Basi di dati 💿

Appunti A.A. 21/22

Introduzione

Informazione, dati e loro gestione

Informatica = la scienza che studia i processi di rappresentazione, **gestione** e trasformazione dell'informazione e la loro automazione.

La gestione delle informazioni comprende diverse operazioni su queste:
- Raccolta, acquisizione
- Archiviazione, conservazione
- Elaborazione, trasformazione, produzione
- Distribuzione, comunicazione, scambio

Nelle attività umane le informazioni vengono gestite in forme diverse:
- idee informali
- **linguaggio naturale** (scritto o parlato, formale o colloquiale, in varie lingue)
- disegni, grafici, schemi
- numeri e codici

e su vari supporti come mente, carta, dispositivi elettronici.

Un **sistema informativo** è una componente di una organizzazione che gestisce le informazioni di interesse (cioè utilizzate per il perseguimento degli scopi dell'organizzazione). Ogni organizzazione ha un sistema informativo (automatizzato o meno).

Le informazioni vengono spesso rappresentate in modo essenziale, compatto, attraverso i dati.

Informazione = una notizia, elemento che consente di avere conoscenza più o meno esatta di fatti, situazioni, modi di essere.

Dato = ciò che è immediatamente presente alla conoscenza, prima di ogni elaborazione o anche elementi di informazione costituiti da simboli che debbono essere elaborati.

Il dato senza una interpretazione è inutile. I dati sono spesso il risultato di forme di organizzazione e codifica delle informazioni.

base di dati =
- Insieme organizzato di dati utilizzati per il supporto allo svolgimento delle attività di un ente (azienda, ufficio, persona)
- Insieme di dati gestito da un DBMS

I DBMS

Il DBMS (Database Management System) è un sistema che permette di gestire molte basi di dati.
Un DBMS è un sistema che gestisce collezioni di dati:
- **Grandi** → il limite deve essere solo quello fisico dei dispositivi. Le dimensioni sono (molto) maggiori della memoria centrale dei sistemi di calcolo utilizzati.

- **Persistenti** → hanno un tempo di vita indipendente dalle singole esecuzioni dei programmi che le utilizzano.
- **Condivise** → tra diversi utenti. Ogni organizzazione è divisa in settori o comunque svolge diverse attività. Ciascun settore/attività ha un sotto-sistema informativo, non necessariamente disgiunto (tanti sviluppatori devono poter usufruire di diverse basi di dati, collegamento con il server web dove si trova il DBMS che risponde alle richieste dei client). Una base di dati è una risorsa **integrata** e **condivisa** fra applicazioni. La condivisione implica:
  - attività diverse su dati condivisi (con conseguente meccanismo di **autorizzazione**)
  - accessi di più utenti ai dati condivisi (con conseguente controllo della **coerenza**)

Il DBMS garantisce:
- **Privatezza** → un DBMS permette un accesso **protetto** ai dati. Utenti diversi possono avere accesso a diverse porzioni della base di dati e possono essere abilitati a diverse operazioni su di esse. (Es. ad alcuni utenti può essere negato l'accesso a dati sensibili oppure possono leggere i dati e ma non modificarli).
- **Affidabilità** → resistenza a malfunzionamenti HW e SW, una base di dati deve essere conservata a lungo, deve essere persistente.
- **Efficienza** → utilizzare al meglio le risorse di spazio di memoria(principale e secondaria) e tempo (di esecuzione e di risposta). I DBMS con tante funzioni rischiano l'inefficienza.
- **Efficacia** → rendere produttive le attività dei loro utilizzatori, offrendo funzionalità articolate, potenti e flessibili.

I problemi che possono sorgere in una base di dati sono la **ridondanza** e il rischio di **incoerenza** .

I DBMS (di oggi) sono:
- Sistemi basati sul **modello relazionale**:
  - Soluzioni commerciali → DB2 (IBM), Oracle, SQLServer (Microsoft)
  - Soluzioni Open → MySQL, PostgreSQL, Access (Microsoft Office)
- Sistemi emergenti per applicazioni Internet basati su **modelli non relazionali** come chiave-valore, singola collana, grafo → HBase, CouchDB, Amazon Dynamo, Mongo DB, Neo4j: database nati per essere distribuiti su più macchine (utilizzo di Cluster).
- Sistemi relazionali **distribuiti** altamente scalabili → Google utilizza sistemi distribuiti per gestire la sua organizzazione interna e per le loro applicazioni e basi di dati. L'obiettivo di Google è quello delle advertising → gestione della pubblicizzazione

Transazione = è un **insieme di operazioni** da considerare indivisibile (atomico), corretto anche in presenza di concorrenza e con effetti definitivi.
Le transazioni sono:
- **Atomiche** → una sequenza di operazioni correlate deve essere eseguita **per intero** o per niente (es. il trasferimento di fondi da un conto A ad un conto B: o si esegue il prelevamento da A e il versamento su B o nessuno dei due).
- **Concorrenti** → bisogna mantenere **coerente** lo stato della base di dati anche di fronte a transazioni concorrenti (es. se due agenzie richiedono lo stesso posto (libero) sul treno si deve evitare di assegnarlo due volte).
- **Permanenti** → la conclusione positiva di una transazione corrisponde ad un impegno (**commit**) a **mantenere traccia** del risultato in modo definitivo, anche in presenza di guasti e di esecuzioni concorrenti.

Il blocco di istruzioni si concluderà con l'esito della transazione, che può essere:
- **Positivo** → si utilizza l'istruzione **commit** e significa che tutte le istruzioni del blocco sono state eseguite con successo.
- **Negativo** → si utilizza l'istruzione **rollback** che permette di ri-eseguire le operazioni al contrario fino a far ritornare il database nello stato precedente all'inizio della transazione.

In cosa si differenzia il DBMS dal file system?
DBMS vs File System→ La gestione di insiemi di dati grandi e persistenti è possibile anche attraverso sistemi più semplici, gli ordinari file system dei sistemi operativi.
- I file system però prevedono forme rudimentali di **condivisione**: "tutto o niente"; invece, i DBMS estendono le funzionalità dei file system, fornendo più servizi ed in maniera integrata.

- Oltre alla differenza nella condivisione dei dati questi software differiscono anche per la **descrizione** dei dati. Nei programmi tradizionali che accedono a file, ogni programma contiene una descrizione della struttura del file stesso, con i conseguenti rischi di incoerenza fra le descrizioni (ripetute in ciascun programma) e i file stessi. Nei DBMS, esiste una porzione della base di dati (il catalogo o dizionario) che contiene una descrizione centralizzata dei dati, che può essere utilizzata dai vari programmi.

I modelli dei dati

Rappresentando i dati a livelli diversi garantiamo l'**indipendenza** dei dati dalla **rappresentazione fisica**, i programmi fanno riferimento alla struttura a livello più alto, e le rappresentazioni sottostanti possono essere modificate senza modificare i programmi. Ciò viene fatto attraverso il concetto di modello dei dati.

Modello dei dati = E' un insieme di concetti utilizzati per organizzare i dati di interesse e descriverne la struttura. La componente fondamentale sono i meccanismi di strutturazione o **costruttori di tipo** (si può rappresentare come un insieme di tabelle). Come nei linguaggi di programmazione esistono meccanismi che permettono di definire nuovi tipi, così ogni modello dei dati prevede alcuni costruttori.

Per esempio, il modello relazionale prevede il costruttore relazione, che permette di definire insiemi di record omogenei.

