Conceptual Design

L'analisi dei requisiti non è standardizzabile ma ci sono alcune regole pratiche nella fase di sviluppo della base di dati.

- *raccolta dei requisiti*: completa individuazione dei problemi che l'applicazione deve risolvere e le caratteristiche che dovrà avere;
- *analisi dei requisiti*: organizzazione delle specifiche dei requisiti.
  Principali fonti di informazione:
- gli *utenti dell'applicazione*, mediante le interviste oppure attraverso una documentazione scritta;
- *documentazione esistente*, moduli, regolamenti interni, procedure aziendali e normative. E' richiesta un'attività di raccolta e selezione che viene assistita dagli utenti a carico del progettista;
- *realizzazioni preesistenti*, applicazioni che si devono rimpiazzare o che devono interagire con il sistema da realizzare.

Alcune regole generali:

- *scegliere il corretto livello di astrazione*: evitare termini troppo generici o troppo specifici;
- *standardizzare la struttura delle frasi*: è preferibile utilizzare sempre lo stesso stile sintattico;
- *evitare frasi contorte*: le definizioni devono essere semplici e chiare;
- *individuare sinonimi/omonimi e unificare i termini*: l'uso di sinonimi e omonimi può generare ambiguità e perciò si tende ad utilizzare un termine unico per i sinonimi, mentre nel caso di omonimi si utilizzano termini diversi;
- *rendere esplicito il riferimento tra termini*: l'assenza di un contesto di riferimento può rendere alcuni concetti ambigui, allora occorre esplicitare il riferimento tra termini;
- *costruire un glossario dei termini*: utile per la comprensione e la precisazione dei termini usati, deve contenere per ogni termina una breve descrizione, possibili sinonimi e altri termini nel glossario con i quali esiste un legami logico.

Design Pattern

Ci sono alcune buone pratiche per una corretta rappresentazione concettuale dei dati, come dei *criteri generali di rappresentazione* o *design pattern*, ovvero soluzioni progettuali a problemi comuni.

Criteri generali di rappresentazione

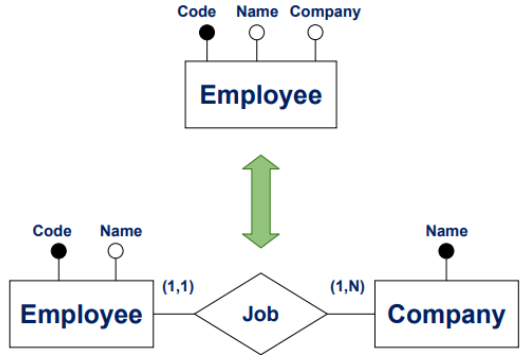
Premessa, non esiste una rappresentazione univoca di un insieme di specifiche, ma è utile avere delle indicazioni sulle scelte più opportune:

- se un concetto ha proprietà significative e/o descrive classi di oggetti con esistenza autonoma, è opportuno rappresentarlo con un'entità;
- se un concetto ha una struttura semplice e non possiede proprietà rilevanti associate è opportuno rappresentarlo con un attributo di un altro concetto a cui si riferisce;
- 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;

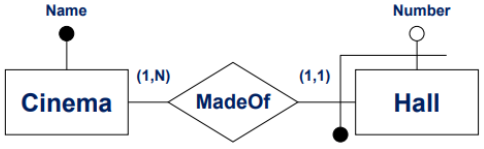
- se uno o più concetti risultano essere casi particolari di un altro, è opportuno rappresentarli facendo uso di una generalizzazione.
  Questi criteri hanno validità generale, cioè sono indipendenti dalla strategia di progettazione scelta.

Pattern di Progetto

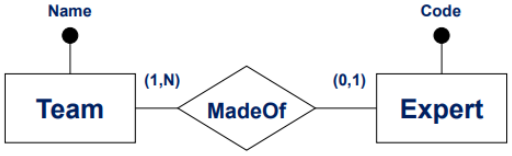
1. **Reificazione degli attributi**:
   Viene rappresentato un *impiegato* con codice, nome e azienda, ma non stiamo rappresentando anche il concetto di *azienda*; per poterlo rappresentare esplicitamente reifichiamo l'attributo facendolo diventare un'entità.



