Analisi e Progettazione Concettuale

Raccolta dei requisiti

Dove si raccolgono i requisiti:

- dagli utenti del sistema
- nelle documentazioni relative al sistema
- da realizzazioni preesistenti (nello stesso contesto o in altri contesti)

Caratteristiche di una buona raccolta dei requisiti

- corretto livello di astrazione
- semplicità
- completamente esplicito (nulla di lasciato implicito)
- standardizzazione (dei termini e delle strutture)
- lessico e glossario

E' sempre e costantemente necessario verificare con l'utente la corretta comprensione da parte di entrambi e la consistenza e coerenza di tutto quanto descritto.

I requisiti non possono essere solo sui DATI, ma anche sulle OPERAZIONI da effettuare su tali dati, avendo quindi un approccio OLISTICO all'analisi.

Principio di indeterminazione di Heisenberg (reciproco)

L'osservatore influenza il sistema osservato (ma anche il sistema osservato influenza l'osservatore).

Problema intrinseco della raccolta ed analisi dei requisiti: raccogliendo le informazioni se ne viene influenzati e quindi non si è più oggettivi, e interagendo con gli utenti non si è più neutrali.

Il rischio è conservare senza motivo caratteristiche dei sistemi e dei metodi esistenti senza metterli in discussione (e quindi compromettendo il nuovo sistema con eredità del vecchio), o essere prevenuti nei confronti di parti o utenti del sistema sulla base di quanto dettoci in precedenza.

[Il principio di indeterminazione di Heisenberg postula l'impossibilità di determinare mediante osservazione contemporaneamente la posizione e la quantità di moto di una particella elementare, in quanto l'una esclude l'altra. In prima approssimazione si dice che la misura della posizione influenza la quantità di moto, ed in generale che l'osservatore modifica il sistema osservato osservandolo.]

Documentazione di progetto

Il modello E-R rappresenta in modo formale i concetti ma non li descrive, e non è adatto per la rappresentazione di vincoli complessi.

E' necessario integrare il modello E-R con una descrizione (formale o no) delle rimanenti parti del sistema.

Business Rule

Le business rule relative all'analisi dei dati sono di tre tipi:

- descrizione di concetti – in forma testuale libera
- dichiarazione di vincoli d'integrità

  – asserzioni del tipo: <ente del modello> **deve / non deve** <proprietà o espressione>
- derivazione di una proprietà da altre esistenti

  – operazioni aritmetiche o logiche per dedurre proprietà o enti da altre informazioni presenti nel modello, del tipo: <risultato> **si ottiene** <espressione o formula>

Vi sono poi altre business rule che meglio si adattano alla modellazione dei processi aziendali, descrivendo appunto le regole aziendali che governano la trasformazione dei dati, ed hanno la forma:

se <condizione iniziale> **allora** <descrizione del risultato di un processo di trasformazione>

Strumenti di progettazione assistita (CASE)

Computer-Aided Software Engineering.

Gli strumenti C.A.S.E. aiutano lo sviluppo del software attraverso interfacce grafiche, automatismi, generatori e librerie di funzionalità.

Obiettivi degli strumenti CASE :

- semplificare la scrittura di codice
- automatizzare i passi ripetitivi (un buon programmatore non ripete tre volte lo stesso processo, ma scrive un programma che lo esegue infinite volte!)
- garantire conformità agli standard
- facilitare il lavoro cooperativo

Tipologie:

1. **Tools** - focalizzati su singoli task del processo di sviluppo del software

2. **Workbenches** - insieme di tool coordinati per supportare un intero segmento dello sviluppo

3. **Environments** - ambiente integrato che copre “tutto” il ciclo di sviluppo

UML

UML : Unified Modeling Language - "linguaggio di modellazione unificato"

Linguaggio di modellazione e specifica basato sul paradigma object-oriented.

Il nucleo del linguaggio fu definito nel 1996 da "i tre amigos" (Grady Booch, Jim Rumbaugh e Ivar Jacobson) all'interno dell'OMG (Object Management Group), raccogliendo le best practices per definire uno standard unico.

UML svolge la funzione di "lingua franca" della progettazione e programmazione a oggetti, descrivendo soluzioni progettuali in modo standard, sintetico e comprensibile a tutti.

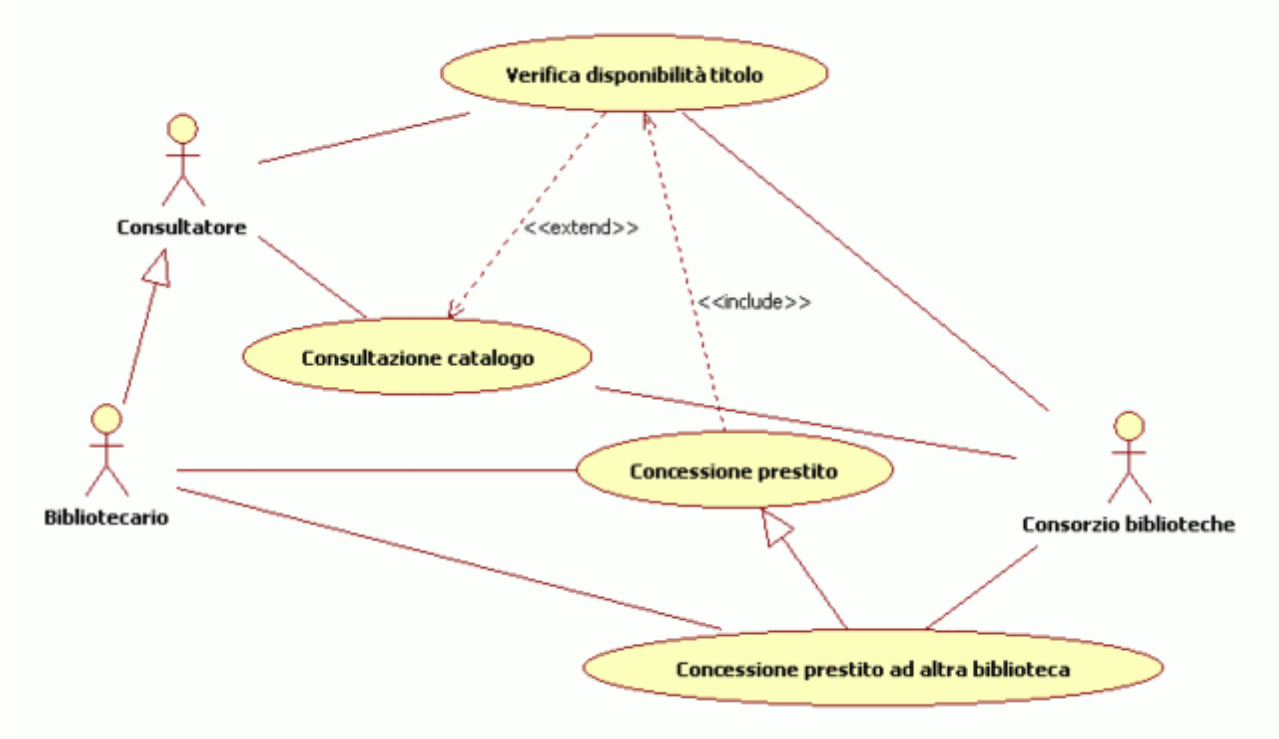
Viene usato nella progettazione del software per descrivere il **comportamento** e la **struttura** di un sistema.