Schema = è l'insieme di attributi di una tabella, descrive la struttura di solito lo schema resta invariato nel tempo.
Istanza = è una tupla, cioè una singola riga della tabella. sono i valori attuali e possono variare.

Identifichiamo 2 tipi principali di modelli:
- **Logici** → Adottati nei DBMS esistenti per l'organizzazione dei dati. Sono utilizzati dai programmi e sono indipendenti dalle strutture fisiche (es.: **relazionale**, chiave-valore, grafo, a oggetti, basato su XML).
- **Concettuali** → Permettono di rappresentare i dati in modo indipendente da ogni sistema. Cercano di descrivere i concetti del mondo reale e sono utilizzati nelle fasi preliminari di progettazione. Il più diffuso è il modello **entity-relationship**.

Architettura semplificata di un DBMS:

DB ←→ Schema interno ←→ Schema logico ←→ Utente.

Lo schema logico è una descrizione della base di dati nel modello logico(es. la struttura della tabella), lo schema interno o fisico invece è la rappresentazione dello schema logico per mezzo di strutture di memorizzazione (es. i record con puntatori ordinati in un certo modo).

Architettura a 3 livelli per DBMS:

DB←→ schema interno ←→ schema logico ←→ schemi esterni (viste)

L'architettura a 3 livelli definisce 3 schemi:
- **esterno**→ descrizione di parte della base di dati in un modello logico ("**viste**" parziali, derivate, anche in modelli diversi)
- **logico** → descrizione dell'intera base di dati nel modello logico "principale" del DBMS
- **interno** → (o fisico) rappresentazione dello schema logico per mezzo di strutture fisiche di memorizzazione e di accesso veloce.

L'**indipendenza dei dati** è una proprietà desiderabile e importante per un DBMS. L'articolazione in livelli facilita la sua realizzazione. L'accesso ai dati avviene solo tramite il livello esterno (che può coincidere con il livello logico). Ci sono 2 forme di indipendenza:
- **fisica**→ Si ha indipendenza fisica quando è possibile cambiare lo schema interno senza cambiare lo schema logico. Permette ad esempio di migliorare le prestazioni di accesso a

determinati dati modificando il metodo di memorizzazione dei dati senza modificare le tabelle (schema logico).
- **logica**→ Si ha indipendenza logica quando lo schema logico può essere modificato senza modificare gli schemi esterni. Es.: si può estendere lo schema logico con nuovi attributi per una tabella o con nuove tabelle senza dover cambiare le "viste". Un DBMS che supporta indipendenza logica deve però limitare certe operazioni, infatti, le viste devono rimanere "possibili", cioè non si può eliminare a livello logico un attributo di una vista non calcolabile in altro modo.

Un DBMS che supporta indipendenza logica e fisica permette modifiche ai 2 livelli: modifica automaticamente i mapping da un livello all'altro e garantisce che le applicazioni che accedono ai dati tramite schemi esterni continuino a funzionare. Nella pratica pochi DBMS hanno implementato completamente questa architettura.

Linguaggi per basi di dati

Per i DBMS relazionali lo standard è SQL. SQL è un linguaggio utilizzato in diversi modi:
- testuale interattivo → da terminale
- comandi SQL immersi in un linguaggio ospite → PHP , pascal, java, c...
- comandi SQL immersi in un linguaggio ad hoc, con altre funzionalità→ Oracle PL / SQL
- implicito con interfacce amichevoli(senza linguaggio testuale) → MS access

Esempio query:
- **SELECT** Corso, Aula, Piano
- **FROM** Aule, Corsi
- **WHERE** Nome = "Aula" **AND** Piano = "Terra"

Ci sono 2 tipi di linguaggi, entrambi implementati da SQL:
Data Manipulation Language = (DML) per l'interrogazione e l'aggiornamento di (istanze di) basi di dati.
Data Definition Language = (DDL) per la definizione di schemi (logici, esterni, fisici) e altre operazioni.
SQL include istruzioni DML e DDL.

Personaggi e interpreti:
- progettisti e realizzatori di DBMS
- progettisti della base di dati e amministratori della base di dati
  - Data Base Administrator = Persona o gruppo di persone responsabile del controllo centralizzato e della gestione del sistema, delle prestazioni, dell'affidabilità, delle autorizzazioni. Le funzioni del DBA includono quelle di progettazione, anche se in progetti complessi ci possono essere distinzioni.
- progettisti e programmatori di applicazioni
- utenti

Vantaggi e svantaggi dei DBMS

\+ dati come risorsa comune, base di dati come modello della realtà
\+ gestione centralizzata con possibilità di standardizzazione ed "economia di scala"
\+ disponibilità di servizi integrati
\+ riduzione di ridondanze e inconsistenze
\+ indipendenza dei dati (favorisce lo sviluppo e la manutenzione delle applicazioni)

\- costo dei prodotti e della transizione verso di essi
\- non scorporabilità delle funzionalità (con riduzione di efficienza)

//Perché usare un DBMS?

Il Web e l'accesso ai DBMS

Modello client/server

Le applicazioni usano API per accedere al DBMS. il funzionamento del paradigma client/server è semplice:
- il client richiede una pagina,
- il server riceve la richiesta e invia il file al client,
- il browser riceve e interpreta il file visualizzando la pagina web.

Script = Oltre al semplice trasferimento di informazioni, viene eseguito del codice, lato:
- client → sono in JavaScript e permettono di eseguire diverse visualizzazioni in base ad azioni o eventi dell'utente.
- server → il web server memorizza un file che contiene il file HTML e il codice dello script. Quando la pagina viene richiesta, il server esegue il codice di script che genera altre informazioni di script e invia il risultato al client.

Architettura WEB-DB multi-tier

Uno script lato server può interagire con una base di dati per leggere e/o scrivere dati.
3-tier → web client, web server, DB:
- il web client invia la richiesta al web server
- il web server invia la richiesta al DB
- il DB invia la risposta alla richiesta al web server
- il web server invia la risposta alla web client

Web application e DBMS

Applicazione eseguita dentro un Internet browser:
- segue il paradigma client/server appoggiandosi ad un web server ed eseguendo script locale
- per migliorare i tempi di risposta aggiorna indipendentemente parti diverse di una pagina
- permette una limitata memorizzazione di dati in locale (di solito i dati sono gestiti lato server)
- non necessita installazione (viene caricata come una pagina web)
- altamente portabile e adattabile a dispositivi client diversi

Mobile app e DBMS

→ applicazione scritta in codice nativo per una certa piattaforma (Android, etc) che si appoggia a servizi esterni via HTTP. Differenze da una web app:
- aspetto grafico tipico della piattaforma (miglior interfaccia utente)
- maggiore reattività
- accesso a componenti HW del dispositivo e a un DBMS/datastore locale

Il modello relazionale

Esistono 3 modelli logici (i programmatori fanno le interrogazioni sul modello logico) tradizionali:
- gerarchico
- reticolare
- relazionale

Gerarchico e **reticolare** utilizzano riferimenti espliciti (puntatori fra record), mentre il **relazionale è basato su valori**, i riferimenti fra dati in relazioni diverse sono rappresentati per mezzo di valori dei domini che compaiono nelle ennuple. Vengono confrontati valori in colonne che semanticamente sono corrispondenti.
Il modello relazionale è stato proposto per favorire l'indipendenza dei dati e si basa sul concetto matematico di relazione (con una variante). Le relazioni hanno naturale rappresentazione per mezzo di tabelle.
Una relazione si definisce:
- relazione **matematica** → teoria degli insiemi. Ogni relazione è un insieme → non c'è ordinamento fra le n-uple. Le n-uple sono distinte e ciascuna n-upla è ordinata: l'i-esimo valore proviene dall'i-esimo dominio.
- relazione secondo il modello relazionale dei dati
- relazione che rappresenta una classe di fatti, nel modello Entity-Relationship; tradotto anche con associazione o correlazione.

