DATA LIFECYCLE

- Si descriva il ciclo di vita dei dati secondo la metodologia CRISP-DM. Il processo standard intersettoriale per il data mining, noto come CRISP-DM, è un modello di processo standard aperto che descrive gli approcci comuni utilizzati dagli esperti di data mining. Il data lifecycle è composto da sei fasi che non sono necessariamente sequenziali. Nella maggior parte dei casi ci si sposta avanti e indietro tra le fasi in base alla necessitá.

- **Busines understanding**: in questa fase si elabora il piano del progetto utilizzando le informazioni raccolte durante il processo di analisi, si comprendono gli obbiettivi e si traduce il problema dell'utente in un problema di data mining.
- **Data understanding**: questa fase implica un esame più attento dei dati disponibili per il data mining, per evitare problemi imprevisti durante la fase successiva di Data Preparation. La comprensione dei dati implica l'accesso ad essi e la successiva analisi mediante tabelle e grafici in modo da definire la qualità dei dati e descrivere i risultati di tali passaggi nella documentazione del progetto.
- **Data preparation**: comprende tutte le attività necessarie a creare il dataset finale (selezione di attributi, eventuale trasformazione di variabili e pulizia dei dati). A seconda degli obiettivi la preparazione dei dati implica: unione di record di dati, scelta di un sottinsieme di dati campione, aggregazione di record, cteazione di nuobi attributi, ordinamento dei dati per la modellazione, rimozione o sostituzione di valori nulli o mancanti, partizionamento del dataset in train data e test.
- **Modeling**:la modellazione viene in genere eseguita in più iterazione di solito, si eseguono diversi modelli e si utilizzano parametri di default, per poi regolare i parametri o tornare alla fase di Data Preparation per manipolazioni richieste dal modello scelto.
- **Evaluation**: questa è una delle ultime gasi del progetto. Dopo aver determinato nella fase Modeling, che i modelli creati sono tecnicamente corretti ed efficaci, secondo i criteri per la valutazione dell'esito del data mining, questi vengono analizzati al fine di verificare che siano sufficientemente precisi e robusti da rispondere adeguatamente agli obbiettivi dell'utente.
- **Deployment**: è la fase finale ed il modello costruito deve essere messo a disposizione degli utenti. Si scrive un report oppure si richiede di implementare un sistema di data mining controllabile direttamente dall'utente. In generale questa fase include due attività: Pianificazione e monitoraggio dei risultati e completamento di operazioni riepilogative, come la stesura di un report finale e la verifica del progetto.

- Definizione di data management. Data Managemente si riferisce al processo e alle pratiche usate per assicurare che il dato sia propriamente collected, stored e mimized durante tutto il suo lifecycle.

DATA ACQUISITION

- Definizione data acquisition. Data acquisition è il processo di collezionare ed importare i dati per le analisi. Gli step che la compongono dipendono dai requisiti specifici per l'analisi e dalle caratteristiche dei dati. Quando acquisiamo i dati dobbiamo fare delle assunzioni, come il fatto che non possiamo essere completamente sicuri riguardo l'informazione che raccogliamo.

- Tecniche di acquisizione di dati. Ci sono divesi metodi che si possono usare per acquisire i dati, che dipendono da dove i dati sono situati e dalle licenze di essi. I metodi sono download, tramite API e tramite web scraper. Il download è il più facile, e nel caso sia possibile è sempre la prima scelta, spesso però chi detiene i dati non li condivide cosi facilmente. Può essere che i dati vado acquisiti tramite API (Application Programming Interface), un set di interfacce standardizzate che permettono a differenti sistemi software di comunicare tra loro. L'API fa si che differenti applicazioni si scambino i data per eseguire azioni specifiche. Di fatto mettono a disposizione i dati all'utente autorizzato che dovrà interagire con il sistema per ricevere degli output. Infine nel caso non sia possibile ne il download ne l'uso di API la soluzione è il web scraping. Lo scraping consiste nel simulare attraverso algoritmi l'attività di un utente su una pagina web e acquisire tutte le informazioni viste, cercando di non venire intercettati e bloccati, la legalità non è chiara.

- Tecniche di acquisizione API . Per acquisire i dati è possibile usare le API. GLi Application Programming Interface (API) sono un interfaccia standardizzata e sicura che permette ad applicazioni diverse di comunicare. Le API permettono di richiedere dai a provider di terze parti in modo da accedere ai dati di nostro interesse. Le API sono tipicamente di dati statici. Un tipo comune di API sono le REST API (REpresentational State Transfer) che usano un protocollo HTTP per trasferire informazioni tra applicazioni in modo standardizzato, efficiente e sicuro. Sono sviluppate per essere flessibili, scalabili e facili da usare. I comandoi fondamentali sono CRUD (create, retrive, update, delate). La maggior parte delle API richiede di autenticarsi e un

modo comune di farlo è quello di usare una API Key, ovvero una stringa univoca di caratteri assegnata all'utente utilizzata per autorizzare le richieste sull'API. Alcune API sono libere, altre danno un numero di informazioni (o un tipo) gratuitamente e per il resto bisgona pagare, altre ancora sono completamente a pagamento. Le API REST mettono a disposizione un insiemen di URL che il client usa per accedere alle informazioni del servizio web, queste URL (per le API) sono chiamati "endpoint". Il risultato di una richiesta (fatta con filtri ed endpoint) è spesso un file JSON (basato su objects e array), non esiste un qiuery lenguage standard per JSON, ad esempio si può usare Kinme. Le API possono avere problemi con dati in real-time.

- Perché Kafka garantisce la velocità. Le API quando si trovano a gestire dati in real-time (il che non vuol dire veloci, ma con costrutti temporali espliciti ed impliciti) possono avere dei problemi che risultano in data loss. Kafka può essere usato per questo tipo di raccolta dati. Kafka è una piattaforma per il data streaming distribuita che è pensata per gestire un grande volume di dati in tempo reale. Tramite Kafka è possibile operare modifiche o trasformazioni dei dati durante il flusso. Non è quindi necessario affidarsi a un servizio esterno per gestire attività di storage e processing. Proprio per questo motivo kafka garantisce la velocità nella raccolta dei dati. Kafka può essere però usato solamente se il provider lo permette.

- Definire tecniche di web scraping. Se un provider non vuole condividere i suoi dati l'unica soluzione è estrarli da pagine web con uno scraper. Ci sono diverse tecniche di scraping, come estensioni del browser ("Web Scraper") le quali sono limitate nella loro capacità e non gestiscono bene le pagine web complesse, in più non schivano la detection dal sito. Per schivare la detection possono servire delle funzioni di sleep tra una richiesta e l'altra. Scraper più complessi usano linguaggio HTML (HyperText Markup Language) che è un linguaggio markup usato per le strutture delle pagine web, che usa tags per descrivere le strutture. Per estrarre i dati c'è bisogno di sfruttare regular expression o XPATH. Questa soluzione è disponibile in Knime o in python con la libreria BeautifulSoup. A volte c'è bisogno di un passaggio ulteriore in questo caso Selenium, libreria per controllare web brouser tramite programmi e automatizzazioni, è l'unica soluzione.