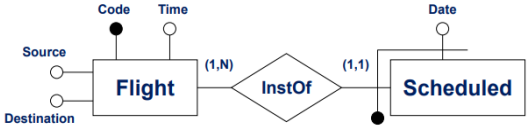
2. **Part-Of**
   Si vuole rappresentare il fatto che un'entità è parte di un'altra entità, tipicamente queste relazioni sono uno a molti e si rappresentano in due forme:
   - l'esistenza di un'occorrenza dell'entità "parte" dipende dall'esistenza di un'occorrenza dell'entità che la contiene:



   - l'entità contenuta nell'altra ha esistenza autonoma:

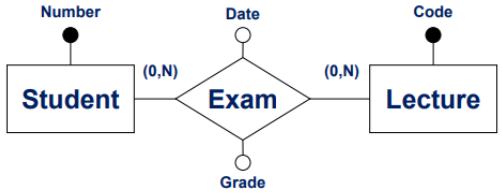


3. **Instance-Of**:
   Talvolta sono necessarie due entità distinte, una che coinvolgono la rappresentazione astratta e un'altra che memorizza le informazioni necessarie per i nostri requisiti.
   - un'entità che descrive il concetto astratto di volo e un'altra entità che rappresenta il volo reale:

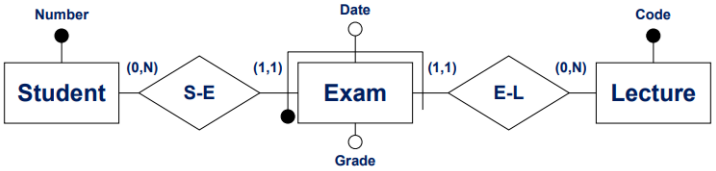


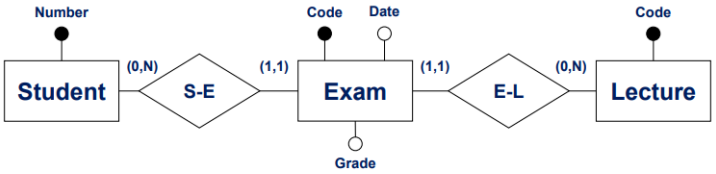
   - viene rappresentato il concetto di torneo sportivo e una sua edizione:

4. **Reification: Binary Relations**:
   si utilizza una relazione, tipicamente molti a molti, per descrivere un concetto che lega altri due concetti, ad esempio: rappresentiamo un'esame come relazione tra *studente* e *corso*:

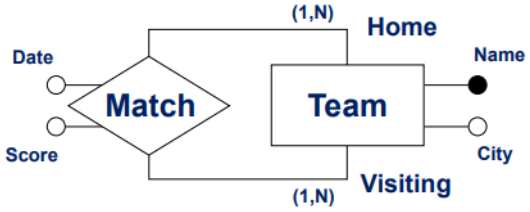


Questa soluzione è valida solo se ogni studente può sostenere una sola volta un certo esame (per def: una occorrenza della relazione *esame* è un insieme di coppie *studente-corso*, senza duplicati).
La soluzione corretta si ottiene reificando la relazione *esame* rappresentandola come entità:

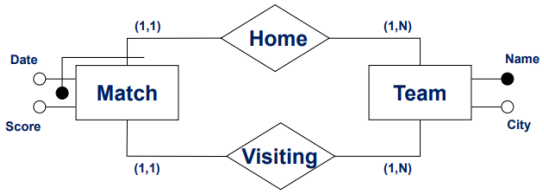




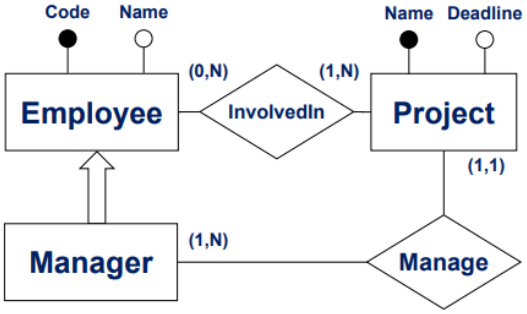
5. **Reification: Recursive Relation**:
   Esempio: il concetto di *partita* può essere visto come una relazione ricorsiva sull'entità squadre,



   ma se in un torneo due squadre si incontrano più volte è necessario reificare la relazione binaria:

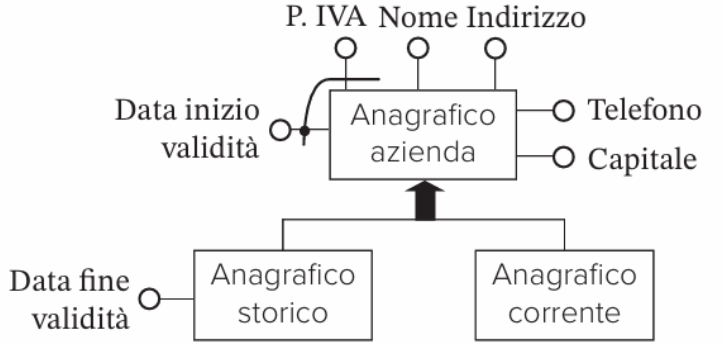


   L'identificazione dell'entità *partita* coinvolge solo la data e la squadra che gioca in casa perché si assume che una squadra non possa giocare due partite nello stesso giorno.
6. **Generalizzazioni**:
   - *specific case*:
     Le generalizzazioni sono usate per definire casi specifici, nell'esempio *manager*, dell'entità data, *employee*; in questo caso non tutte gli impiegati gestiscono un progetto:

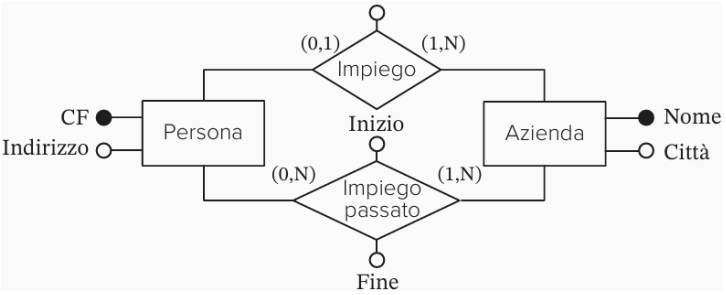


   - *storicizzazione di un'entità*:
     esempio, vogliamo memorizzare le informazioni correnti di un'azienda, tenendo traccia dei dati che sono variati.
     soluzione: utilizzare due entità con gli stessi attributi, una rappresenta il concetto di interesse con le informazioni aggiornate e l'altra lo storico. Le proprietà di queste entità vengono messe

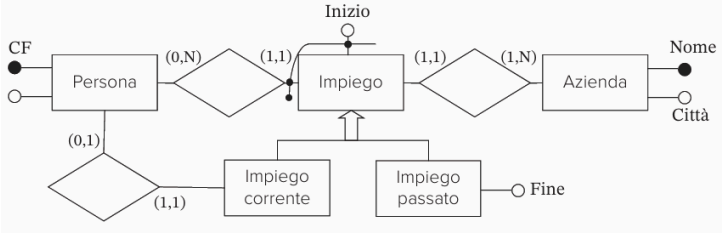
a fattor comune mediante una generalizzazione la cui entità genitore rappresenta tutte le informazioni anagrafiche delle aziende, sia quelle correnti sia quelle passate; in più vengono introdotti degli attributi per definire l'intervallo di validità dei dati (data inizio e data fine). L'identificazione si ottiene aggiungendo alla chiave originaria (p. iva) la data di inizio, istante che diventerà anche la data di fine validità delle informazioni che vengono soppiantate.