Prodotto cartesiano di due insiemi = insieme di tutte le coppie che si possono ottenere. Il primo elemento della coppia viene dal primo dominio, e il secondo elemento della coppia dal secondo.
La relazione è un sottoinsieme del prodotto cartesiano dei due insiemi. $r \subseteq D_1 \times D_2$

Lo schema relazionale definisce i domini associando loro un nome unico nella tabella (attributo) che ne descrive il ruolo. La struttura di ogni dominio è non posizionale.

In una tabella che rappresenta una relazione l'ordinamento tra le righe/colonne è irrilevante.
Una tabella rappresenta una relazione se:
- le righe sono diverse fra loro,
- le intestazioni delle colonne sono diverse tra loro e
- i valori di ogni colonna sono fra loro omogenei e, all'interno di essa,

Vantaggi del modello relazionale:
- indipendenza dalle strutture fisiche che possono cambiare dinamicamente
- si rappresenta solo ciò che è rilevante dal punto di vista dell'applicazione
- l'utente finale vede gli stessi dati dei programmatori
- i dati sono portabili più facilmente da un sistema ad un altro
- i puntatori sono direzionali

Schema di relazione = ha nome R e un insieme di attributi $R(A_1, \ldots, A_n)$

Schema di base di dati = insieme di schemi di relazione $R = \{R_1(X_1), \ldots, R_n(X_n)\}$

Una **ennupla** su un insieme di attributi X è una funzione che associa a ciascun attributo A in X un valore del dominio di A. t[A] denota il valore della ennupla t sull'attributo A .

Informazioni incomplete = non è detto che ci siano tutti i dati richiesti per la realizzazione del DB. Il modello relazionale impone ai dati una struttura rigida:
- le informazioni sono rappresentate per mezzo di ennuple
- solo alcuni formati di ennuple sono ammessi: quelli che corrispondono agli schemi di relazione

I dati disponibili possono non corrispondere al formato previsto.

Si utilizza il **valore nullo NULL** che denota l'assenza di un valore del dominio (ma non è un valore del dominio), quindi, t[A], per ogni attributo A, è un valore del dominio dom(A) oppure il valore nullo.

Il valore nullo può rappresentare un valore:
- sconosciuto → sappiamo che esiste il dato, ma non lo conosciamo
- inesistente → sappiamo che per quel record/tupla non esiste il dato, non c'è (es. nome della moglie/del marito di un utente che non è sposato)
- senza informazione → non sappiamo se c'è il dato e non lo conosciamo (es. secondo nome di qualcuno)

I DBMS non distinguono i tipi di valore nullo.

Vincoli d'integrità = è una proprietà che deve essere soddisfatta dalle istanze di una base di dati. Ogni vincolo può essere visto come un predicato che può assumere il valore vero o falso: l'istanza soddisfa il vincolo, se il predicato assume il valore vero.
In generale a uno schema di DB si associa un insieme di vincoli e si considerano corrette (lecite o ammissibili) solo le istanze che soddisfano tutti i vincoli predefiniti.
I vincoli d'integrità sono utili nella **progettazione** e vengono usati dai DBMS nell'esecuzione delle **interrogazioni**. I vincoli corrispondono a proprietà del mondo reale modellato dalla base di dati.

Interessano a livello di schema (in riferimento a tutte le istanze); ad uno schema associamo un insieme di vincoli e consideriamo corrette (valide, ammissibili) le istanze che soddisfano tutti i vincoli (le istanze potrebbero soddisfare anche altri vincoli "per caso", momentanei).

Esistono due grandi tipologie di vincoli: quelli intra-relazionali che interessano una sola tabella e quelli inter-relazionali che definiscono legami tra due o più tabelle.
- **Intra-relazionali** → vincoli su valori (o di dominio), vincoli di ennupla. I vincoli di ennupla esprimono condizioni sui valori di ciascuna ennupla indipendentemente dalle altre ennuple, mentre i vincoli di dominio coinvolgono un solo attributo. Sintassi → espressione booleana di condizioni che confrontano valori di attributo o espressioni aritmetiche su di essi. Es. (Voto ≥ 18) AND (Voto ≤ 30)
- **Inter-relazionali** → Vincoli di integrità referenziale su piu relazioni, che quindi impediscono l'introduzione di inconsistenza dei dati in una tupla che dipende dalla presenza o meno di un'altra tupla in un'altra tabella.

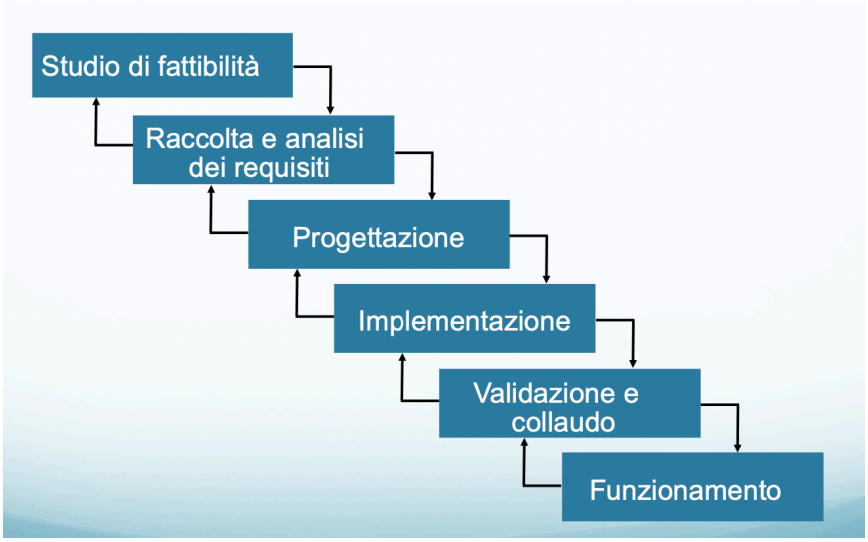
Chiave e superchiave = Insieme di attributi che identificano univocamente le tuple di una relazione.
Un insieme K di attributo è **superchiave** per r se r non contiene due ennuple distinte t1 e t2 con t1[K] = t2[K].
K è **chiave** per r se è una **superchiave minimale** per r (cioè non contiene un'altra superchiave).

L'esistenza delle chiavi garantisce l'accessibilità a ciascun dato del DB. Le chiavi permettono di correlare i dati in relazioni diverse (modello relazionale è basato su valori). In presenza di valori nulli, i valori della chiave non permettono di identificare le tuple e di realizzare facilmente i riferimenti da altre relazioni.

Chiave primaria = La chiave primaria è una chiave che non ammette valori nulli e usa come notazione la sottolineatura.
informazioni in relazioni diverse sono correlate attraverso valori comuni, in particolare, valori delle chiavi (primarie). Le correlazioni devono essere coerenti.
Un vincolo di integrità referenziale (**foreign key**) fra gli attributi X di una relazione R1 e un'altra relazione R2 impone ai valori su X in R1 di comparire come valori della chiave primaria di R2.
I **vincoli di integrità referenziale** possono essere resi meno restrittivi in presenza di valori nulli e possono essere utilizzati meccanismi per il supporto alla loro gestione (tramite azione compensative a seguito di violazioni come eliminazione a cascata → eliminando un dato referenziato, bisogna eliminare anche la referenza, o introduzione di valori nulli).