DATA STORAGE

- Data modeling descrizione. Data modeling è il processo di design e di creazione di un modello che rappresenti parte del mondo in modo che si possa essere stored (write) e quiried (read). A data model è una rappresentazione di strutture dati e relazioni esistenti in un particolare dominio, e consiste in un modello concettuale e un linguaggio per descrivere e fare query sui dati. Un data model è tipicamente implementato usando un DBMS che supporti la semantica del modello. I modelli di dati più famosi sono quello relazionale (relational) basato sul modello entità e relazione e NoSQL che cercano di superare i limiti del modello relazionale e ognuno di essi ha caratteristiche diverse ed la scelta del modello dipende dagli scopi della ricerca. Le feature più importanti sono: Machine redability, Ezpressive power, Simplicity, Flexibility, Standardization.

Relational Model

- Modello relazionale descrizione. Il modello relazione è un modello logico di strutturazione dei dati di un databease. È basato sull'idea di rappresentare dati come righe in delle tabelle e ogni tabella con un nome univoco e con colonne con nomi univoci. Le tabelle possono essere legate tra di loro usando foreign keys, che sono colonne con valori che si riferiscono a primary key in altre tabelle. In genere ogni tabella destive un entità che può avere relazioni che possono essere 'una ad una', 'una a molti' o ;molti a molti'. Nel modello relazionale 'no iformation' è un informazione. Il modello relazionale è basato sulla "closed world assumption" ovvero sul fatto che tutto quello che è rilevante è contenuto nel database, se qualcosa non è presente è assunto falso. Un altra caratteristica è il "minimization principle" ovvero il fatto che i dati immagazzinati devono essere minimizzati per efficienza. SQL (Structured Query Language) è un linguaggio usato per interagire con i database che seguono il modello relazionale. I modelli relazionali sono ben gestiti da RDBMS, hanno aspetti positivi come quello di schema rigido e di seguire le proprietà ACID (Atomicity, Consistency, Isolation, Durability). I vantaggi di questo modello possono risultare come limiti, poiché: un attributo può avere un solo valore, non è compatibile con i moderni linguaggi di programmazione ad oggetti, non supporta relazioni cicliche, ed è difficile modificare lo schema una volta implementato. Inoltre non può fare scale out ma solo scale up.

- Le proprietà ACID. Con il termine ACID si indicano le proprietà logiche che devono avere le transizioni nei database relazionali.

- **Atomicity**: questa proprietà assicura che la transizione sia tutto o nulla. O tutta l'operazione va a buon fine o niente viene completato. Questo aiuta a prevenire inconsistenza e corruzione dei deati.
- **Consistency**: Una transizione deve lasciare il database in uno stato consistente, inipendentemente dal suo completamento. Questo vuol dire che la transizione non deve violare nessun costrutto o regola del database.
- **Isolation**: Le transizioni devono essere isolate tra di loro, cosi che gli effetti di una transizione non siano visibili ad altre fino a che non è completata. Questo aiuta a prevenire conflitti e corruzione dei dati.

- **Durability**: Una volta che una transizione è completata i suoi effetti sono permanenti e non sono soggetti a perdite, anche in caso di chras del sistema.

Queste proprietà forniscono un meccanismo per assicurare la correttezza e la consistenza di un database, in modo che ogni modifica è un gruppo di operazione che agisce su un'unità singola, producendo risultati consistenti. Queste proprietà sono presenti nel RDBMS.

- Approcci di memorizzazione in un DBMS. !!!!!!!!!!!!!!

NoSQL

- Schema NoSQL, differenze con Relational. NoSQL sta per "Not only SQL" e si riferisce ad una grande varietà di modelli pensati per scalare in orizzontale tra più server. Sono spesso usati in situazioni troppo complesse per essere gestite bene da un database relazionale. Una delle caratteristiche principali è il loro essere "schema-free"il che vuol dire che lo schema può variare da record a record. Questi modelli si basano sulla "open world assumption" che vuol dire che i dati fuori dal database potrebbero essere rilevanti, questo in contrasto ai relazionali per i quali esiste solo ciò che si ha. I database NoSQL permettono grande flessibilità e complessità però possono non essere la scelta migliore nel caso si voglia forte integrità e consistenza. I modelli NoSQL possono seguire sia i BASE principless che il CAP theorem. Il primo è un insieme di linee guida per il comportamento dei database NoSQL, i secondo sono principi che si applicano ai sistemi distribuiti.

- Pro e contro schema free NoSQL. I database NoSQL forniscono diversi vantaggi importanti rispetto ai relazionali. Lo schema free, ovvero il fatto che non esista uno schema prefissato e che esso può cambiare da record a record, permette una maggior adattabilità e flessibilità, ma anche il fatto che potrebbe non garantire lo stesso livello di integrity e consistenza di un modello relazionale. Lo schema free rende il modello scalabile in orizzontale e questo garantisce alte performance a alta disponibilità. Il non avere uno schema fisso implica però anche che il database sia difficile da interrogare. Non si è in grado di 'normalizzare' i dati, il che può rendere l'archiviazione dei dati più complessa e le query di grandi dimensioni più lente.

- Spiegazione CAP theorem Il CAP theorem è un principio che si applica ai sistemi distribuiti, ciò include alcuni database NoSQL. Il CAP theorem dice che: è impossibile per un sistema distribuito di computer, garantire simultaneamente tutte e tre: Consistency, Avability, Patition Tollerance. Consistency dice che tutti i nodi devono vedere gli stessi dati allo stesso momento, Avability diche che tutte le richieste ricevono una risposta (che sia positiva o fallita), Partition tollerance dice che il sistema continua ad operare nonostante la perdita o il fallimento di una parte di esso. Sono possibili solo due, dunque i RDBMS seguono CA generalmente, mentre alcuni NoSQL seguono CP o AP.

- BASE principles. Per certe situazioni un modello basato sulle proprietà ACID (Atomicity, Consistency, Isolation, Durability) potrebbe essere troppo rigido e quindi non adatto al funzionamento del database. Quindi alcuni moelli NoSQL si basano su u modello più morbido, basato sulle proprietà BASE.

- **Basic Avability**: Il dataset è sempre disponibile a richieste anche se non è possibile garantire forte consistenza tutto il tempo. Questo è possibile utilizzando un approccio distribuito alla gestione dei database.

- **Soft State**: Lo stato del database puo variare nel tempo (schemi diversi per i dati), il database può non imporre regole rigide per la consistenza.