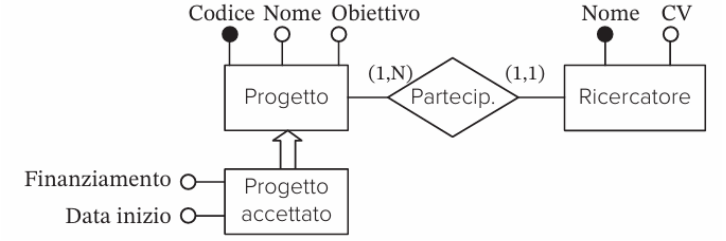
- *storicizzazione di una relazione*:
  esempio, vogliamo rappresentare gli impieghi presenti e passati di una persona.
  soluzione uno: rappresentare separatamente i dati correnti e i dati storici e introdurre opportuni attributi per specificare gli intervalli di validità.



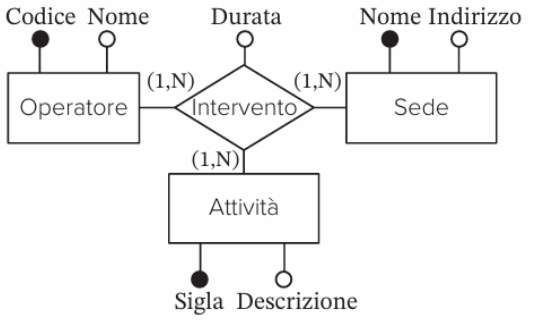
  Ma, osservando le cardinalità delle partecipazioni dell'entità *persona* alle due relazioni, vediamo che non possiamo rappresentare il fatto che una persona possa aver lavorato, in periodi diversi, per la stessa azienda.
  soluzione due: reificazione delle relazioni, inoltre è necessario l'inserimento di un vincolo esterno allo schema che impone che tutte le occorrenze della relazione tra *persona* e *impiego corrente* compaiano anche tra le occorrenze della relazione tra *persona* e *impiego*.



- *estendere un concetto*:
  vogliamo rappresentare il fatto che un certo concetto subisce un'evoluzione nel tempo che può essere diversa per le diverse occorrenze del concetto.
  esempio: si vuole rappresentare progetti che vengono proposti con l'obiettivo di ottenere un finanziamento. Solo alcuni vengono accettati e per questi vengono aggiunte ulteriori informazioni: data di inizio ufficiale e finanziamento.

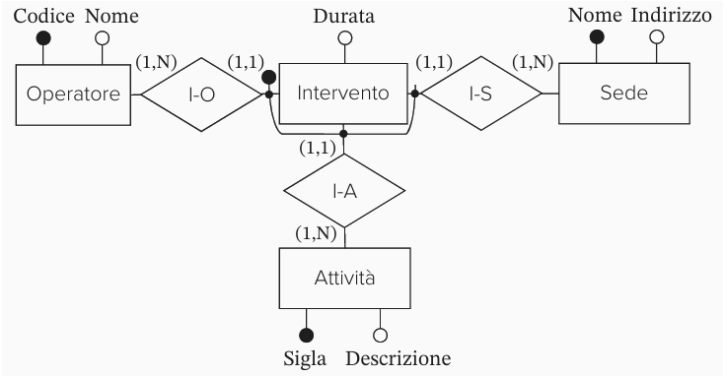


- **Relazioni ternarie**:
  esempio 1: si vuole modellare il caso in cui un operatore può effettuare operazioni che consistono in attività diverse svolte in sedi diverse, in ogni sede possono operare operatori diversi svolgendo attività diverse e le attività possono essere svolte da operatori diversi e in sedi diverse.

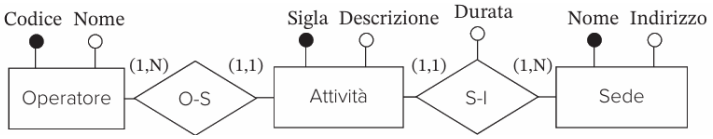


  Questo schema può essere reificato per modellare altre realtà:

  - Ogni lavoro è definito da un *Operatore* (I-O) che lavora in una certa *Sede* (I-S) per una certa *Attività* (I-A).