Progettazione di basi di dati: metodologie e modelli

La progettazione di basi di dati è un'attività che fa parte del processo di sviluppo dei sistemi informativi. Si parla quindi, di ciclo di vita dei sistemi informativi, cioè delle attività svolte nello sviluppo e nell'uso di essi e attività iterative(revisioni). Queste attività non sono quasi mai strettamente sequenziali perchè spesso durante l'esecuzione di un'attività si ha bisogno di modificarne una precedente, ottenendo un ciclo di operazioni.



Ciclo di vita di un sistema informativo:

- **Studio di fattibilità** → definiti costi delle varie alternative e priorità di realizzazione dei componenti
- **Raccolta e analisi dei requisiti** → sono studiate le proprietà e le funzionalità che avrà il sistema e vengono stabiliti i requisiti HW e SW del sistema informativo. Questa fase produce un documento informale contenente tutti i dati e le operazioni su essi.
- **Progettazione** → si divide in 2: progettazione dati e progettazione applicazioni. Nella prima si individua la struttura e l'organizzazione che i dati avranno, nell'altra si definiscono le caratteristiche dei prog. applicativi. queste due attività sono complementari e possono essere eseguite in parallelo o in cascata.
- **Implementazione** → realizzazione dei sistema info secondo la struttura e le caratteristiche definite formalmente nella fase precedente. la base di dati viene costruita e popolata.
- Validazione e collaudo → viene verificato il corretto funzionamento e la qualità. Vengono svolte delle sperimentazioni che devono prevedere tutte le condizioni operative.
- **Funzionamento** → il sistema informativo diventa operativo ed esegue i compiti richiesti. Se non ci sono errori di funzionamento questa attività richiede solo manutenzione e gestione. Oltre alle attività citate talvolta può essere aggiunta quella di prototipizzazione che consiste nell'uso di specifici strumenti SW per realizzare rapidamente una versione semplificata del sistema per sperimentare le sue funzionalità. Anche con la verifica del prototipo potrebbe esserci la necessità di effettuare revisioni.

Dal momento che i dati hanno un ruolo centrale nel sistema informativo ci focalizziamo ad analizzare la seconda e terza fase: *Analisi dei requisiti* e *Progettazione*.

Per garantire un prodotto di buona qualità bisogna seguire una Metodologia di progettazione. Inizialmente avviene una decomposizione dell'intera attività di progetto in fasi indipendenti, si definiscono una serie di strategie e criteri per la scelta tra le varie alternative ed infine tramite alcuni modelli di riferimento si descrivono i dati in input e output delle varie fasi. Le proprietà di una metodologia sono:

- generalità rispetto alle applicazioni e ai sistemi
- qualità del prodotto rispetto alle risorse
- facilità d'uso delle strategie e dei modelli di riferimento

Per le basi di dati, in modo specifico, viene utilizzata una metodologia di progetto che si articola in 3 fasi a cascata in cui si separano le decisioni su “cosa” rappresentare (concettuale) e su “come” farlo (logico e fisico). Progettazione:

1. **concettuale** → viene rappresentato il contenuto informativo della base di dati, le specifiche della realtà d’interesse, le classi di oggetti con le relative correlazioni senza considerare gli aspetti implementativi. Questo livello produce un documento chiamato schema concettuale che ha una grande efficacia grafica.
2. **logica** → lo schema concettuale viene tradotto in schema logico che consiste nella descrizione dei dati ancora indipendenti da dettagli fisici. Le scelte progettuali si basano su criteri di ottimizzazione delle operazioni sui dati. Si usano tecniche formali per verificare la qualità dello schema logico ottenuto; nel modello relazionale la tecnica utilizzata è detta normalizzazione .
3. **fisica** → lo schema logico viene completato con la specifica dei parametri fisici di memorizzazione dei dati.

Nella progettazione concettuale si fa uso delle specifiche sui dati, mentre le specifiche sulle operazioni servono solo a verificare che lo schema concettuale sia completo e in grado di fornire le info necessarie per eseguire tutte le operazioni.

Nella progettazione logica, lo schema concettuale in ingresso riassume le specifiche sui dati, mentre si utilizzano le specifiche sulle operazioni insieme alle previsioni di carico applicativo, per ottenere uno schema logico che renda queste operazioni efficienti.

Nella progettazione fisica si usa lo schema logico e le specifiche sulle operazioni per ottimizzare le prestazioni, tenendo conto delle caratteristiche del DBMS adottato.

Modello E/R (Entità - Relazione)

Il modello E/R è il modello concettuale di dati più utilizzato, che fornisce una serie di strutture (costrutti) che descrivono la realtà d'interesse facile da comprendere. I costrutti servono a definire degli schemi che descrivono l'organizzazione e la struttura delle occorrenze dei dati. Questo modello mette a disposizione una serie di costrutti e per ognuno di esso una rappresentazione grafica opportuna.

Entità = rappresenta la realtà di interesse ed è una classe di oggetti che hanno proprietà comuni ed esistenza autonoma. Un'occorrenza di un'entità è un oggetto della classe che quell'entità rappresenta, ad esempio "Roma" e "Milano" sono occorrenze dell'entità "Città". In uno schema, ogni entità esiste indipendentemente dalle sue proprietà, ha un nome (singolare) che la identifica univocamente e viene rappresentata graficamente tramite rettangolo.

Relazione = rappresenta un legame logico fra 2 o più entità per esempio "residenza" tra "persona" e "città". Una occorrenza di relazione è una ennupla fatta di occorrenze di entità, una per ciascuna entità coinvolta. Ogni relazione ha un nome che la identifica univocamente e viene rappresentata graficamente tramite rombo con il nome all'interno e linee che connettono la relazione con ciascuna delle sue componenti. L'insieme delle occorrenze di una relazione nel modello E/R è una relazione matematica tra le occorrenze delle entità coinvolte cioè un sottoinsieme del loro prodotto cartesiano. Tra le occorrenze di una relazione non ci possono essere ennuple ripetute.

Ci sono diversi tipi di relazione:

- Binarie → una relazione tra esclusivamente due entità.
- Ricorsive → una relazione tra un'entità e se stessa.
- Multiple (ennarie) → una relazione che coinvolge più di due entità.

Attributi = sono proprietà che descrivono le entità o le relazioni a cui appartengono e derivano dalla realtà di interesse. Vengono rappresentati graficamente legandoli con delle linee alle relazioni/entità e definite con pallini vuoti. Sono identificati dal nome, dal tipo e dal range di valori che possono assumere. Uno o più attributi candidati sono quegli attributi che potrebbero diventare chiave primaria.

Tipologie di attributi:

- Semplici → hanno un tipo semplice, sono atomici e univoci.
- Composti → sono suddivisibili in cui esistono sotto attributi che li compongono (es. data → gg,mm,aa)
- Multipli → possono assumere un certo insieme di valori in un intervallo ben determinato.

Cardinalità di una relazione = definisce il numero minimo e massimo di istanze con cui un'entità può partecipare in una relazione. La cardinalità minima: = 0 indica che l'entità partecipa in modo opzionale alla relazione, mentre > 0 l'entità partecipa in modo obbligatorio. E' possibile assegnare un qualunque intero non negativo alla cardinalità di una relazione con l'unico vincolo che la cardinalità minima sia < o = di quella massima.