La notazione UML è semi-grafica e semi-formale: un modello UML è costituito da una collezione di diagrammi, costruiti da elementi grafici (con significato formalmente definito), elementi testuali formali, ed elementi di testo libero.

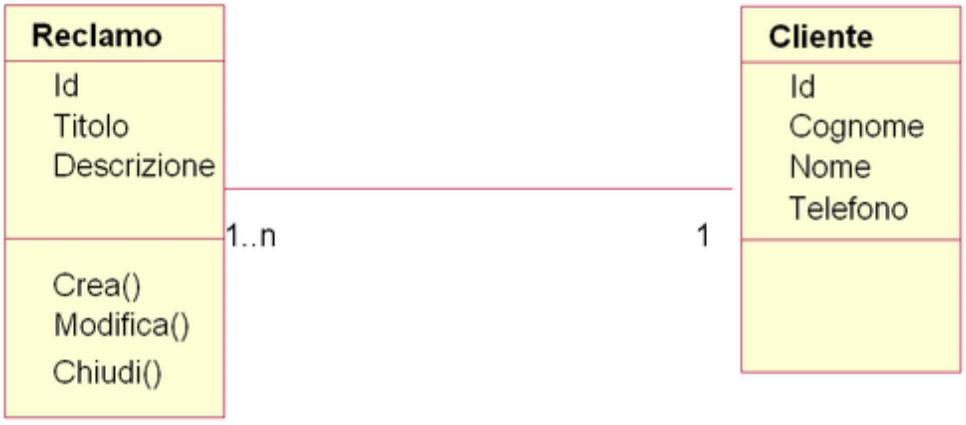
I diagrammi classici UML

- Use Case Diagram
- Class Diagram
- Object Diagram
- Statechart Diagram
- Activity Diagram
- Sequence Diagram
- Communication Diagram / Collaboration Diagram
- Component Diagram
- Deployment Diagram