- **Eventual Consistency**: Prima o poi nel futuro i dati nel database convergeranno ad uno stato di consistenza, ma non c'è garanzia sul quando. Quindi quando nuovi dati vengono aggiunti al sistema essi si propagheranno gradatamente un nodo alla volta, fino a far diventare consistente l'intero sistema. Questo è in contrasto con la 'immediate consistency" delle proprietà ACID.

- Tipi e differenze modelli NoSQL. Ci sno svariati ti pi di database NoSQL possibili.

- **Key-value stores**: Il database raccogli i dati come un inseme di coppie chiave-valore. La chiave è usata per identificare il valore, che può essere qualsiasi tipo di dato. Sono molto efficienti per cercare e spesso sono usati per grandi quantità di dati che non necessitano query complesse. Sono basati su algoritmi di hash e sono efficenti nella data distribution.

- **Column Family Stores**: Questi database immagazzinano i dati in colonne al posto che in righe, dove la chiave si riferisce ad un insieme di colonne. Spesso sono usati in ambito di big data, perché sono distribuiti tra molti server.

- **Document Database**: Questo tipo di database è in forma di documenti, che sono tipicamente in formato JSON. Sono flessibili e permettono grande varietà di strutture dati.

- **Graph Database**: Immagazzinano i dati sotto forma di ”node” e ”edge”, che possono essere usati per rappresentare entità e relazioni tra esse. Sono adatti a strutture complesse e spesso usati per il networking, tuttavia non scalano molto bene.
- **RDF db, Tuple stores**:

- Si descriva il Document Database. Il Document database è un modello NoSQL conosciuto per lo schema flessibile e per la possibilità di immagazzinare i dati in strutture nested multidimensionali, detti documenti. Lo schema è flessibile e modificabile nel tempo, poiché non c'è bisognio di una struttura rigida. I documenti possono essere messi in relazione tra loro con dei riferimenti e ciò rende il modello document based particolarmente adatto a rappresentare le gerarchie. Un esempio è MongoDB, dove i dati sono immagazzinati in formato Bson. Si può accedere ai dati grazie agli indici. Non utilizza join. Ogni documento può contenere zero, uno o più valori embedded, e le query possono essere fatte su ogni livello del documento. Ogni documento in MongoDB deve essere immagazzinato in una collezione, che a sua volta deve essere immagazzinato in un database. I dati possono venire creati e manipolati semplicemente inserendo documenti in collections, i dati caricati possono essere manipolati liberamente anche in grandi quantità, possono essere caricati dati anche da sorgenti esterne. La caratteristica più importante di mongoDB è che offre la possibilità di svolgere operazioni quali le query aggregate. Per eseguirle si usa l'operazione aggregate, che introduce il concetto di pipeline: La pipeline è una sequenza di operazioni di aggregazione sui dati o sui livelli, le opzioni sono delle opzioni aggiuntive che possono essere passate all'operatore. La pipeline di aggregazione parte processando i documenti della collezione e passa il risultato all'operatore successivo fino ad arrivare al risultato finale.

- Si descriva il Graph Database, anche con esempi, le differenze con Document db Un modello a grafo è un modello di rappresentare dei dati come nodi e archi, dove i nodi rappresentano concetti o entità e gli archi le relazioni tra essi. È possibile usare anche le proprietà, attributi che si attaccano ai nodi o agli archi per informazioni aggiuntive. Possono risultare utili per specificare la forza di una relazione o per distinguere differenti istanze dello stesso concetto o entità. Questi modelli sono spesso usati per rappresentare strutture complesse ed interconnesse, e possono essere usati in svariati campi come i social network, sistemi di raccomandazione o bioinformatica. La struttura dei dati è nei dati stessi, è quindi facile da capire e lo schema si modifica nel temo seguendo l'evoluzione dei dati.

Un database a grafo ha due componenti principali, ovvero lo ”storage engine” e il ”processing engine”. Lo storage engine è responsabile per la gestione del data storage, mentre il processing engine è responsabile dell'esecuzione di query e del processare i dati. Insieme sono responsabili delle performance, della scalabilità e della leggibilità del sistema.

Ci sono due tipi di database a grafo: i ”nativi” e i ”non nativi”. I nativi sono ottimizzati per immagazzinare e mappare i dati in grafi, mentre i non nativi immagazzinano i dati in un modello non a grafo, ma supportano un query language che permette di fare query basate sui grafi.

Un problema dei database a grafo è la mancanza di un query language standard. Per questo motivo può essere difficile cambiare o integrare tra diversi graph db, ed è possibile dover ripartire da zero. Neo4j è molto popolare come graph db ed usa ”cypher”, mentre ”gremlins” è usato da diversi graph db. Cypher è un linguaggio pattern-matching espressivo e dichiarativo, ovvero dove si descrive cosa si vuole e non come ottenerlo, i risultati delle query sono tipicamente tabelle, ma non possono essere usati per altre query. Gremilins è un linguaggio vertex-based, anche conosciuto come traversal language, è un domanin-specific language (DSL), che usa espressioni per specificare una serie di passaggi trasversali che si vuole percorre nel grafo, queste espressioni vengono concatenate per formare query. Un alro caso son i ”polyglot db” che permettono diversi linguaggi diversi, però qury complesse hanno cattive performance.

La differenza tra il modello a grafo e quello documentale è che in un database di documenti, il valore contiene dati strutturati o semi-strutturati, questo valore strutturato o semistrutturato è chiamato documento. Mentre nei modelli a grafo le caratteristiche sono diverse poiché contengono nodi e relazioni, ogni nodo contiene delle proprietà, i nodi possono essere etichettati con una o più etichette, le relazioni sono nominali e direzionali. La differenza sostanziale tra modelli documentali e grafo è data dai cicli; l'utilizzo di un albero non permette di poter avere cicli. A differenza di un modello documentale come MongoDB in cui posso distribuire facilmente i dati e avere grandi moli, il modello a grafo non è scalabile così facilmente. Quindi i modelli a grafo non sono scalabili.

Datawarehouse

- Definizione di data warehouse. La definizione di DW è: ”Una data warehouse è un magazzino singolo, completo e consistente, di dati ottenuti da svariate fonti e resi disponibile agli utenti finali in modo che possano capirli e usarli per contesti di business.” Il DW è un database sviluppato per supportare efficientemente query e analisi, tipicamente è organizzato e strutturato in modo da facilitare le operazioni. È spesso usato insieme a tool e tecnologie com OLAP (online analytical processing).

La DW è una collction di dati ”subject-oriented”, ”integrata”, ”time-varying” e ”non volatile”. Subject oriented si riferisce al focus nel riguardo di specifici soggetti, come ad esempio i clienti, questo permette la facilitazione dell'interpretazione. Integrata si riferisce la fatto che la DW combina dati da fonti diverse, questo permette di analizzare i dati di fonti diverse in modo unito

piuttosto che avere diversi dataset disconnessi. Time-varying che le DW immagazzinano dati storici e contemporanei, in modo da analizzare trend. Non-volatile si riferisce al fatto che i dati nella DW non sono aggiornati o cancellati come parte di una normale transizione, questo permette di conservare i dati per tanto tempo.

