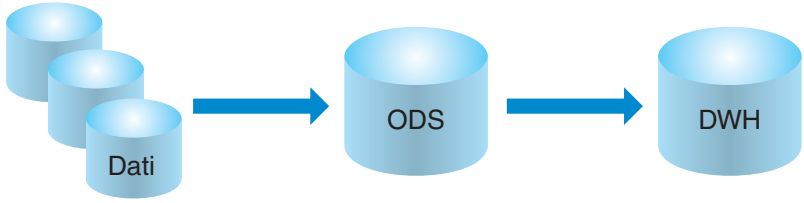
Approfondimento sui pattern architetturali

I pattern architetturali sono soluzioni riconosciute ed adottate per la soluzione di problemi comuni nel campo dell'architettura del software.
La panoramica dei pattern architetturali trattata in questo testo, non intende essere esaustiva, in quanto il tema dei pattern architetturali è in continua e costante evoluzione.

Area di applicazione: integrazione dei dati

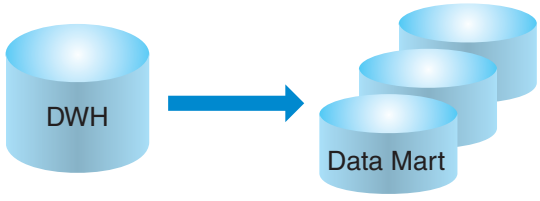
Operational Data Store (ODS) e Data Warehouse (DWH)

Un Operational Data Store, spesso indicato con l'acronimo ODS, è archivio di dati operazionale centrale che raccoglie informazioni da varie fonti dati eterogenee. Tali fonti dati possono essere archivi di procedure diverse, memorizzati in sistemi di database che implementano tecnologie diverse.
L'ODS consente di portare tutti i dati che siano oggetto di un aspetto di analisi in un unico archivio centrale, facilitando la costruzione di nuove relazioni tra dati che normalmente non sono tra loro relazionati.
Un ODS può essere la fonte dati principale di un Data Warehouse.
Un Data Warehouse, spesso indicato con l'acronimo DWH, è un database che svolge il ruolo di storicizzazione dei dati che vengono archiviati in un normale database relazionale. In un'architettura di dati con un DWH, il database relazionale che viene interrogato e che è oggetto di transazioni, contiene solo l'ultima versione di ogni informazione, la più aggiornata. Quando un'informazione viene modificata da una transazione, nel DWH viene memorizzata la nuova informazione, senza però eliminare l'informazione precedente, ma solo indicando che tale informazione non è più quella aggiornata.
Il DWH quindi è un database che può solo crescere e i cui dati storici non vengono mai modificati. Dato che la costruzione di un archivio DWH è onerosa, normalmente il DWH non contiene informazioni aggiornate in tempo reale, ma aggiornate con tempi più dilatati.



Data Mart

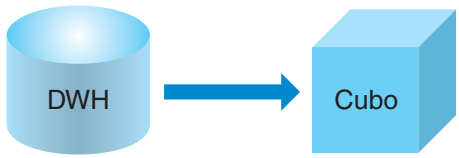
Un Data Mart è una base dati destinata all'analisi e alla reportistica. Se nel DWH è presente il dato grezzo, non ancora processato con strumenti statistici, nel Data Mart vengono archiviati i risultati delle possibili interrogazioni di analisi che possono essere rappresentate su strumenti di reportistica. In un Data Mart per esempio, possono essere memorizzate tutte le totalizzazioni, medie, aggregazioni che è possibile visualizzare sugli strumenti di reportistica. Questo consente agli strumenti di reportistica di non dover generare i dati utilizzando ogni volta interrogazioni complesse, ma utilizzando dati già *pre digeriti*, con il risultato di una risposta più pronta dal punto di vista dell'utente finale.



© Istituto Italiano Edizioni Atlas

Analisi Multidimensionale dei Dati

L'analisi dei dati con un Data Mart, ha una limitazione: presuppone la conoscenza a priori di quelle che saranno le possibili interrogazioni che gli utenti finali potranno fare con i loro strumenti di reportistica.
Con gli strumenti di analisi dei dati multidimensionale, è un software che si occupa di determinare da solo tutte le possibili aggregazioni di dati, in base ad un modello che lascia all'utente finale una libertà molto maggiore di definire le proprie interrogazioni.
Il motore multidimensionale costruisce periodicamente, a partire da un ODS o da un DWH, quello che normalmente viene definito un *Cubo* di dati che viene interrogato con appositi linguaggi di analisi dei dati.



ETL (Data Extraction, Transformation, & Loading)

Le applicazioni di integrazione di dati, estraggono i dati da diverse fonti e li depositano, eventualmente trasformati con aggregazioni, calcoli, totalizzazioni o altre valutazioni, in archivi destinati all'analisi.
I programmi che realizzano tale spostamento e trasformazione dei dati vengono realizzati utilizzando il pattern architetturale ETL.
Il programma individua le fonti e le interrogazioni necessarie per estrarre le informazioni utili all'analisi. Quando viene eseguito i dati vengono estratti e idealmente disposti in un'area dove possono essere lavorati o trasformati. I dati così prodotti vengono memorizzati sugli archivi di destinazione.
Normalmente i programmi ETL vengono eseguiti in base ad una *Pianificazione* (*Schedule*) che va tenuta in considerazione durante la fase di progettazione del programma in quanto influenza il significato che i dati hanno una volta che sono stati lavorati ed archiviati.