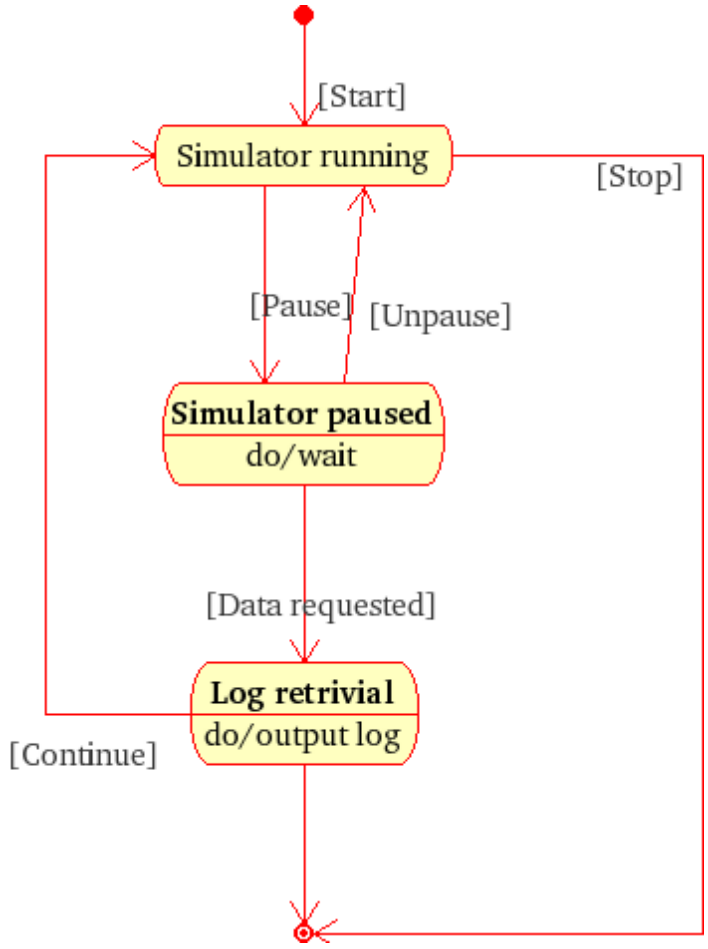
Use Case Diagram



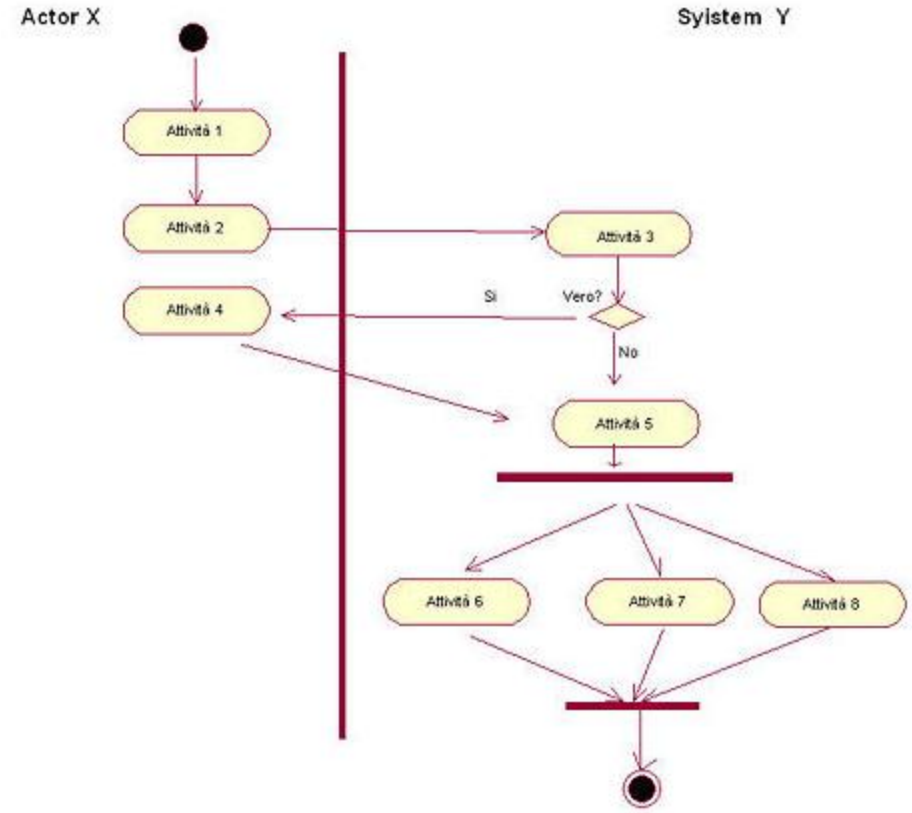
Class Diagram



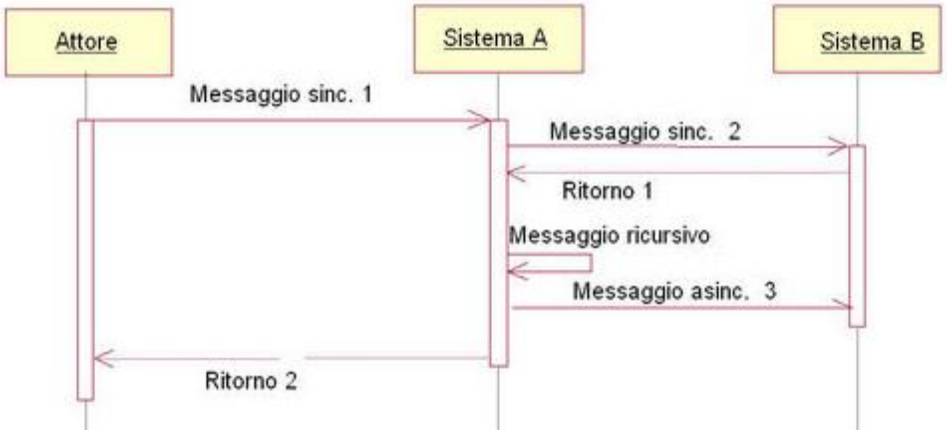
Statechart Diagram



Activity Diagram



Sequence Diagram



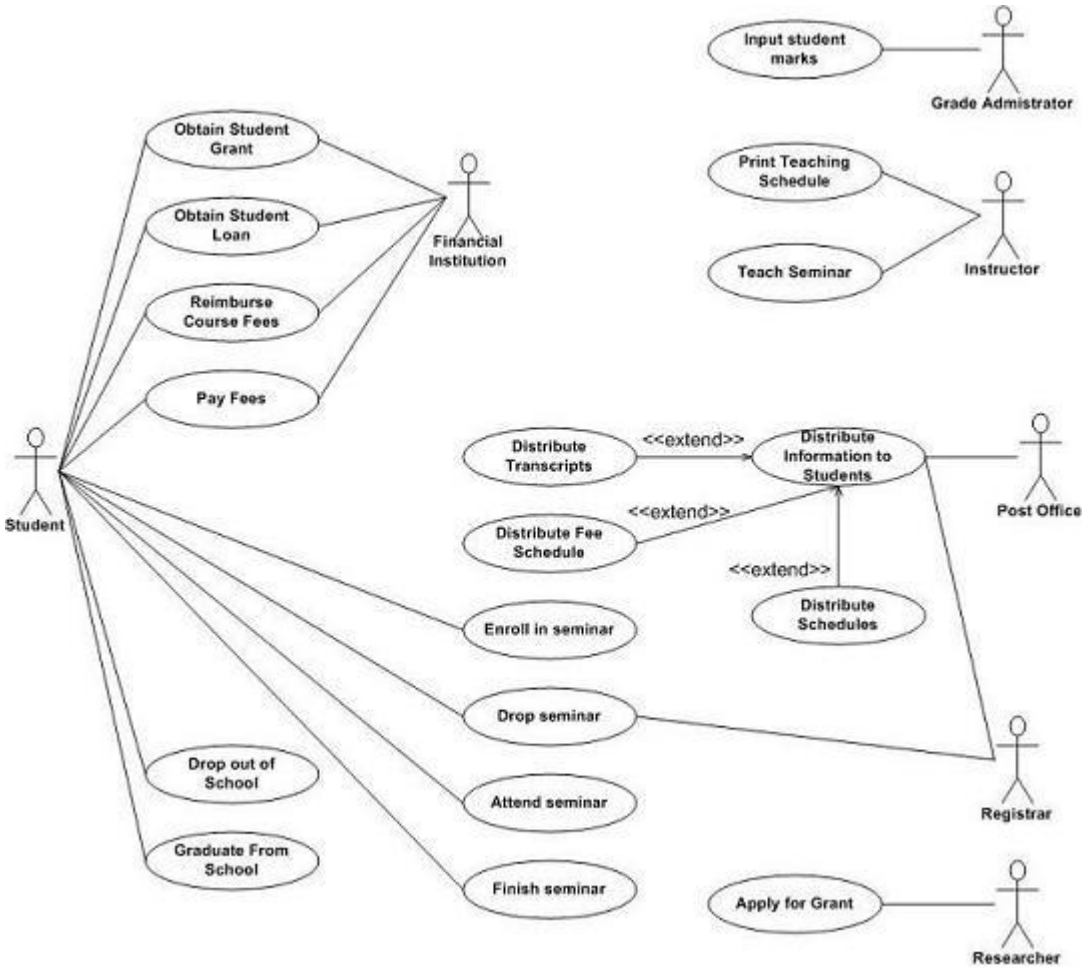
Scenari d'uso (Use Case)