Tipologie classiche di cardinalità delle relazioni:

- 1 a 1 → a un istanza della prima entità corrisponde una e una sola istanza della seconda entità e viceversa
- 1 a N → a un istanza della prima entità corrispondono N istanze della seconda entità
- N a N → a ogni istanza della prima entità possono corrispondere N istanze della seconda entità e viceversa.

cardinalità degli attributi = descrive il numero minimo e massimo di valori dell'attributo associati a ogni occorrenza di entità o relazione. Quando la cardinalità è 1 a 1 viene omessa e vuol dire che associa a ogni occorrenza di entità un solo valore dell'attributo. Un attributo con cardinalità minima pari a 0 è opzionale per la relativa entità o relazione, mentre è obbligatorio se la cardinalità minima è pari a 1. Un attributo è multivalore se la sua cardinalità massima è pari a N.

Identificatore di un entità = ha il compito di identificare in maniera univoca le occorrenze delle entità. Per individuare un identificatore bastano uno o più attributi di una certa entità → identificatore interno . Quando gli attributi di un entità non sono sufficienti ad identificare le sue occorrenze, si combinano gli attributi interni con quelli di un'altra entità → identificatore esterno.

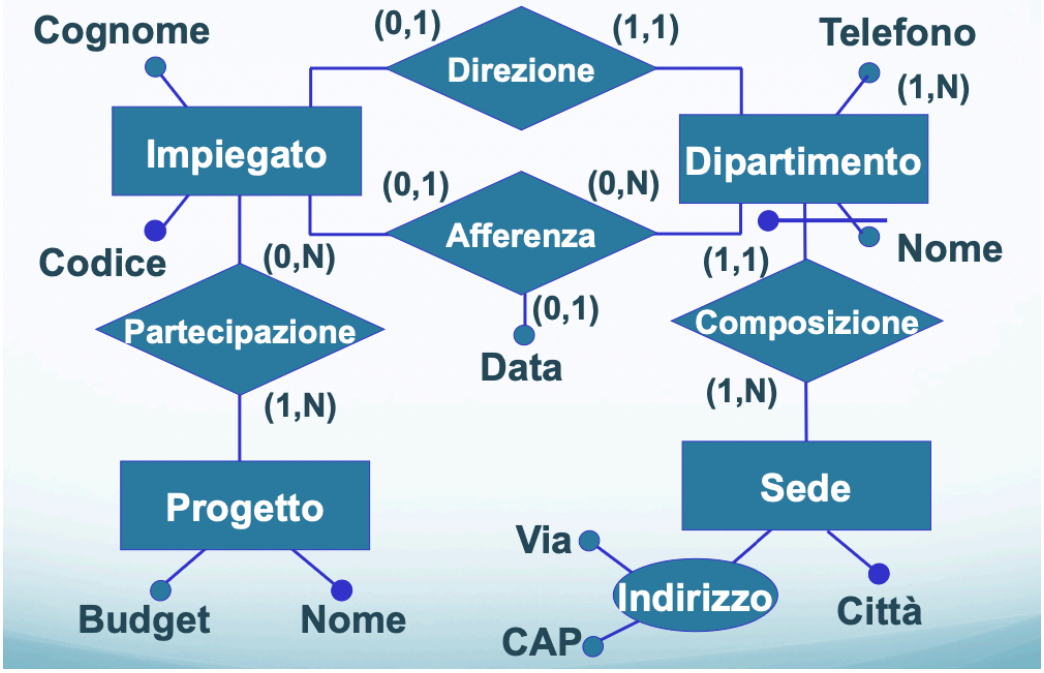
Un identificatore può coinvolgere uno o più attributi, ognuno dei quali deve avere cardinalità (1,1). Un'identificazione esterna può coinvolgere una o più entità, ognuna delle quali deve essere membro di una relazione alla quale l'entità da identificare partecipa con cardinalità (1,1). Essa può anche coinvolgere un entità che è a sua volta identificata esternamente, basta che non siano generati cicli di identificazioni esterne. Ogni entità deve avere almeno un identificatore ma ne può avere più di uno.

Generalizzazioni = rappresentano legami logici tra un entità genitore e uno o più entità figlie. Si dice che il genitore è una generalizzazione delle figlie mentre le figlie sono una specializzazione del genitore.

Proprietà delle generalizzazioni:
- Ogni occorrenza di un entità figlia è anche una occorrenza delle entità genitore.
- Ogni proprietà dell'entità genitore è anche proprietà delle entità figlie (**ereditarietà**).
- Possono esistere gerarchie a più livelli e multiple generalizzazioni allo stesso livello
- Un'entità può essere inclusa in più gerarchie, come genitore e/o come figlia
- Se una generalizzazione ha solo un'entità figlia si parla di **sottoinsieme**
- Una generalizzazione è totale se ogni occorrenza dell'entità genitore è una occorrenza di almeno
- una delle entità figlie, altrimenti è parziale
- Una generalizzazione è esclusiva se ogni occorrenza dell'entità genitore è al più una occorrenza di una delle entità figlie, altrimenti è sovrapposta

Lo schema E/R deve essere corredato da una **documentazione** di supporto che serve a facilitare l'interpretazione dello schema stesso e a descrivere proprietà dei dati non esprimili con i costrutti forniti dal modello E/R. Uno degli strumenti più utilizzati dagli analisti come documentazione integrata sono le regole aziendali che possono essere:
- *Descrizione di un concetto* → definizione precisa di un entità, attributo o relazione del modello E/R
- *Vincolo d'integrità* → documentazione di un vincolo espresso con qualche costrutto del modello E/R o la descrizione di un vincolo non esprimibile con i costrutti del modello E/R
- *Derivazione* → concetto che può essere ottenuto da altri concetti dello schema, attraverso un inferenza o un calcolo aritmetico

Le regole aziendali di tipo descrittivo possono essere prodotte facendo uso del dizionario dei dati che è composto dalla tabelle delle entità e dalla tabella delle relazioni. In aggiunta a queste due, si può fare ricorso ad un'altra tabella in cui troviamo i vincoli non esprimibili col modello E/R.



Vincoli non esprimibili

Vincoli di integrità sui dati
(1) Il direttore di un dipartimento deve afferire a tale dipartimento
(2) Un impiegato non deve avere uno stipendio maggiore del direttore del dipartimento al quale afferisce
(3) Un dipartimento con sede a Milano deve essere diretto da un impiegato con più di dieci anni di anzianità
(4) Un impiegato che non afferisce a nessun dipartimento non deve partecipare a nessun progetto

Dizionario dei dati (entità)

Entità	Descrizione	Attributi	Identificatore
Impiegato	Dipendente dell'azienda	Codice, Cognome, Stipendio	Codice
Progetto	Progetti aziendali	Nome, Budget	Nome
Dipartimento	Struttura aziendale	Nome, Telefono	Nome, Sede
Sede	Sede dell'azienda	Città, Indirizzo	Città

Vincoli non esprimibili

Regole di derivazione
Il budget di un progetto si ottiene moltiplicando per 3 la somma degli stipendi degli impiegati che vi partecipano

Dizionario dei dati (associazioni)

Relazioni	Descrizione	Componenti	Attributi
Direzione	Direzione di un dipartimento	Impiegato, Dipartimento	
Afferenza	Afferenza a un dipartimento	Impiegato, Dipartimento	Data
Partecipazione	Partecipazione a un progetto	Impiegato, Progetto	
Composizione	Composizione dell'azienda	Dipartimento, Sede	

Raccolta e analisi dei requisiti

Requisiti	Regole generali
Documentazione descrittiva	- scegliere il corretto livello di astrazione - standardizzare la struttura delle frasi - suddividere le frasi articolate - separare le frasi sui dati da quelle sulle funzioni
Organizzazione di termini e concetti	- individuare omonimi e sinonimi e unificare i termini - rendere esplicito il riferimento fra termini - riorganizzare le frasi per concetti - costruire un glossario dei termini

Esempio: specifica di biblioteche

I lettori che frequentano la biblioteca hanno una tessera su cui è scritto il nome e l'indirizzo ed effettuano richieste di prestito per i libri che sono catalogati nella biblioteca.

