il trattamento dei big data

Come vengono gestiti questi dati data la loro mole? Ebbene, bisogna capire la piattaforma migliore in base ai tipi di dati.

Il flusso dati continuo viene gestito utilizzando i database Historian, dei database particolari capaci di rilevare trend nei dati, un’ altri metodo molto simile è il Complex Event Processing.

E per quanto riguarda la mole dai dati? Per questo problema esistono appositi standard come Hadoop, un’altro metodo è l’utilizzo di appositi paradigmi, tra quei il migliore è il Map-Reduce.

E per la mancanza di strutturazione? A fronte di ciò, l’utilizzo dei database relazionali non permette una corretta rappresentazione a causa di questo, perciò si utilizzano i database NoSQL, essi permettono più eterogeneità tra i dati. I modelli NoSQL possono esser::

* i modelli a colonna, i quali forniscono una visione dei dati raggruppata in colonne, utile per l’analisi dati ed è molto flessibile sugli schemi di attributi;
* i modelli a grafo, i quali permettono di rappresentare le informazioni su un grafo e le loro relazioni;
* i modelli a documenti si appoggiano a XML o Json per memorizzare i dati in modo strutturato, questi sono basati su tag definiti dall’utente. Questo approccio permette flessibilità di utilizzo.

L’attuale trend è quello di utilizzare soluzioni ibride che utilizzano SQL per le interrogazioni.

## Analisi dei dati

L’analisi dei dati avviene utilizzando varie tecniche di statistica e di intelligenza artificiale, si parte dalla data analytics, al deep learning, al data mining.

## La nuova frontiera della business intelligence

I big data possono essere utili nel supporto alle decisioni insieme al datawarehouse, magari utilizzando l’intelligenza artificiale. Tutto ciò ha varie applicazioni che vanno dal marketing, alle vendite, alla logistica, eccetera.

## Esempio: azienda agricola

Un’azienda agricola deve mantenere le coltivazioni tenendo conto dei fattori climatici e ambientali, oltre a poter utilizzare prodotti regolamentari per eventuali trattamenti.

Quello che serve è un’insieme di tecnologie che permettono il riconoscimento di insetti da parte di droni e da smartphone e un monitoraggio tramite IoT. Questi ultimi dati verranno integrati con dati previsionali meteorologici (i big data appunto).

Oltre a tutto questo, c’è anche la gestione del terreno, utile per capire quando è opportuno bagnare oppure dare dei fertilizzanti.