Uno strumento molto importante per l'analisi dei sistemi è la raccolta e la descrizione di scenari d'uso, cioè di precise sequenze di azioni ed eventi (come la sceneggiatura di un film) relativi a processi reali o da realizzare.

Sono utili soprattutto per la descrizione della parte dinamica del sistema, ma poiché i dati dovranno essere utilizzati all'interno di questo sistema dinamico, per ben progettare i dati devono essere noti anche i processi.

Permettono di far vedere (come in un film) all'utente come sarà il sistema, ma anche di formalizzare la descrizione dell'esistente per una più chiara comprensione da parte di progettisti ed analisti.

Use Case Diagram



Esempio di Scheda per UseCase

Use Case UC-1

Use Case ID:	UC-1		
Use Case Name:	Interrogazione Orale		
Created By:		Last Updated By:	
Date Created:		Date Last Updated:	

Actors:	P - Professore S - Studente
Description:	Uno studente si presenta per sostenere un esame orale
Trigger:	
Preconditions:	1. P prepara l'aula per l'esame 2. P apre l'aula
Postconditions:	1. S festeggia
Normal Flow:	3. S entra nell'aula 4. P fa una domanda a S 5. S risponde bene 6. P da un voto > 18 7. S esce dall'aula contento
Alternative Flows:	5.1 S risponde male 6.1 P da un voto < 18 7.1 S torna a casa triste
Exceptions:	1. S dimentica il libretto a casa
Includes:	
Priority:	
Frequency of Use:	Almeno 6 volte l'anno
Business Rules:	
Special Requirements:	
Assumptions:	S iscritto all'università S ha studiato
Notes and Issues:	

Modellazione Concettuale

Schemi e Istanze

schema: sostanzialmente invariante nel tempo, descrive la struttura (aspetto intensionale)
- nel modello relazionale, le intestazioni delle tabelle

istanza: i valori attuali, che cambiano anche molto rapidamente (aspetto estensionale)
- nel modello relazionale, il "corpo" di ciascuna tabella

Modelli

modelli concettuali: permettono di rappresentare i dati in modo indipendente dal sistema
- cercano di descrivere i concetti del mondo reale
- sono utilizzati nelle fasi preliminari di progettazione

modelli logici: utilizzati per l'organizzazione dei dati da parte dei DBMS
- utilizzati dalle applicazioni
- indipendenti dalle strutture fisiche

Motivi per l'uso dei modelli concettuali

Se partiamo a progettare direttamente dallo schema logico della base di dati di un'applicazione:
- da dove cominciamo?
- rischiamo di perderci subito nei dettagli
- dobbiamo pensare subito a come correlare le varie tabelle (chiavi etc.)
- ci sono grosse rigidità

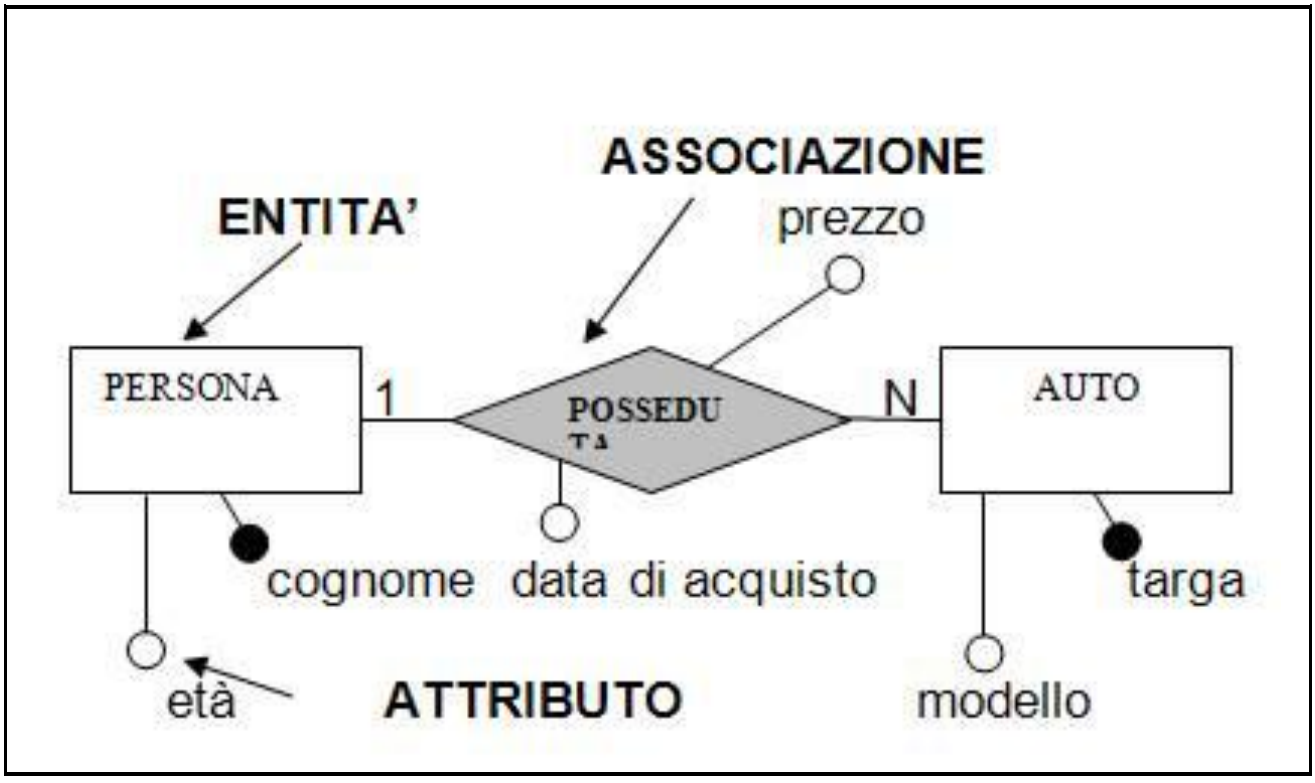
Il modello concettuale ha:
- una rappresentazione grafica (più comunicativa)
- maggiore astrazione rispetto ai dettagli implementativi
- quindi più semplice
- quindi più comprensibile per l'utente/committente del sistema