La combinazione di queste caratteristiche fa si che la DW sia un tool ideale per il decision makeing e per analizzare e immagazzinare grandi quantità a di dati. Tuttavia la fase di creazione è un processo lungo e costoso, e l'integrazione di una DW può portare a trovare molti problemi di data quality.

- Si descrivano le principali differenze anche con esempi applicativi fra l'integration dei dati e i sistemi di data warehouse. Quando ci si trova a dover gestire differenti fonti si hanno due strade, una lazy ovvero la data integration e una edger ovvero le DW.

- **Data Integration**: è query driven, il sistema di integrazione è direttamento connesso con i clienti. Si articola in due fasi: record linkage e data fusion. I dati sono combinati da fonti diverse solo su richiesta e richiedono query specifiche, questo vuol dire che i dati sono integrati e trasformati al momento delle query. Questo può risultare inefficiente per query complesse poiché potrebbe richiedere di combinare e trasformare grandi quantità di dati al momento. Inolte non vengono considerati i dati storici ma solo quelli al momento della query, questo fa si che non sia possibile analizzare trend nel tempo. La connessione con il client appesantisce il workload.

- **Data Warehouse**: i dati immagazzinati sono stati pre-integrati e pre-processati da fonti diverse. I dati sono integrati prima dell'uso, ciò significa che sono già strutturati e organizzati cosi che sia più facile analizzarli ed interpretarli. Una DW è pensata per supportare query complesse e basate sullo storico, in modo da analizzare i trend temporali. Inoltre non è direttamente collegata al client e questo significa che può contenere grandi quantità di dati senza influire sul workload.

- Tipi di architettura DW, prestando attenzione ai data mart. Ci sono sono svariati tipi di architetture della data warehouse, ognuna con le sue feature e i suoi benefici. ETL: Extract, transform, load.

- Single-tier: tutti i componenti sono contenuti in un unico server, è il tipo piu sempilice ed è adatto a piccole organizzazioni con dati semplici.

- Two-tier: la DW è divisa in two tier chiamati "front-end" e "back-end". Il front-end contiente l'interfaccia degli utenti e il tool di reporting, mentre il back-end continente il database e i tool ETL.

- Tree-tier: la DW è divisa in tre tier: "front-end", "middle tier" e "back-end". Il fornt-end contiene l'interfaccia utente e i tool di reporting, mentre il middle tier contiene l'ETL e altre componenti di processo, e il back-end contiene il database. È più scalcabile e flessibile della two tier.

- Hub-and-spoken: la DW è organizzata intorno ad un hub centrale con data mart satelliti connessi all'hub. L'hub contiene la copia originale dei dati. Questa architettura permette una gestione dei dati decentralizzata. Ogni dipartimento fa riferimento al proprio data mart.

- Centralized DW: è un DW dove tutti di dati sono immagazzinati e gestiti da una location centrale. È più un approcio logico che altro.

Nelle DW può capitate che ci siano integrazioni e quality process sulle fonti dati, questo produce dei dati chiamati "reconcilied data". Questo è il vantaggio fondamentale di una two tier architetture che permette una netta separazione tra la fase di estrazione e la fase di feed.

I data mart sono un sottoinsieme o un aggregazione di dati dalla DW. Sono un insieme di dati con rilevanza specifica per un area business. I data mart derivati direttamente dalla DW sono detti "dipendenti", riducendo la dimensione aumentano l'efficienza. In altri casi i data mart sono direttamente riempiti da fonti dati e in questo caso dono detti "indipendenti". In questo caso la mancanza di una DW primari fa si che la fase di design sia più facile ma che nascano conflitti nel caso di svariati data mart indipendenti. Una soluzione ai problemi sono i "data mart bus" che integrano i data mart in modo logico usando dimensioni comuni, oppure le "federation" che fanno sia integrazione logica che fisica tra data mart.

- Multidimensional model, approccio OLAP. Una volta che i dati sono stati puliti integrati e trasformati bisogna capire come ottenere la massima informazione da essi. Principalente ci sono tre approcci per fare query in una DW: "reporting", "OLAP" e "data mining".

Reporting è un permette agli utenti di generare report, non richiede conoscenze IT, la struttura dei report non cambia mai.

Data minig permette di fare una sequenza di domande di solito non dirette. Richiede conoscenze IT e di solito usa algoritmi per analizzare i dati. Serve per analisi dettagliate.

OLAP sta per Online Analitycal Processing, supporta analisi multidimensionali dei dati e richiede quindi di pensare in modo multidimensionale. È organizzato in work session e permette analisi multidimensionali dei dati, inoltre è orientato agli utenti non-IT. Una sessione consiste in un ”navigation path” che riflette analisi di processo su uno o più fatti. I risultati delle query spesso sono usati in altre query, tipicamente sono in formato tabulare evidensiando dimensioni diverse in vari modi. La navigazione di OLAP è basata sul modello multidimensionale.

Il modello multidimensionale è la base per rappresentare query in una DW. I fatti di interesse sono rappresentati in cubi dove: ogni cella contiene misure numeriche che identificano i fatti, gli assi rappresentano una dimensione di interesse, e le dimensioni possono essere la base di della gerarchia tra attributi usata per aggregare i dati immagazzinati nel cubo. OLAP permette svariate operazioni sui data cube. Ad esempio ”slice fissa il valore per una dimensione creando un sub-cube, dice seleziona un sotto insieme di dimensioni per creare un sub-cube. Quando si hanno dati in forme diverse possono essere applicati metodi di aggregazione: ”roll-up” aumenta l'aggregazione rimuovendo livelli di dettaglio dalla hierarchy, ”drill down” diminuisce l'aggregazione introducendo livelli di dettaglio nella hierarchy. Infine ”pivoting” cambia il modo di rappresentare, e ”drill across” joina due cubi.

- Dimensional Fact Model (DFM), parlando anche delle dynamic dimensions. Il dimensional fact model (DMF) è un modello grafico che si usa per il design dei data mart, ovvero sotto insiemi delle DW specializzati per usi specifici. Il DFM è sviluppato in modo da: supportare efficacemente il design concettuale, avere un ambiente intuitivo per la formulazione di query, permettere il dialogo tra desiner e user (per rifinire), creare una piattaforma dove iniziare un progetto logico, fornire documentazioni esaustive.