I libri hanno un titolo, una lista di autori e possono esistere in diverse copie.

Tutti i libri contenuti nella biblioteca sono identificati da un codice. A seguito di una richiesta viene dapprima consultato l'archivio dei libri disponibili (cioè non in prestito). Se il libro è disponibile, si procede alla ricerca del volume negli scaffali;

il testo viene poi classificato come in prestito. Acquisito il volume, viene consegnato all'utente, che procede alla consultazione. Terminata la consultazione, il libro viene restituito, reinserito in biblioteca e nuovamente classificato come disponibile. Per un prestito si tiene nota degli orari e delle date di acquisizione e di riconsegna.

Analisi:

1. trovare i sinonimi, ovvero termini diversi con uguale significato → Copia – Libro – Testo - Volume
2. Separare le frasi
3. Definizione di un glossario che spiega le entità coinvolte, una descrizione di esse, il loro sinonimo (se presente) e i collegamenti con altre entità.

Termine	Descrizione	Sinonimo	Collegamenti
Lettore	Una persona che prende in prestito libri dalla biblioteca	Utente	Copia, Prestito
Libro	Tipo di libro presente in biblioteca. La biblioteca ha una o più copie di uno stesso libro.		Copia
Copia	Ogni copia di un libro presente in biblioteca. Può essere prestato a un lettore.	Libro, Testo, Volume	Libro, Lettore, Prestito
Prestito	Un prestito fatto a un lettore: ogni prestito si riferisce ad una copia di un libro.		Lettore, Copia

Quale costrutto E-R va utilizzato per rappresentare un concetto presente nelle specifiche?
Bisogna basarsi sulle definizioni dei costrutti del modello E-R:

- **Entità**: se un concetto ha proprietà significative e/o descrive classi di oggetti con esistenza autonoma
- **Attributo**: se un concetto ha una struttura semplice e non ha proprietà rilevanti associate è opportune rappresentarlo con un attributo di un altro concetto (entità/relazione) a cui si riferisce.
- **Relationship**: se sono state individuate due o più entità e nei requisiti compare un concetto che le associa, questo concetto può essere rappresentato da una relazione.
- **Generalizzazione**: se uno o più concetti risultano essere casi particolari di un altro, è opportuno rappresentarlo facendo uso di una generalizzazione

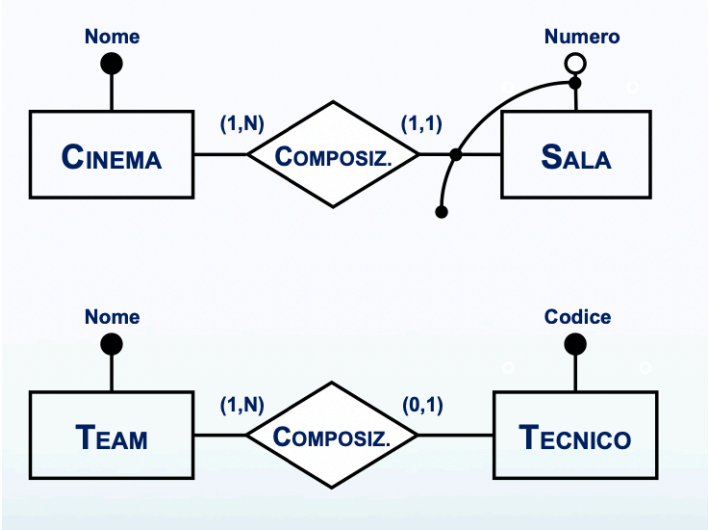
Design pattern = sono soluzioni progettuali a problemi comuni. Largamente usati nell'ingegneria del software.

Pattern di progetto

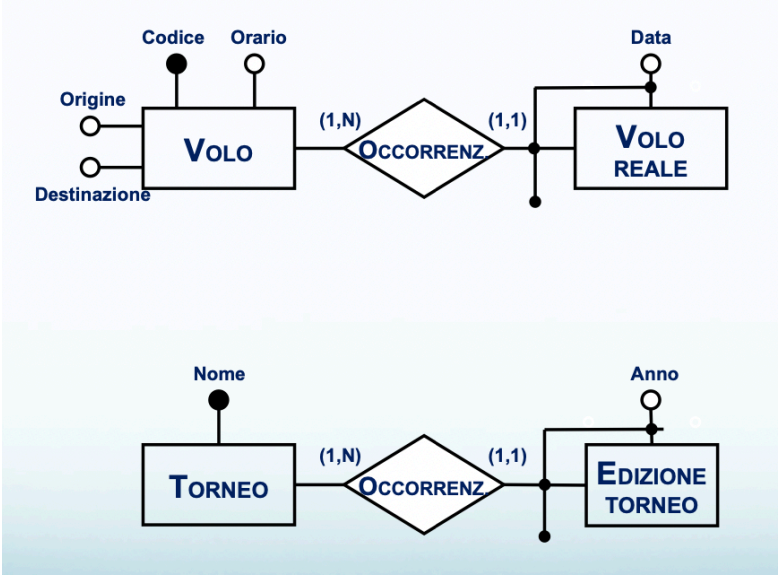
Vediamo alcuni pattern (di progetto) comuni nella progettazione concettuale di basi di dati.

Part-of = si può presentare in due forme:

- può esistere un occorrenza dell'entità che dipende dall'esistenza di una occorrenza dell'entità che la contiene e richiede un identificazione esterna (cinema e sala)
- l'entità contenuta nell'altra ha esistenza autonoma, in cui la partecipazione alla relazione è opzionale.

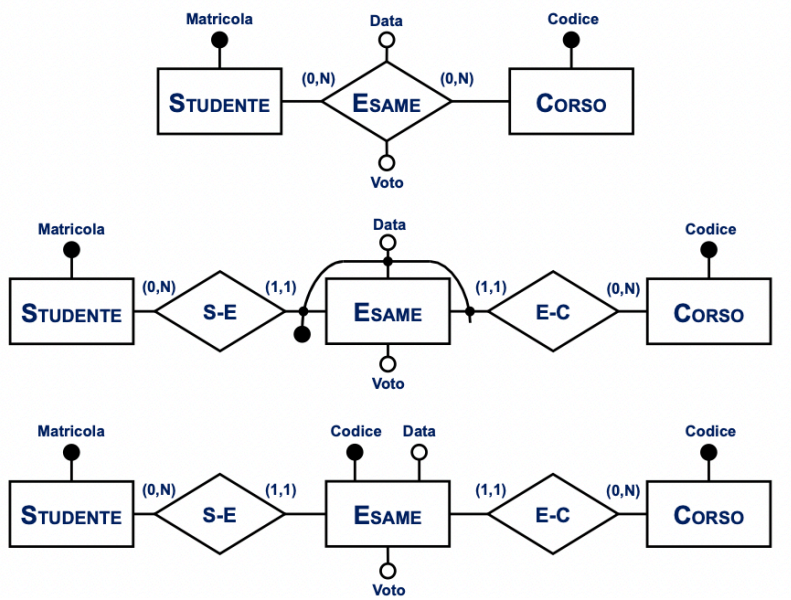


Instance-of = le occorrenze di un entità della relazione sono istanze di occorrenze dell'altra entità.