Area di applicazione: sviluppo generico di programmi e procedure

Layers

Come già discusso e sperimentato nel testo il pattern architetturale Layers promuove la realizzazione di componenti con chiare e limitate responsabilità.
Tipicamente l'architettura Layers assume la foma più specifica di *Three-tier architecture*, già discussa nel testo, in cui tre diversi strati determinano la struttura di un'applicazione complessa:

- Data-tier: lo strato dei dati, dove si definisce la struttura dei dati con cui un programma opera e dove contestualmente si definisca la modalità con cui questi dati devono essere conservati, la cosiddetta *persistenza dei dati*.
- Logic-tier: nello strato logico i dati messi a disposizione dallo strato dei dati realizzano la logica applicativa, seguendo le regole del programma e costruendo l'ossatura del servizio che l'utente finale percepisce.
- Presentation-tier: lo strato di presentazione infine, definisce la forma e l'aspetto con cui dati e procedure possono essere utilizzate dall'utente finale.

Inversion of Control

Quando si realizza un programma utilizzando un linguaggio di programmazione orientato agli oggetti, le classi che sono le piccole porzioni di programmi realizzano le funzionalità dell'intero programma cooperando tra loro in relazioni d'uso che vengono dette *accoppiamenti*.

© Istituto Italiano Edizioni Atlas

Gli accoppiamenti sono essenziali per far funzionare correttamente il software, ma è dimostrato che maggiore è il numero di accoppiamenti, maggiore è il costo di realizzazione e di manutenzione del software e minore è la capacità di comprenderne e prevederne il funzionamento nelle varie situazioni di utilizzo. È preferibile che gli accoppiamenti avvengano non tra classi e classi, ma tra classi ed interfacce.

L'inversione del controllo è quello stile architetturale che si realizza promuovendo gli accoppiamenti nella forma classe-interfaccia e demandando la selezione della corretta implementazione dell'interfaccia da accoppiare ad un programma di controllo esterno a tutte le classi.

In questo modo il controllo si inverte, nel senso che non sono più le classi a decidere autonomamente quali sono le implementazioni dei loro accoppiamenti, ma è un controllore centralizzato.

Area di applicazione: sviluppo di interfacce utente

Model-View-Controller

Come già discusso e sperimentato nel testo il pattern architetturale Model View Controller, suggerisce di definire il programma che realizza un'interfaccia utente affidando alle classi del programma tre diverse tipologie di responsabilità:

- Model: è l'insieme delle classi del modello; queste rappresentano le informazioni che si vogliono visualizzare sull'interfaccia utente; possono essere le informazioni base, per esempio estratte da un database oppure anche una rielaborazione di queste informazioni.
- View: è l'insieme delle classi di visualizzazione; queste si occupano di generare i componenti visibili del programma, le vere e proprie interfacce. La loro responsabilità è quella di disporre gli oggetti sullo schermo, deciderne colori, dimensioni, effetti e capacità di interazione con l'utente.
- Controller: è l'insieme delle classi di controllo; queste si occupano di comprendere le richieste espresse dall'utente attraverso gli strumenti messi a disposizione delle classi di visualizzazione. Una volta comprese le richieste dell'utente, le classi di controllo utilizzano le librerie del programma per reperire le informazioni di base e preparare un nuovo modello da inviare a nuove classi di visualizzazione.

Event-driven architecture

L'architettura guidata da eventi, pone l'accento sulla produzione, identificazione, utilizzo e reazione agli eventi. Gli eventi sono associati a classi che ne definiscono la struttura e ne permettono la gestione da programma.

Nella programmazione di interfacce grafiche per esempio generalmente abbiamo classi di oggetti di interfaccia quali bottoni, campi di testo, etichette. A queste può essere associata la possibilità di rilevare eventi di interazione da parte dell'utente, quali pressione dei tasti o movimento del mouse, utilizzo dei tasti della tastiera o anche semplicemente lo scorrere del tempo.

Il programmatore decide di utilizzare la possibilità di un componente di reagire ad un evento e ne programma la reazione.

Implicit invocation

Il pattern architetturale di invocazione implicita è una forma specifica del pattern di Inversione del Controllo, in cui gli oggetti anche hanno bisogno di interazione con altri oggetti, esprimono tale bisogno *lanciando in broadcast* un evento specifico.

Le implementazioni che sono in grado di fornire il servizio sono in grado di *ascoltare* questi eventi e di fornire i servizi richiesti.

© Istituto Italiano Edizioni Atlas

Area di applicazione: sviluppo di servizi distribuiti

Service-oriented architecture

Come già discusso e sperimentato nel testo il pattern architetturale Service Oriented Architecure promuove la realizzazione di una logica applicativa complessa attraverso la realizzazione di una collezione di servizi in grado di realizzare ciascuna, porzioni circoscritte del compito complessivo del software.
La realizzazione di questi programmi come servizi, ne consente la distribuzione su sistemi costituiti da più nodi distribuiti sulla rete.

Peer-to-peer

In contrapposizione alla service oriented architecture in cui un servizio complessivamente e fornito attraverso una collezione di servizi di dettaglio più fine, nell'architettura Peer-to-peer (spesso abbreviate con l'acronimo P2P) ogni elemento eseguibile su un nodo svolge tutti i compiti previsti, ma è in grado di cooperare con uno o più altri nodi dello stesso tipo (i peer) per svolgere il compito in minor tempo o in diversi punti della rete.

© Istituto Italiano Edizioni Atlas