Il DFM consiste in un inseme di ”fact schema” che modellano gli elementi principali del data mart. Ne vediamo ora i costrutti base:

- Un ”fatto” è un concetto di interesse per il processo decisionale, ha aspetti dinamici che fa si che evolva nel tempo. (es. cliente che compra qualcosa).
- Una ”misura” è una proprietà numerica di un fatto. (es. vendite misurate dallo scontrino).
- Una ”dimensione” è una proprietà con dominio finito che descrive la coordinata di analisi. (es. prodotto, negozio, data). Un fatto descrive una relazione molti a molti tra dimensioni.
- Gli ”attributi dimensionali” descrivono le dimensioni.
- La ”gerarchia” è un albero direzionato i cui nodi sono attributi dimensionali e i cui archi modellano associazioni molti ad uno tra coppie di attributi dimensionali.

Fatto -¿ Vendita , Misura -¿ quantità venduta , Dimensione -¿ negozio.

Costrutti avanzati:

- Un ”attributo descrittivo” contiene informazioni aggiuntive riguardo gli attribuiti dimensionali di una gerarchia, al quale e connesso con una relazione uno ad uno. Non viene usato per aggregazioni perché ha valori continui e per il tipo di relazione.
- Un ”attributo cross-dimensional” è un attributo dimensionale o descrittivo il cui valore è determinato dalla combinazione di due o più attributi dimensionali (un giorno può essere feriale o festivo in base alla città/regione).
- La ”convergenza” è quando due attributi dimensionali possono essere connessi da due o più cammini direzionati distinti, sempre che ciascuno di essi rappresenti ancora una dipendenza funzionale. (negozio città-regione-stato oppure negozio-distretto-regione-stato).
- La ”gerarchie condivise” è un abbreviazione usata per denotare il fatto che una porzione di gerarchia è replicata più volte nello schema.
- Un ”modello ad arco multiplo” è un associazione molti a molti tra due attributi dimensionali.
- Una ”gerarchia incompleta” è una gerarchia in cui per alcune istanze risultano assenti uno o più livelli di aggregazione. (la valle d'Aosta ha solo una provincia).

- DMF to logical model, Star schema/ snowflake schema. Per convertire un DFM in un modello logico sono necessari diversi step che partono dallo schema concettuale e finiscono con la produzione dello schema logico per il data mart. Durant e il la fase di design è possibile applicare dei principi per ottimizzare lo schema per utilizzi specifici. Cio spesso implica inserire redundancy e dernormazizzare relazioni per migliorare le performance delle query. Questo risulta in uno schema logico che può essere usato come base per l'implementazione di un data mart.

Le operazioni svolte durante il design logico sono: la scelta dello schema logico, la traslazione dello schema concettuale, la scelta delle gerarchie dinamiche, l'ottimizzazione. Lo schema logico può essere uno star sckema o un snow flake.

Lo "star schema" consiste in una singola tabella di fatti circondata da un numero di tabelle di dimensioni. La tabella di fatti contiene i valori numerici che vengono analizzati e le tabelle di dimensioni contengono le categorie o gli attributi con i quali i dati sono organizzati. Questo porta ridondanza nei dati.

Lo "snowflake schema" è una variante dello star schema che normalizza ulteriormente le tabelle. Le tabelle di dimensioni sono ulteriormente divise in sotto tabelle, che possono ricollegarsi alla tabella di dimensioni principale. Questo permette una granularità maggiore e più efficienza nelle quir, ad un costo di complessità maggiore e di performance peggiori.

Per tradurre un DFM in uno star schema si crea una tabella contenente tutte le misure e gli attributi descrittivi direttamente collegati ai fatti, e per ogni gerarchia si crea una tabella di dimensioni contenente tutti i suoi attributi. In oltre è necessario gestire anche i costrutti avanzati.

Ci sono opinioni contrastanti riguardo allo snowflake perché genera contrasto con la filosofia della DW, e la 'beautifaction' non necessaria lo rende complesso da supportare. Però può rappresentare un grande risparmio di spazio, il che però va valutato rispetto al costo in performance.

DATA PREPARATION

- Si descriva, anche per mezzo di esempi, la differenza tra l'integration dei dati e l'enrichement di un dataset evidenziando le parti in comune e i differenti scenari applicativi. La "data integration" è il processo di combinare dati da svariate fonti in una singola visione unificata. Questo può riguardate l'estrazione di dati da fonti diverse, trasformarli in un formato comune e caricarli in una central repository come una DW. La data integration si articola in due fasi: record linkage (identificazione dei set di record che identificano lo stesso "oggetto reale") e data fusion (scelta di un record unico rappresentativo del set precedente).

La "data enrichement" è il processo di aggiunta di contesto o informazione ai dati. Questo può riguardare aggiungere fonti esterne per più contesto, usare algoritmi e machine learning per ottenere insight dai dati.

Un esempio di integrazione dei dati: per ottenere una visione del cliente a 360 gradi, le informazioni da combinare possono includere dati provenienti da diversi sistemi aziendali, (dai registri del traffico Web, dal software di marketing, dalle applicazioni rivolte ai clienti, dai sistemi di supporto alle vendite e alla clientela e persino dai partner). Le informazioni provenienti da tutte queste sorgenti quindi devono essere unificate per esigenze di analisi o interventi operativi Per quanto riguarda l' arricchimento dei dati possiamo far riferimento, al settore delle vendita al dettaglio, dove le aziende arricchiscono i dati dei clienti per promozioni personalizzate e mirate per ogni cliente.

- Parlare dell'heterogeneity, citando anche gli homogeneus model. Un modello "omogeneo" è un modello costuito usando i dati da una singola fonte o dominio. I tadi usati sono assunti simili in stuttura e formato, e il modello è fatto per operare su un tipo specifico di dati.

Un modello "eterogenoe" è un modello costruito usando dati da fonti diverse. I dati possono avere differenti strutture o formati. Questi modelli sono più robusti e flessibili degli omogenei dato che possono gestire un ampia gamma di input. Però d'altro canto potrebbero essere più complessi da costruire e mantenere.

Nel contesto della data integration l'eterogeneità si riferise a differenti strutture, formati e contenuti di dati da diverse forme. L'eterogeneità può essere:

- Name heterogeneity: differenze nelle nominazioni usate tra le diverse fonti. Possono essere "sinonimi" (diverso nome stesso concetto), "omonimi" (stesso nome diverso concetto. Hypernomies
- Type heterogeneity: differenza di tipo tra i dati. Lo stesso concetto può essere rappresentato con due differenti strutture concettuali in due schemi.
- Model heterogeneity: Differenza di modelli o frameworks usati in diverse fonti per rappresentare i dati.

- Metodologia della schema integration. La schema integration si compone di tre fasi.

1. **Pre-integration**: prende in input n schemi dalle fonti, e restituisce gli schemi omogenizzati, utilizza trasformazione di modelli e reverse engieneering.

Di fatto prepara i dati per l'integrazione. Può essere di due tipi: "binary" e "n-ary". Dove binary consiste nel combinare due fonti dati in un singolo dataset integrato, può essere "composed" (concatena un dataset all'altro quando hanno formatio simile), oppure "balanced" (seleziona sottoinsiemi rappresentativi dai due dataset). N-ary combiana tre o piu fonti in un unico dataset, può essere "one step" (tutto inseme, utile quando hanno formato simile) oppure "iterative" (procede gradualmente aggiungendo una struttura alla volta)

2. **Corrispondences investigation**: prende in input gli n schemi e restituisce gli schemi e le corrispondenze, utilizza tecniche per scoprire le corrispondenze.