Pattern di progetto che coinvolgono relazioni

Reificazione di una relazione = Nel caso in cui si utilizza una relazione per descrivere un concetto che lega altri due concetti viene svolta una reificazione della relazione ovvero viene trasformata in una entità. (Es. di reificazione di associazione binaria: Studente → Esame → Corso; Esame diventa un'entità).



Reificazione di relazione ricorsiva = Nel caso in cui ci sia una relazione ricorsiva su un'entità e di questa relazione si vogliono avere più istanze con le stesse occorrenze dell'entità, è opportuno reificare questa relazione trasformandola quindi in entità e aggiungendo le opportune relazione con l'entità appena creata.

Reificazione di attributo di relazione = Nel caso in cui ci sia un attributo su una relazione che esprime un concetto rilevante per l'applicazione bisogna reificare l'attributo della relazione e la relazione stessa.

Reificazione della relazione ternaria = Nel caso in cui si presenta una relazione ternaria si può reificarla facendola diventare entità ed inserendo tre relazioni.

Pattern di progetto che coinvolgono generalizzazioni

Storicizzazione di concetto = Da un'entità padre vengono create due entità figli con gli stessi attributi, una delle figlie tiene conto dello storico e l'altra rappresenta il concetto di interesse con le informazioni aggiornate.
Storicizzazione di relazione = Per storicizzare una relazione bisogna reificarla utilizzando una generalizzazione.
Evoluzione di concetto = Un certo concetto può subire una certa evoluzione nel tempo che può essere diversa per l'evoluzione del concetto. Ad esempio, ci possono essere dei progetti che vengono proposti per ottenere dei finanziamenti, non tutti finiranno tra quelli accettati che avranno informazioni aggiuntive (finanziamento e data inizio).

Strategie di progetto

Top-Down = Nella metodologia top-down, lo schema concettuale viene prodotto con una serie di raffinamenti successivi. Partendo da uno schema iniziale che descrive tutte le specifiche con pochi concetti molto astratti. Nel passaggio da un livello di raffinamento all'altro vengono eseguite alcune trasformazioni elementari chiamate primitive di trasformazione top-down:

- definizione degli attributi di un entità/relazione
- reificazione di un attributo di un entità
- decomposizione di una relazione in due relazioni distinte
- trasformazione di un entità in una gerarchia di generalizzazione

+ Vantaggio: il progettista può descrivere inizialmente tutte le specifiche dei dati, trascurando i dettagli per poi entrare nel merito di un concetto alla volta.
- Svantaggio: si può utilizzare solo quando si possiede sin dall'inizio una visione globale e astratta di tutte le componenti del sistema, quindi non è utilizzabile per le applicazioni complesse.

Bottom-Up = Nella metodologia bottom-up, le specifiche iniziali sono suddivise in componenti via via sempre più piccoli fino a quando queste componenti descrivono un frammento elementare della realtà di interesse. Ogni frammento elementare viene rappresentato con uno schema concettuale. Infine i vari schemi ottenuti vengono fusi fino ad ottenere lo schema concettuale finale. A differenza della strategia top-down in cui ogni livello di raffinamento presenta a grandi linee tutti i componenti, nella strategia bottom-up i vari concetti vengono via via introdotti durante le varie fasi. Anche in questo caso per arrivare allo schema finale vengono utilizzate delle primitive di trasformazione bottom-up con quali si introducono nuovi concetti non ancora rappresentati:

introduzione di una nuova entità/relazione
individuazione di un legame tra diverse entità riconducibile ad una generalizzazione
aggregazione di una serie di attributi in una entità/relazione, cioè quando dalle proprietà si individua l'esistenza di un'entità

+ Vantaggio: si adatta a una decomposizione del problema in componenti più semplici e facilmente individuabili e quindi dà la possibilità a diversi progettisti di portare avanti un unico progetto.
- Svantaggio: richiede operazioni di schemi concettuali diversi che nel caso di schemi complessi presentano sempre grosse difficoltà.

Inside-out = La metodologia inside-out è un caso particolare della metodologia bottom-up, inizialmente si individuano i concetti più importante per poi muovere versi quelli meno importante attraverso una navigazione tra le specifiche.

+ Vantaggio: non richiede passi di integrazione perché è necessario esaminare, di volta in volta, tutte le specifiche per individuare concetti non ancora rappresentati descrivendo nel dettaglio quelli nuovi.
- Svantaggio: non è possibile procedere per livelli di astrazione come avviene nella strategia top-down.

Strategia mista = Si combinano i vantaggi di top-down e bottom-up. Il progettista suddivide i requisiti in componenti separate come nella strategia bottom-up, ma allo stesso tempo definisce uno schema scheletro che contiene a livello astratto i livelli principali. Questo schema scheletro serve a favorire le fasi di integrazione degli schemi sviluppati separatamente. Questa è la strategia più flessibile tra quelle viste perché si adatta bene sia alle esigenze di suddividere un problema complesso in sottoproblemi sia in quella di procedere in raffinamenti successivi.

Progettazione logica

L'obiettivo dello schema logico è "tradurre" lo schema concettuale in uno schema logico che rappresenti gli stessi dati in maniera corretta ed efficiente.
In questa fase di progettazione si hanno in
- input: schema concettuale, informazioni sul carico applicativo e sul modello logico da utilizzare
- output: schema logico, documentazione associata e vincoli d'integrità

Prima di procedere con la ristrutturazione dello schema E/R si analizzano le *prestazioni* dello schema concettuale tramite alcuni strumenti che sono utilizzati per prendere decisioni durante la ristrutturazione.
Per analizzare le prestazioni di uno schema E/R si possono effettuare studi di massima di due parametri :
- **tempo**, costo di un'operazione → valutato in termini di numero di occorrenze che mediamente vanno visitate per rispondere ad una operazione
- **spazio**, occupazione di memoria → valutato in termini di spazio di memoria necessario per memorizzare i dati descritti dallo schema

Per studiare questi due parametri bisogna conoscere informazioni come volume dei dati (numero di occorrenze previste e dimensioni di ciascun attributo) e caratteristiche delle operazioni (tipo, frequenza e dati coinvolti). Il volume dei dati e le caratteristiche delle operazioni possono essere descritte tramite due tabelle:
- **tavola dei volumi** = contiene i concetti (entità/relazioni) dello schema con il volume previsto a regime.
- **tavola delle operazioni** = contiene le operazioni, la frequenza prevista e un simbolo che indica se l'operazione è iterativa o batch. Per ogni operazione possiamo descrivere graficamente i dati coinvolti con uno **schema di operazione**, cioè il frammento di schema concettuale interessato dall'operazione.

Gli schemi di operazione vengono riassunti nella tavola degli accessi in cui viene riportato il concetto, il costrutto, il numero di accessi previsti, per quella operazione, e il tipo (lettura/scrittura).