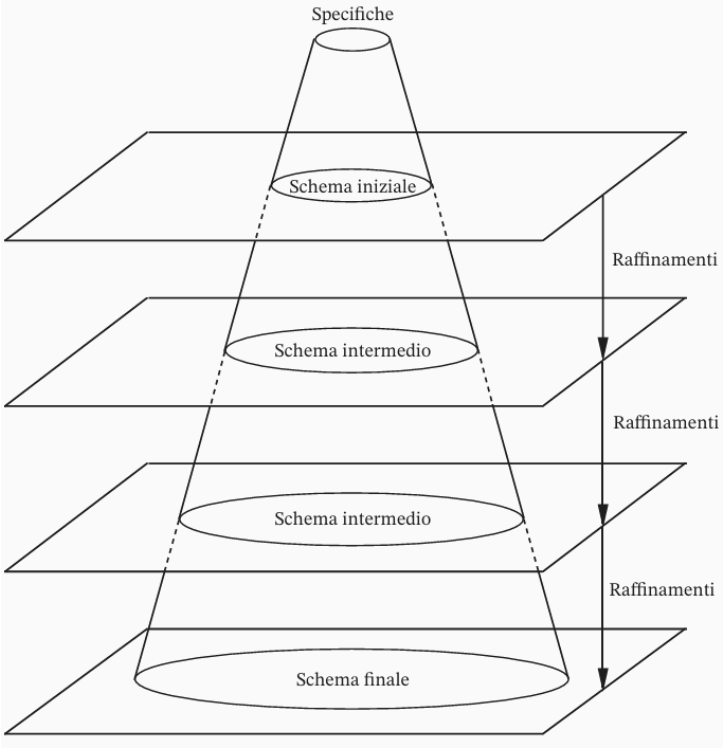
  - Se un'attività può essere eseguita da un solo operatore in un solo ufficio, allora la reificazione può essere semplificata.



Strategie di Progetto

Strategia top-down

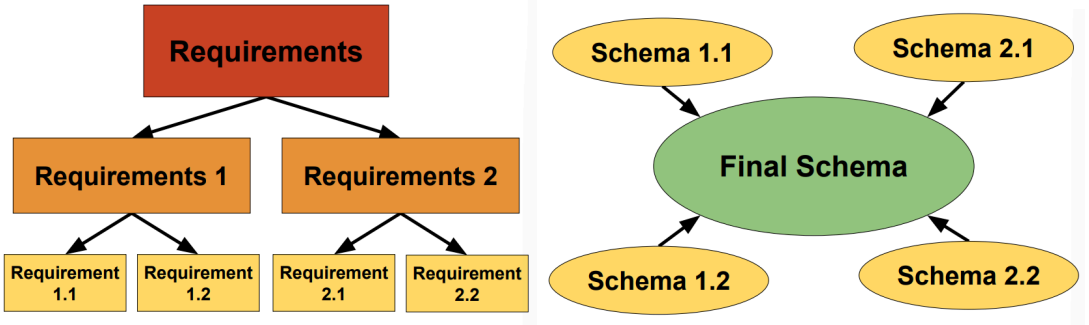
Si parte da uno schema iniziale che descrive tutte le specifiche con pochi concetti molto astratti, questo schema viene poi via via raffinato mediante trasformazioni che aumentano il dettaglio dei vari concetti.



Ognuno di questi piani descrive le stesse informazioni a un diverso livello di dettaglio. Con questa strategia tutti gli aspetti presenti nello schema finale sono presenti, in linea di principio a ogni livello di raffinamento. Il vantaggio sta nel poter descrivere inizialmente tutte le specifiche dei dati trascurandone i dettagli, per poi entrare nel merito di un concetto alla volta. Questo però è possibile solo quando si possiede una visione globale di tutte le componenti del sistema.