La tecnica di modellazione concettuale più diffusa è quella "**Entity-Relationship**" (E-R)

Modello Entity-Relationship

Elementi e costrutti del modello E-R:

- **Entità** - rappresentazione di una categoria di oggetti del mondo reale che viene modellato
- **Attributo** – caratteristica dell'entità d'interesse nel modello (aggregabili in attributi composti)
- **Relazioni o associazioni** (Relationship) – collegamento tra diverse entità del modello
- **Cardinalità** – delle relazioni ma anche degli attributi
- **Identificatore** – insieme di attributi e/o relazioni che identificano un'entità
- **Generalizzazione** – classificazione gerarchica delle entità con ereditarietà delle caratteristiche (attributi, relazioni, etc..)



Entità

Di solito si tratta di classi di oggetti (non singoli oggetti ma insiemi di oggetti con caratteristiche comuni) che possiedono un significato ed un'esistenza autonoma

Tutto dipende dal contesto applicativo, quella che può essere un'entità per un progetto potrebbe non esserlo in un altro (persone come entità in un sistema anagrafico, come valore numerico in un sistema geografico).

Ogni singola entità è indicata con un nome, espressivo all'intero del contesto applicativo.

Si seguono di solito delle opportune convenzioni, come quella di utilizzare nomi al singolare.

Attributi

Sono in prima approssimazione i campi di una tabella.

Possono anche essere **COMPOSTI.**

Relationship

Legame logico fra due o più entità, **rilevante** nell'applicazione di interesse

Il legame padre-figlio è fondamentale in un sistema per l'anagrafe civile, ininfluente in un sistema di gestione clienti.

Anche le relazioni hanno un nome, che sia possibilmente significativo e possibilmente non un verbo ("succedere", "contiene", ...) ma un sostantivo ("successione","componente", ...).

E' possibile definire per le varie entità che sono collegate da una relazione dei ruoli, che appunto dettagliano qualora sia necessario che parte gioca la singola entità nella relazione.

Tra due entità possono esservi più relazioni, ognuna con un suo particolare significato (ad esempio tra le entità "persona" e "città" possono esserci le relazioni "Residenza", "Nascita", "Domicilio",..).

Le relazioni possono essere tra più di due entità, ed in alcuni casi possono essere "opzionali" tra due o più entità.

Le relazioni possono essere anche ricorsive sulla stessa entità.

In questo caso di solito è necessario definire i ruoli all'interno della relazione che altrimenti potrebbe essere interpretata nel verso sbagliato (ad esempio per la relazione "genitore" sull'entità "persona" deve essere indicato quale è la "madre" e quale il "figlio").

Promozione di una relazione

Un attento esame delle relazioni a volte porta a "promuovere" una relazione e trasformarla in un'entità.

Ad esempio un esame visto in un primo momento come una relazione tra studente e corso può poi diventare per esigenze applicative un'entità autonoma con una relazione con studente ed una con corso.

Di solito questo avviene quando ad una relazione iniziano ad essere associati molti attributi.

Cardinalità

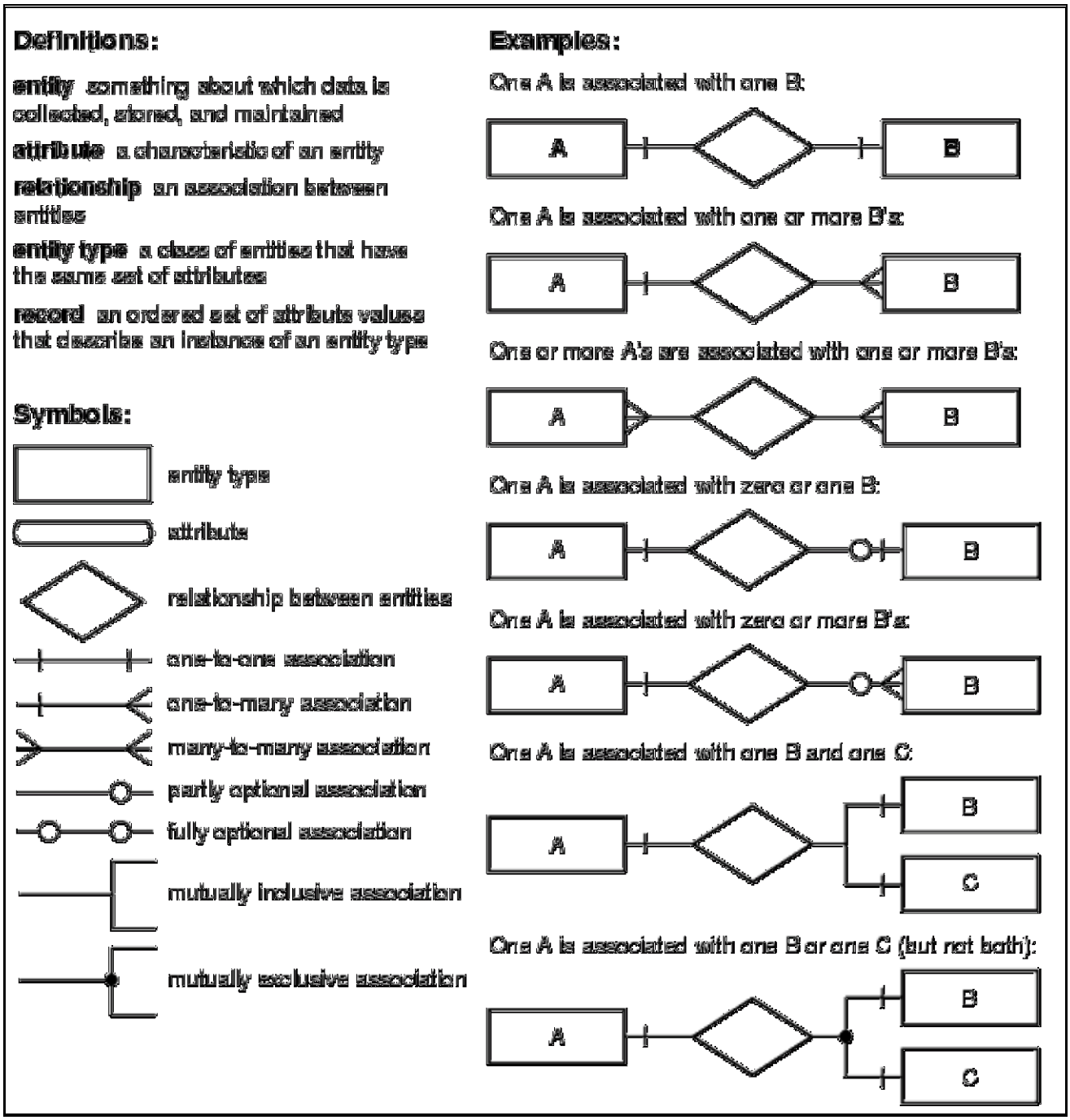
Indica genericamente la numerosità di un ente.