La progettazione logica si articola in due fasi:
- Ristrutturazione dello schema E/R
- Traduzione verso il modello logico

Ristrutturazione dello schema E/R

fase indipendente dal modello logico scelto, si basa su criteri di ottimizzazione dello schema e semplificazione della fase successiva. Questa fase prevede 4 passi:
1. Analisi delle ridondanze
2. Eliminazione delle generalizzazioni
3. Partizionamento/accorpamento di entità e associazioni
4. Scelta degli identificatori primar

1. **Analisi delle ridondanze** → una ridondanza in uno schema concettuale consiste nella presenza di un dato che può essere derivato da altri dati, tramite operazioni. Ci sono varie forme di ridondanza:
- Attributi derivabili, occorrenza per occorrenza, da altri attributi delle stesse entità/associaz
  - Attributi derivabili da attributi di altre entità/associaz, tramite funzioni aggregative
  - Attributi derivabili da operazioni di conteggio delle occorrenze
- Associaz derivabili dalla composizione di altre associazioni in presenza di cicli

La presenza di un dato derivato ha il vantaggio di semplificare le interrogazioni, ma prevede maggiore occupazione di memoria e necessita di effettuare operazioni aggiuntive per mantenerlo aggiornato. La decisione di mantenere o eliminare una ridondanza va presa confrontando il costo di esecuzione delle operazione che coinvolgono il dato ridondante e la relativa occupazione di

memoria nei casi di presenza o assenza della ridondanza.

2. **Eliminazione delle generalizzazioni** → dal momento che non è consentito rappresentare direttamente una generalizzazione, è necessario trasformare questo costrutto in altri del modello E/R (entità e associazioni). Per fare ciò abbiamo tre alternative possibili:

1. Accorpamento delle figlie nel padre → le entità figlie vengono eliminate e le loro proprietà vengono aggiunte all'entità padre. La cardinalità dell'attributo tipo (che sarà aggiunto nell'entità padre per identificare le figlie) sarà:
   - (0,1) → (p,e) parziale, esclusiva
   - (1,1) → (t,e) totale, esclusiva
   - (0,N) → (p,c) parziale, condivisa
   - (1,N) → (t,c) totale, condivisa
2. Accorpamento del genitore nelle figlie → l'entità genitore viene eliminata e, per ereditarietà, le sue proprietà vengono aggiunte alle entità figlie
3. Sostituzione della generalizzazione con associazioni → la generalizzazione si trasforma in due associazioni (1,1) che legano l'entità genitore con entrambe le entità figlie (non ci sono trasferimenti di attributi)

La scelta di queste alternative va fatto considerando occupazione di memoria e costo delle operazioni coinvolte.

L'alternativa 1. è conveniente quando le operazioni non fanno molta distinzione tra le occorrenze e gli attributi. Si ha uno spreco di memoria per la presenza di valori nulli, ma un minor numero di accessi e quindi un costo minore delle operazioni.

L'alternativa 2. è possibile solo se la generalizzazione è totale altrimenti le occorrenze che non stanno in nessuna delle entità figlie non sarebbero rappresentate. In questo caso si risparmia memoria (rispetto all'alternativa 1.) e si riduce il numero di accessi (rispetto all'alternativa 3.).

L'alternativa 3. è conveniente quando la generalizzazione è parziale e ci sono operazioni che si riferiscono solo a occorrenze delle entità figlie o dell'entità padre. In questo caso si ha un risparmio di memoria (rispetto all'alternativa 1.) per l' assenza di valori nulli , ma un incremento degli accessi per mantenere la consistenza delle occorrenze rispetto ai vincoli introdotti.
Le alternative viste non sono le uniche ammesse, ma è possibile effettuare ristrutturazioni che sono combinazioni delle tre presentate.

3. **Partizionamento/accorpamento dei concetti** → Il partizionamento/accorpamento di entità o associazioni serve a garantire maggiore efficienza delle operazioni. Gli accessi si riducono separando attributi di uno stesso concetto che vengono acceduti da operazioni diverse e raggruppando attributi di concetti diversi che vengono acceduti dalle medesime operazioni.

- *Partizionamento delle entità* : questa ristrutturazione è conveniente se le operazioni che coinvolgono frequentemente l'entità originaria accedono a due categorie distinguibili di attributi. Un partizionamento di questo tipo è un esempio di decomposizione verticale di un entità perché si suddivide il concetto operando sui suoi attributi. Nella decomposizione orizzontale la suddivisione avviene sulle occorrenze delle entità.
  Il partizionamento orizzontale ha l'effetto collaterale di dover duplicare tutte le associazioni a cui l'entità originaria partecipa, mentre il partizionamento verticale genera entità con pochi attributi che possono essere tradotte in strutture logiche su cui tramite un solo accesso si recuperano molti dati.
- *Eliminazione di attributi multivalore* : gli attributi multivalore non possono essere rappresentati e quindi devono essere ristrutturati. Quando si presenta questa situazione l'attributo multivalore diventa un'entità legata all'entità originaria con opportuna associazione.
- *Accorpamento di entità* : è l'operazione inversa del partizionamento che si presenta nel caso in cui ci siano operazioni frequenti che accedono alle occorrenze di due entità che vengono accorpate. In questo modo si risparmiano gli accessi, ma è possibile che si presentino valori nulli. Gli accorpamenti si effettuano, in genere, su associazioni (1,1), raramente su associazioni (1,N) e mai su associazioni (N,N).

4. **Scelta degli identificatori principali** → operazione indispensabile per la traduzione nel modello relazionale. Nel caso in cui esistano entità per le quali sono stati specificati più identificatori bisogna decidere quali di questo sarà la chiave primaria. I criteri di decisione sono:

- Gli attributi con valori nulli non possono essere chiave primaria
- Un identificatore con meno attributi è da preferire a uno che ne ha di più (semplicità)
- Un identificatore interno è da preferire ad uno esterno
- Bisogna scegliere l'identificatore utilizzato da molte operazioni per accedere alle occorrenze di un entità

Nel caso in cui nessuno degli identificatori candidati soddisfa tali requisiti viene introdotto un nuovo attributo (codice) alle entità generati appositamente per questo scopo.

Traduzione verso il modello logico

si riferisce a uno specifico modello logico e può includere un ulteriore ottimizzazione in base al modello scelto. A partire da uno schema E/R ristrutturato si traduce caso per caso:

- entità e associazioni molti a molti (N,N): per ogni entità si ha una relazione con lo stesso nome, gli stessi attributi e per chiave il suo identificatore. Per l'associazione invece si avrà una relazione con lo stesso nome, gli stessi attributi con l'aggiunta delle chiavi primarie delle entità coinvolte. Le associazioni ternarie si traducono in maniera analoga, prestando attenzione ad alcuni casi particolare nei quali l'insieme delle chiavi delle relazioni costituisce una superchiave ridondante della relazione che rappresenta l'associazione nello schema E/R.
- entità e associazioni uno a molti (1,N) : gli attributi dell'associazione vengono inseriti negli attributi dell'entità con cardinalità (1,1)
- entità con identificatore esterno : nel caso in cui sono presenti entità identificate esternamente e che quindi partecipano sempre con cardinalità (1,1), la chiave primaria dell'altra entità che partecipa all'associazione è parte della chiave delle entità a 1.
- entità e associazioni uno a uno (1,1) : in presenza di associazioni (1,1) obbligatorie per entrambe le entità è possibile rappresentare l'associazione in una delle relazioni che rappresentano le entità, in base alla modalità di accesso. Nel caso in cui ci sia un associazione (1,1) con partecipazione opzionale per una sola entità, si inserisce nell'entità con partecipazione obbligatoria la chiave dell'entità opzionale più eventuali attributi dell'associazione.