   Determina come dati da fonti diverse possono essere integrati e combinati. Un aspetto importante è il relativismo semantico, ovvero l'idea che differenti fonti possono avere schemi o strutture diverse per organizzare e rappresentare i dati. Si crea quindi una guida che mostra come i dati da fonti diversi possono integrarsi. Per fare questo si crea uno "schema di corrispondenze" per comparare schemi e identificare discrepanze, da risolvere a mano o in modo automatizzato.

3. **Schema integration and mapping generation**: prende in input n schemi e corrispondenze e restituisce lo schema integrato e le regole di mappatura tra lo schema integrato e gli schemi di input, utilizza una nuova classificazione dei conflitti e la soluzione dei conflitti.

   È normale integrando da fonti diverse di imbattersi in conflitti, per risolverli è necessario usare regole di trasformazione che specificano come i dati devono trasformarsi. È necessaria quindi una "classificazione dei nuovi tipi di conflitti" e delle "regole di integrazione".

   Ci sono svariati tipi di conflitti. I conflitti di classificazione si riferiscono a discrepanze nella classificazione o nel raggruppamento di attributi. I conflitti descrittivi si riferiscono a discrepanze in descrizione o definizione di attributi. I conflitti strutturali si riferiscono a discrepanze in strutture o formati di attributi. I conflitti di frammentazione si riferiscono a discrepanze nel modo in cui i dati sono suddivisi (add esempio temporalemente). Il conflitto a livello di istanza si riferisce alla discrepanza in un valore o istanza degli attributi, può essere un attribute conflict o un key conflict. Per gestire i conflitti a livello di istanza si può agire nel momento del design o nel momento della query. Per risolvere si usano "resolution function".

   Le regole di integrazione sono usate per specificare come i dati di diverse fonti devono trasformarsi quando si integrano. Le regole sono definite tra lo schema integrato e gli schemi delle fonti dati. È fondamentale per la risoluzione dei conflitti. Può riguardare la rinominazione, o la conversione di tipo dei dati.

- Differenze tra l'integration di due o di multiple tabels. L'integrazione può avvenire tra due tabelle o tra più fonti dati. Nel primo caso vuol dire integrare due fonti in un unica tabella applicando deduplication; nel secondo caso combinare più fonti in una tabella creano relazioni, spesso usa la normalizzazione.

Decuplicare in genere si riferisce a togliere rimuovere i duplicati, per farlo va fatto un data quality assesemente poi si procede con la data fusion.

Per integrare più tabelle è necessario aumentare la qualità con un "data quality improvement". Ci sono diverse tecniche che comprendono: data cleasing (identificare e correggere gli errori), data standardization (convertire i dati ad un formato comune), data enrichement (aggiungere più contesto ai dati), data deduplication (identificare e rimuovere record duplicati).

Una volta fatto è possibile procedere con la fase di "record linkage" che consiste nell'identificare records che si riferiscono alla stessa entità. Questo può essere difficoltoso e ci sono svariate tecniche per farlo:

- empirical: basata su anaisi statistiche dei dati, si basa sull assunzione che record che si riferiscono alla stessa entità abranno simele valore.
- probabilisitc: usa modellis statistici per stimare la probabilità che due record siano riferiti alla stessa entità. Come prima cosa si applica una "normalization" ovvero i record sono riorganizzati sotto un formato comune. Poi si applica una funzione di distanza e si decide una soglia per applicare la decisone.
- knowledge based: si basa su regole che devono applicarsi. Relativa a domini specifici e conoscenze specifiche.
- mixed perobabilistic e knowledge: usano modelli statistici e conoscenza specifica, sono di solito più accurate di quelle solo probabilistiche o knowledge based.

Una volta fusi i dati nella fase di "data fusion" è necessario gestire i conflitti. I conflitti possono essere gestiti in svariati modi: coflict ignoring (pass it on/ consider all possibilities), conflict avoiding (take the information/ no gosssiping/ trust uour frients), coflic resolution (deciding/ meditating).

DATA QUALITY

- Definizone Data Quality. La qualità è adattamento all'uso. La qualità dei dati inficia le decisioni fatte su di essi. La qualità non è solo sui record ma anche sugli attributi. Il mantra è 'garbage in garbage out'. È importante fidarsi della fonte dei dati. Il traid off della data quality è questione di "usefunlness" e "faithfulness". Qualità sono caratteristiche che si basano sull'abilità di soddisfare le necessità degli user. Una dimensione è una caratteristica specifica che descrive la qualità dell'informazione (di solito non misurabile). Le metriche sono un insieme di elementi che comprendono metodologie e unità di misura. È difficile dire quante misure ci siano.

I dati possono essere inaccurati (non corretti), incompleti (non pienamente rappresentativi) oppure inconsistenti (non concordano con altri dati). Le misure possono essere "oggettive" (formali e precise, indipendenti dalla percezione umana) o "soggettive" (dipendono dalla percezione).

DQ dimension in relational model

- Accuracy, con esempi per syntactic e sematic. L'accuratezza può essere definita su valori alfanumerici, su tuple, su relazioni, ma non su valori numerici. L'accuratezze di un valore v è la distanza tra v e v' considerato come il valore corretto che v vuole rappresentare. Dato che di solito v' è sconosciuto può essere costosa da calcolare. Laccuratezza può essere "sintattica" o "semantica".

Un dato si dice non sintatticamente accurato, quando esso non esiste nel dominio di referenza dei valori da lui assumibili. Un esempio di dato non sintatticamente accurato potrebbe essere: considerata una colonna "auto tedesche", il valore "Ferrari" non è sintatticamente corretto in quanto il termine Ferrari si riferisce ad un auto italiana. Ci sono due modi di calcolarla, i metodi basati sulle stringhe e quelli basati sui tiken. I metodi basati sulle stringhe usano la funzione di distanza che può essere normalizzata (EDnorm) o meno (UED).

L'accuratezza semantica è definita come il grado con cui i dati rappresentano correttamente i fatti del mondo reale. Un dato si dice non semanticamente accurato quando quest'utlimo non rappresenta correttamente i fatti del mondo reale. Per esempio se abbiamo una colonna con i nome dei dipendenti ed mettiamo il nome "Andrea Rossi" e poi scopriamo che il vero nome del dipendente è "Alessandro Rossi". Il dato "Andrea Rossi" sarebbe non semanticamente accurato.