Di due tipi:

- cardinalità di una relazione
- cardinalità di un attributo

La **cardinalità di una relazione** è una coppia di numeri interi che indica, per ogni "ramo" di una relazione (cioè per ogni entità che appare nella relazione), il numero minimo e massimo di volte che una relazione può esistere per una particolare occorrenza (oggetto) di un'entità.

Ad esempio la relazione ricorsiva "genitore" avrà una cardinalità 0-n per il ramo "madre" (cioè una persona può essere "madre" da 0 a n volte), mentre avrà una cardinalità 1-1 per il ramo "figlio" (cioè una persona deve sempre essere "figli" una ed una sola volta).



Identificatore

Insieme di attributi e/o di relazioni che identifica un'entità.

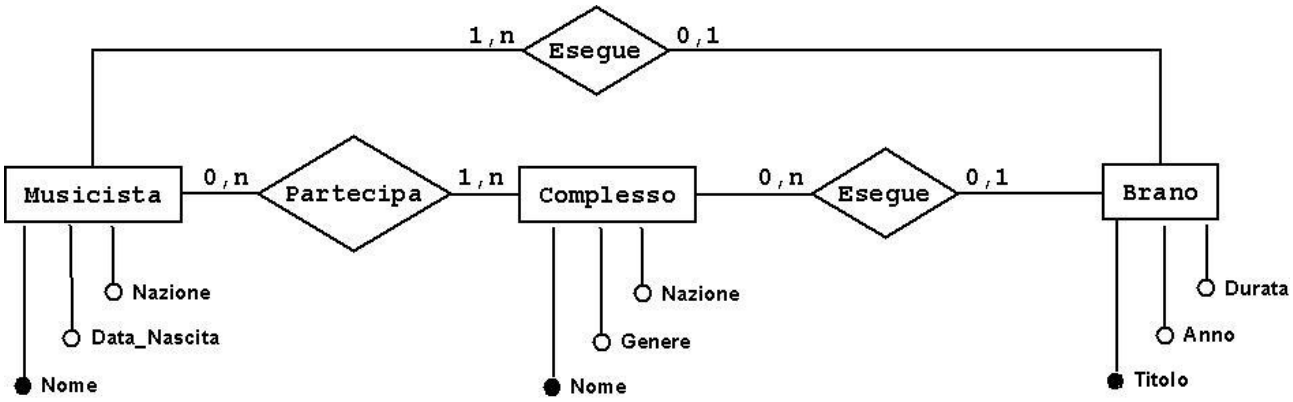
Gli identificatori sono i candidati a diventare le chiavi.

Generalizzazione

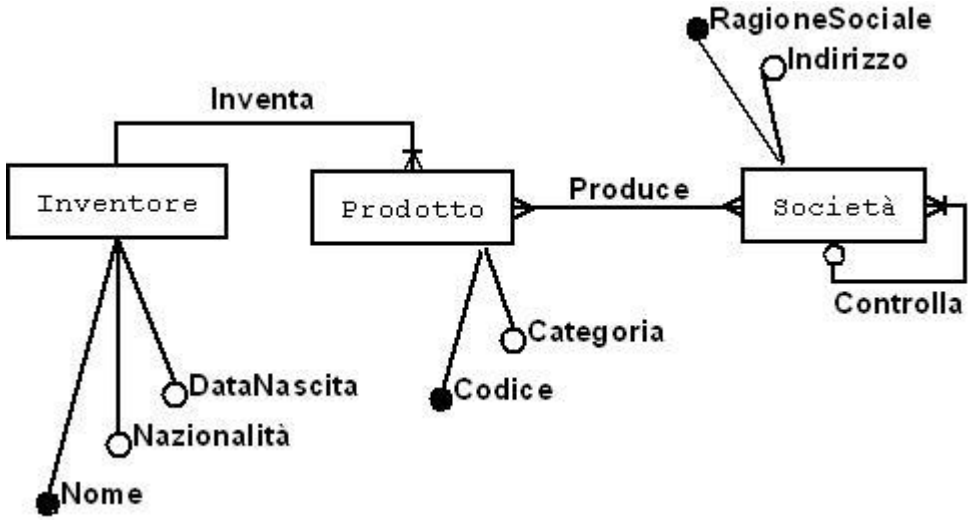
Una serie di entità potrebbero avere delle caratteristiche in comune, come ad esempio un impiegato, uno studente, un professore: hanno tutte un nome, un cognome, una data di nascita, etc.