Strategia bottom-up

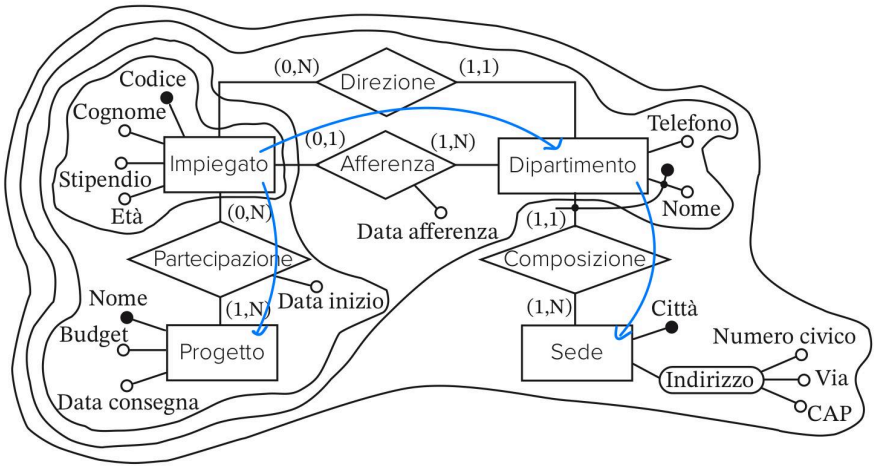
Le specifiche iniziali sono suddivise in componenti via via sempre più piccole. Le varie componenti vengono rappresentate da semplici schemi concettuali, questi vengono poi fusi fino a giungere allo schema concettuale finale. Questo schema si ottiene attraverso alcune trasformazioni elementari, denominate *primitive di trasformazione bottom-up*.
Vantaggio: si adatta a una decomposizione del problema in componenti più semplici, affrontabili in parallelo da progettisti diversi.
Svantaggio: richiede operazioni di integrazione di schemi concettuali diversi, che in caso di problemi complessi, presentano grosse difficoltà.



Strategia inside-out

Può essere vista come un caso particolare della strategia bottom-up. Si individuano inizialmente solo alcuni concetti importanti e poi si procede, a partire da questi. Si rappresentano prima i concetti in

relazione con i concetti iniziali, per poi muoversi verso quelli più lontani.



Rule of Thumb

Usare sempre un approccio misto:

- prima, si crea una bozza utilizzando le entità più rilevanti;
- successivamente, si decompone lo schema;
- infine, raffinare (top-down), integrare (bottom-up) ed espandere (inside-out).

Metodologia Generale

1. **Analisi dei requisiti**:
   - costruire un glossario dei termini;
   - analizzare i requisiti ed eliminare le ambiguità presenti;
   - raggruppare i requisiti in insiemi omogenei.
2. **Passo Base**:
   - individuare i concetti più rilevanti e rappresentarli in uno schema scheletro.
3. **Passo di decomposizione** (da effettuare se necessario):
   - effettuare una decomposizione dei requisiti con riferimento ai concetti presenti nello schema scheletro.
4. **Passo iterativo**, da ripetere per tutti i sottoschemi (se presenti) finchè ogni specifica è stata rappresentata:
   - raffinare i concetti presenti sulla base delle loro specifiche;
   - aggiungere nuovi concetti allo schema per descrivere specifiche non ancora descritte.
5. **Passo di integrazione** (da effettuare solo se è stato eseguito il passo 3):
   - integrare i vari sottoschemi in uno schema generale facendo riferimento allo schema scheletro.
6. **Analisi di qualità**:
   controllare la qualità dello schema e modificarlo.

Qualità di uno schema concettuale

- **Correttezza** → quando utilizza propriamente i costrutti messi a disposizione.
- **Completezza** → quando rappresenta tutti i dati di interesse e quando tutte le operazioni possono essere eseguite a partire dai concetti descritti nello schema.

- **Leggibilità** → quando rappresenta i requisiti in maniera naturale e facilmente comprensibile (es. mediante una scelta opportuna dei nomi da dare ai concetti).
- **Minimalità** → quando tutte le specifiche sui dati sono rappresentate una volta sola nello schema (es fonte di ridondanza: presenza dei cicli dovuta alla presenza di relazioni e/o generalizzazioni).