- Desctizione di Completeness. La completezza può essere definita su valori alfanumerici, su tuple, su attributi, su relazioni, su tabelle e su un oggetti. La completezza (su tuple, attributi e tabelle) valuta i valori nulli e si definisce assumendo la "closed woeld assumption". Un ipotesi alternativa è assumere la "open world assumption" introducendo cosi la completezza su oggetti, che prende in considerazione che gli oggetti che possono essere rappresentati sono più che tuple.

- Dimensioni Time related, Dato aggiornato Ci sono due dimensioni temporali della data quality che sono "currency" e "timliness".

Currency si riferisce a quanto velocemente il dato è aggiornato tenendo conto del corrispondente reale. una prima Una prima misura della della currency è il ritardo temporale tra il tempo t1 dell' evento del mondo reale che ha provocato la variazione del dato, e l'istante t2 della sua registrazione nel sistema informativo. La currency quindi è rappresenta come la differenza tra tempo di arrivo alla organizzazione e tempo in cui è effettuato l'aggiornamento; essa è misurabile se c'è un log degli arrivi e degli update inoltre per valori con periodicità di aggiornamento nota la currency è calcolabile in maniera approssimata. Può essere visto anche come "ultimo aggiornamento".

La Timeliness invece misura il tempo che intercorre tra quando il dato è disponibile e quando il dato è utile. Al contrario della currency, è dipendente dal processo, ed è associata al momento temporale in cui deve essere disponibile per il processo che utilizza il dato per esempio possono esistere dati con elevata currency, ma ormai obsoleti per il processo che li usa.

- Consistency. Ci sono due sigificati di consistenza. Il primo si riferisce alla consistenza del dato rispetto ai costrutti di integrità definiti sullo schema (Zip code deve essere consistente con la città), il secondo è consistenza della differenza di rappresentazione dell'oggetto reale nel database (l'indirizzo deve essere rappresentato con lo stesso formato in tutto il database).

I vincoli di consistenza sono restrizioni applicate al database per assicurare che sia consistente e coerente con il sistema. Mentre le business rule sono stabilite dall'organizzazione che governa il data management e usa i dati.

ADVANCED DATA MANAGEMENT

- Big Data, con differenza da Open data. Big Data non sono solo dati, e c'è differenza da Open Data. I Big data si basono sulle 5V (Value, Volume, Velocity, Varaiety, Veracity) mentre gli opend data si basano su visibility e value. Analizzare i big data su un sicgolo server è impossibile, la soluzione sono architetture distribuite parallelamente. I sistemi paralleli possono avere svariati problemi. Come ad esempio :

- Syncronization: mancanza di coordinazione tra processi che causa conflitti.

- Deadlock: due o più processi si bloccano aspettando i risultati uno dall'altro, può crare un ciclo.
- Bandwith: la quantità di dati che può essere trasmessa attraverso un canale è un fattore limitante per le performance.
- Cordination: la coordinazione attraverso svariati processi, include la sincronizzazione.
- Failure: uno o più processi possono dare errore o spegnersi, il che può generare un effetto a cascata.

- Si descrivano le principali architetture per la distribuzione dei dati. L'architettura dei dati è un insieme di regole, politiche, standard e modelli che regolano e definiscono il tipo di dati raccolti e il modo in cui vengono utilizzati, archiviati, gestiti e integrati all'interno di un'organizzazione e dei suoi sistemi di database.

I primi database sono stati progettati per essere eseguiti su un singolo computer, i dati erano memorizzati e gestiti in un'unica posizione (database centralizzato). Tuttavia, l'efficienza di questo tipo di progettazione dipende dalla connettività della rete e dal traffico di dati e richieste, e inoltre l'efficienza generale del sistema è ridotta a causa della presenza di una sola copia del dato. Con questo tipo di progettazione non ci sono impostazioni di tolleranza al fallimento e in caso di errori hardware tutti i dati vengono persi.

Per questo motivo si è passati alla progettazione di Database distribuiti, i quali possono essere dislocati su più computer situati nello stesso luogo oppure distribuiti in una rete di computer connessi tra loro sotto forma di sistema distribuito. La distribuzione consiste nel dividere i dati e memorizzarli in posti diversi, provvedendo così ad avere possibilità di eseguire operazioni in parallelo. I database distribuiti permettono alle applicazioni di non conoscere nulla sulla dislocazione dei dati, ma accedervi come se fosse un database centralizzato. Nonostante i vantaggi, tra cui migliore disponibilità, autonomia e affidabilità,, i databse distribuiti presentano un'architettura molto complessa ed i costi sono molto elevati; inoltre hanno bisogno di una maggiore sicurezza. Un'altra criticità da considerare è che mancano gli standard per convertire un database centralizzato in un database distribuito. I database distribuiti possono essere omogenei e dunque tutti i siti hanno lo stesso software, sono a conoscenza gli uni degli altri e cooperano con il vantaggio di apparire come un unico sistema, oppure possono essere eterogenei e dunque siti differenti possono usare software e schemi differenti, rendendo problematica l'elaborazione delle query. I siti in un database distribuito eterogeneo possono ignorare la presenza degli altri siti e fornire solo strutture limitate per l'elaborazione delle richieste dell'utente.

- Architetture per l'elaborazione delle query distribuite/volume di dati, (HDSF, MapReduce, Handoop) Le architetture principali per l'elaborazione delle query sono:

- **HDFS**: L'architettura HDFS (Hadoop Distributed File System) è un file system distribuito progettato per girare su hardware base (commodity hardware). L'HDFS è particolarmente tollerante agli errori ed è progettato per girare su macchine poco costose, inoltre fornisce un accesso ad alta velocità ai dati delle applicazioni ed è ideale per applicazioni con data set di grandi dimensioni. HDFS ha un'architettura master/slave ed un cluster HDFS consiste in un singolo NameNode e in un server master che gestisce il file system detto NameSpace e che regola gli accessi ai file da parte dei clients inoltre sono anche presenti dei DataNodes generalmente uno per ogni nodo nel cluster, che gestiscono lo storage collegato ai nodi su cui vengono eseguiti. Il NameNode esegue le operazioni del file system NameSpace, ovvero apertura, chiusura e "rename" dei file e delle directories i noltre, determina la mappatura dei blocchi nei DataNodes, e reindirizza le richieste di operazioni di lettura e scrittura da parte del file system dei clients ai DataNodes proprio quest'ultimi permettono anche la creazione, l'eliminazione e la replica dei blocchi sotto istruzioni del NameNode. Esiste inoltre un Transaction Log che registra ogni operazione operazione avvenuta su un file.
- **Map Reduce**: Map Reduce è un motore di computazione distribuito dove ogni programma è scritto in stile funzionale ed è eseguito in parallelo. Prende spunto dalle funzioni "Mapcar" e "Reduce" del LISP, che applicano rispettivamente un'operazione su tutti gli elementi di un insieme e ne combinano i valori restituendone il risultato. L'architettura è di tipo master-slave dove il nodo master ha il compito di gestire la coda dei job, suddividere i work nei vari blocchi e notificarne il termine/fallimento invece i nodi slave gestiscono i singoli job, inviando al nodo master informazioni sullo svolgimento. Ogni job è caratterizzato dalle operazioni di map e di reduce. La fase di Map esegue lo stesso codice su un grande ammontare di record, i dati di input vengono trasformati in coppie chiave-valore e "filtrati" in un altro insieme di coppie chiave-valore. Questo secondo insieme viene restituito al framework, che esegue delle operazioni di shuffle sort, raggruppando gli elementi con la stessa chiave. La fase di Reduce aggrega i risultati intermedi (combinando gli elementi con la stessa chiave) e genera l'output.
- **Hadoop**: Hadoop è un framework che supporta applicazioni distribuite con elevato accesso ai dati, unisce il sistema MapReduce (parallel and distributed computation) e l'HDFS (hadoop storage and file system). Le caratteristiche principali sono che è un modello di storage scalabile ed economico. Man mano che i volumi di dati aumentano, aumenta anche