In questo caso si può pensare ad un'entità ”Persona” che è la generalizzazione di tutte queste entità e definire che le varie impiegato, studente e professore derivano da questa entità generale, ereditando da essa tutte le caratteristiche e definendone eventualmente di nuove.

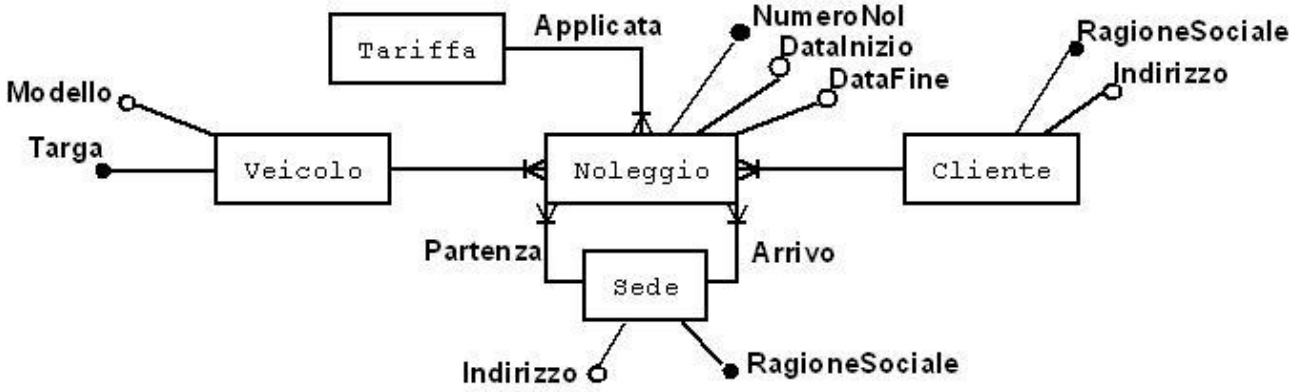
Esempi di diagrammi ER



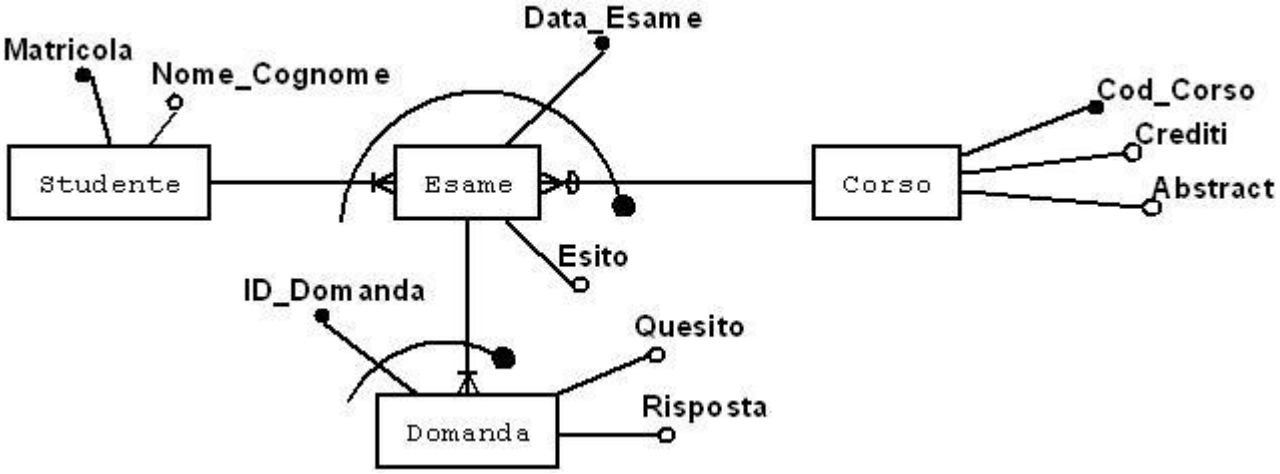
Struttura semplice



Relazione ricorsiva



Più relazioni tra due entità



Chiavi composte

METODOLOGIA DI DISEGNO

Concetti di base del processo di disegno di un DB

Passi essenziali di disegno

Il disegno di un DB può essere effettuato seguendo una serie di passi elementari, cioè “eseguendo” una specie di algoritmo.

Come già più volte detto non esiste un algoritmo unico e perfetto per il disegno di un DB a partire da delle specifiche testuali, sia perché esistono diversi metodi e strategia che possono essere vantaggiosi o problematici a seconda del problema specifico che si deve affrontare, ma anche per la diversa forma che può assumere la descrizione testuale di uno stesso problema, con maggiore o minore rigore formale.

Di seguito si fornisce quindi una traccia più che un algoritmo, che dev'essere seguita sempre con un forte spirito critico in modo da non farsi fuorviare del metodo rispetto al contesto specifico del problema.

1. Leggere il testo
2. Individuare le entità
3. Verificare se non si sono entità duplicate
   - Eventuale cambio dei nomi per semplicità
4. Individuare le associazioni tra le entità
5. Verificare se alcune associazioni hanno attributi
6. Calcolare la cardinalità delle associazioni
7. Trovare gli attributi espliciti delle entità
8. Definire le chiavi primarie delle entità
9. Identificare eventuali Gerarchie
10. Stesura dello schema E-R

Leggere il testo

Senza un'attenta lettura ed una comprensione complessiva del testo non si può sperare di creare un disegno corretto e soprattutto corrispondente al contenuto del testo.

Spesso una sola lettura, una singola passata, non è sufficiente per una piena comprensione, ed in alcuni casi può essere necessario rivolgersi a chi ha scritto il testo per comprendere meglio alcuni dettagli che risultano non chiari. Questo può essere dovuto a diversi fattori, che vanno dalla non condivisa base di conoscenza sul problema affrontato dal testo alla difficoltà intrinseca di tradurre in poche e semplici parole concetti che richiederebbero approfondite analisi. Un caso tipico è quello di concetti che lo scrittore da per impliciti in quanto facenti parte del dominio delle conoscenze standard di un certo contesto operativo, ma che non sono poi di comune dominio e quindi immediatamente accessibili al lettore.