il costo dell'archiviazione di quei dati online.L' affidabilità viene garantita poichè i dati in Hadoop sono memorizzati in modo ridondante in più server e possono essere distribuiti su più rack di computer per questo un guasto di un server non si traduce in una perdita di dati semplicemente l'elaborazione passa ad un altro server. Inoltre è un modello di elaborazione scalabile ed : I dati possono essere caricati in Hadoop senza dover essere convertiti in un formato altamente strutturato e normalizzato.

- Descrizione e utilità di Apache Spark. Apache Spark è un general-purpose procssing engine. Il processing engine al posto di ammettere solo le operazioni map e reduce, definisce un grande numero di possibili operazioni. È un sofrware open source che supporta Jave, Scala e Python. Il costrutto chiave è RDD (Resilient Distributed Dataset). Spark supporta le analisi sui dati, machine learning, grafici e tanto altro. Può leggere e scrivere da svariati tipi di dati e permette lo sviluppo con diversi linguaggi.

Ci sono diversi componenti specializzati: Spark core (execution engine), Spark sql (permette un interfaccia di lavoro sql), Spark straming (processa dati in tempo reale), Spark MLlib (libreria per machne learning), e GraphX (permette analisi grafiche). Una delle caratteistiche principali di spark è la possibilità di eseguire task localmente o su un cluster di macchine, questo permette di scalare bene e gestire un workload ampio. Per accedere a dati esteerni spark usa Handoop InputFormat API.

Un dataframe è una struttura distribuita in Apache Spark che si organizza in colonne, simile al database relazionale. Ci sono svariati modi di costruire un Dataframe in Spark. Inoltre Spark miglio handoop inserendo Apache Spark Handoop 2.0 con svariati miglioramenti, supporto di Yarn. Apcache Accumulo, immagazzinamento chiave-valore distribuito permette di ottimizzare le perfomance; mentre Apache HBase, che è un colum-oriented database distribuito permette analisi in tempo reale.

- Si illustri il concetto di data lake i suoi componenti principali. Un data lake è una struttura capace di contenere un'enorme quantità di dati salvati in ogni formato in maniera non costosa; È un'infrastruttura in cui lo schema e i requisiti dei dati non sono definiti in partenza ma vengono definiti al momento dell'interrogazione: si tratta dello schema on-read, caratteristico di un approccio bottom up, caratterizzato dall'osservazione sperimentale. Pertanto i data lake acquisiscono e conservano dati non ancora elaborati per uno scopo specifico, il dato quindi non viene definito fino al momento in cui non viene eseguita una query che lo coinvolga. Il grande vantaggio è quello di archiviare dati con formati molto differenti senza necessità di doverli uniformare e "normalizzare" e da questo ne consegue che possono estrarre dati da qualunque fonte informativa senza che questa sia organizzata con strutture e caratteristiche definite a priori.

L'architettura di un datalake di solito consiste nelle seguenti componenti: Data Ingestion (collezziona da diverse fonti), Data Storage (mette le righe dei dati collezionati nel data lake), Data Processing (trasforma le righe in un formato che può eseguire query), Data Catalog (raccolta di metadati), Data Access (permette agli utenti l'accesso ai dati), e Data Governance (politiche e procedure del managing dei dati

L"approccio "destrutturato" del Data Lake garantisce una serie di vantaggi di natura operativa e gestionale:

- Ampliamento delle informazioni: i Data Lake ampliano in modo sensibile il numero di dati che gli analisti possono utilizzare e contemporaneamente , crescono anche le modalità di analisi che possono essere impiegate,trattandosi di dati non elaborati.

- Riduzione dei costi di archiviazione e gestione: i Data Lake consentono un notevole risparmio economico , la maggior libertà garantita da essi, infatti, dà modo di non dover definire in precedenza strutture (software e hardware) per l'archiviazione dei dati. Quindi organizzare e conservare le informazioni attraverso dei file system distribuiti, permette di abbattere i costi di gestione dell'intera infrastruttura.

- Riduzione dei tempi di analisi: all'interno del Data Lake, è possibile trovare i dati nella loro "forma naturale", senza che i dati siano stati alterati da altre analisi.

- Differenze e analogie tra data lake e data warehouse. Pur potendo sembrare simili, Data Lake e Data Warehouse sono estremamente differenti l'uno dall'altro sotto diversi punti di vista, infatti, si tratta di approcci antitetici alla gestione dei Big Data, che prendono le loro caratteristiche da strutture e obiettivi completamente differenti l'uno dall'altro.

- Struttura dei dati:nel Data Lake le informazioni non sono strutturate né elaborate al contrario, prima di poter essere immagazzinati in un Data Warehouse, i dati hanno bisogno di essere analizzati e strutturati, così da poter essere "inquadrati" all'interno di un telaio predefinito e "statico".

- Analisi dei dati: alla differente strutturazione delle informazioni corrisponde anche una loro differente analisi ,nei Data Lake, l'analisi avviene in un secondo momento, ossia quando vengono letti ed estratti dal "flusso" (di analisi "on read") invece nel Data Warehouse, invece, l'analisi è preliminare, in modo da poterli "adattare" alla struttura preesistente (analisi "on write").

- Finalità dei dati: Trattandosi di informazioni non elaborate, i dati vengono archiviati nel Data Lake senza alcuna finalità preimposta; le informazioni che confluiscono in un Data Warehouse hanno invece una finalità ben precisa e possono essere utilizzate solo per lo scopo pensato inizialmente.

L'unico, vero, punto di contatto tra Data Lake e Data Warehouse sta dunque nella loro funzione originaria: in entrambi i casi abbiamo a che fare con un approccio alla gestione dei Big Data, che dovrebbe consentire all'analista di poter ricavare nuovi insight e informazioni rilevanti per le loro attività. Nonostante si tratti di due approcci antitetici (che è in antitesi), Data Lake e Data Warehouse non si escludono a vicenda.