Individuare le entità

Verificare se non si sono entità duplicate

Eventuale cambio dei nomi per semplicità

Individuare le associazioni tra le entità

Verificare se alcune associazioni hanno attributi

Calcolare la cardinalità delle associazioni

Trovare gli attributi espliciti delle entità

Definire le chiavi primarie delle entità

Identificazione di eventuali gerarchie

Stesura dello schema E-R

Ristrutturazione degli schemi E-R

Serve per:

- facilitare la successiva traduzione dello schema E-R in uno schema relazionale.
- ottimizzazione delle prestazioni

Passi di ristrutturazione

- analisi delle ridondanze
- trasformazione delle generalizzazione
- eliminazione attributi multi-valore
- accorpamento entità (associazioni 1-1)
- partizionamento verticale di entità (parte dei dati di un'entità non vengono quasi mai usati)
- partizionamento orizzontale di associazioni multiple
- identificatori principali

La ristrutturazione deve essere guidata di due obiettivi:

1. pulizia del disegno
2. ottimizzazione delle prestazioni

Questi due obiettivi spesso vanno di pari passo, ma a volte divergono, ed è quindi necessario trovare il giusto equilibrio tra “perfezione formale” del disegno e “sporca ottimizzazione” delle operazioni.

Analisi delle ridondanze

Le ridondanze creano problemi si allineamento dei dati, ma in alcuni casi possono velocizzare le interrogazioni.

Vantaggi:

- interrogazioni più semplici

Svantaggi:

- complicazione ed appesantimento degli aggiornamenti e **rischio di inconsistenza**
- maggiore occupazione di spazio

Le ridondanze possono essere dovute:

- attributi ripetuti tra entità
- attributi i cui valori possono essere altresì derivati da altri attributi della stessa entità

- attributi i cui valori possono essere derivati da attributi di altre entità
- attributi i cui valori possono essere derivati da conteggi su entità
- associazioni derivabili da altre associazioni

Trasformazione delle generalizzazioni

Le generalizzazioni possono essere trasformate secondo tre linee principali, in base alle tipologie di partizionamento delle informazioni tra entità generale ed entità di dettaglio, sia dal punto di vista della completezza che dell'uso che ne viene fatto.

Le tre linee guida sono:

1. un'unica tabella che accorpa entità generale e tutte le tipologie di dettaglio (minimo comune multiplo, 1 sola tabella) – necessario metodo per identificare il tipo
2. replica della struttura del generale nelle varie entità di dettaglio (n tabelle)
3. entità separate per parte generale e per i vari dettagli (1 + n tabelle)

Ogni linea presenta vantaggi e svantaggi che vanno pesate sulla base dell'uso che si fa dei dati.

1.conviene se gli accessi al generale ed al dettaglio sono contestuali

2.conviene se gli accessi ai dettagli sono distinti per tipologia

3.conviene se gli accessi ai dettagli sono separati dagli accessi alle informazioni generali

Eliminazione attributi multi-valore

Gli attributi multi-valore possono essere mappati con una tabella di dettaglio con una associazione uno a molti con quella di origine.

Intrinsecamente i campi multi-valore hanno dei limiti dal punto di vista della numerosità massima gestibile e dell'accesso al singolo valore, problemi che possono essere superati trasformando appunto in una tabella di dettaglio che mantiene una dipendenza con l'entità principale.

Accorpamento entità (associazioni 1-1) - Partizionamento verticale di entità

Ogni volta che si presenta una associazione 1-1 tra due entità si deve valutare se tale associazione non nasconda in realtà un'identità tra le due entità coinvolte, cioè che le due entità modellate rappresentino aspetti diversi della stessa entità reale.

Dev'essere quindi valutata attentamente l'opportunità di accorpare in un'unica entità gli attributi delle due entità di partenza, tenendo presente che a volte considerazioni di carattere prestazionale possono spingere a non effettuare tale operazione.

Nel senso opposto le varie entità devono essere analizzate in funzione delle interrogazioni svolte su di esse in modo da ottimizzarne l'accesso.

Infatti se una parte dei dati non viene quasi mai acceduta può essere "parcheggiata" in una entità associata che pur mantenendo le informazioni non appesantisce le interrogazioni "abitudinali".

Oppure se alcune interrogazioni si focalizzano su una parte di dati ed altre su una parte diversa, queste parti possono essere focalizzate "specializzando" due sotto entità in funzione delle interrogazioni.

Partizionamento orizzontale di associazioni multiple

Una associazione 1-n o n-m può avere caratteristiche di accesso particolari, cioè il legame può essere prioritario su una parte di questa associazione, soprattutto quando tale associazione descrive la storia di un'entità o di una associazione che varia nel tempo. Ad esempio una associazione tra un'automobile e tutti i suoi proprietari avrà una focalizzazione sul proprietario attuale, essendo gli altri temporalmente superati.

In tal caso si può prendere in considerazione l'opportunità di dividere la relazione originaria in due relazioni che rappresentano una la caratteristica prioritaria (ad esempio la situazione corrente) e l'altra tutte le caratteristiche secondarie (ad esempio lo storico di tutte le relazioni).

Identificatori principali

Un passo fondamentale è l'analisi delle entità per l'individuazione degli identificatori principali (o chiavi) delle entità stesse.

I criteri di base per l'individuazione degli attributi adatti sono:

- assenza di opzionalità (cioè not null)
- minimalità (dannoso aggiungere attributi superflui)
- valutazione dell'utilizzo relativamente alle varie operazioni

Se nessun insieme di attributi soddisfa queste caratteristiche si può ricorrere alla creazione di nuovi attributi ad hoc: i **codici identificativi**.

Tali codici assumono valori generati appositamente, di solito attraverso **SEQUENCE**