# Basi di Dati

Leonardo Valente

March 10, 2022

## Contents



# 1 Modello ER

Qualsiasi progetto, prima di essere mandato in produzione, segue un ciclo di vita, di solito composto da:

- Studio di fattibilità: definizione dei costi e delle priorità
- Raccolta di analisi e dei requisiti: studio delle proprietà del sistema
- Progettazione: di dati e funzioni
- Implementazione: ovvero la realizzazzione del progetto
- Validazione e collaudio: la fase di sperimentazione
- Funzionamento: fase di produzione

- Manutenzione: dove arriva il vero guadagno

Questo ciclo di vita segue un **modello a spirale**, quindi un ciclo dove in ogni fase è possibile andare avanti o indietro in base alle esigenze.

## 1.1 Progettazione

La progettazione delle applicazioni schematizza le operazioni sui dati e progetta il software. E' opportuno quindi seguire una **metodologia di progetto**. Essa ci permette di **suddividere** la progettazione in fasi indipenti, fornendo delle **strategie** da seguire e dei **criteri** di scelta.
Nella progettazione di Database ci sono 3 fasi di progettazione:

- Progettazione concettuale (modello ER)
- Progettazione logica
- Progettazione fisica

Ognuna di queste fasi si basa su un modello che permette di generare una rappresentazione formale (di solito uno schema) del nostro universo.

### 1.1.1 Progettazione concettuale

Traduce i requisiti del sistema in un modello ER, espresso in modo indipendente dalle scelte implementative.
La descrizione si deve concentrare sui **dati** e sulle loro **relazioni**, non sulle scelte implementative.

### 1.1.2 Progettazione logica

Consiste nella traduzione dello schema concettuale nel modello dei dati del DBMS (Modello relazionale).

### 1.1.3 Progettazione fisica

Completa lo schema logico ottenuto con le specifiche proprio dell'HW/SW scelto. Viene interamente effettuato dal DBMS.

## 1.2 Introduzione al Modello ER

Il modello Entità - Relazione è un **linguaggio grafico semi-formale** per la rappresentazione di schemi concettuali. Esso è ormai diventato uno *standard* nelle metodologie di progetto.

Iniziamo facendo una distinzione fra **Entità** e **Relazioni**.

- **Entità**: classe di oggetti dell'applicazione di interesse con proprietà comuni e con esistenza "autonoma", della quale si vogliono registrare fatti specifici.
  Le entità hanno degli attributi che la descrivono.
  Una **occorrenza** (o istanza) di una entità è il singolo oggetto creato sulla base dell'entità da cui deriva
  Le entità vengono rappresentate nel modello ER tramite dei *rettangoli*.

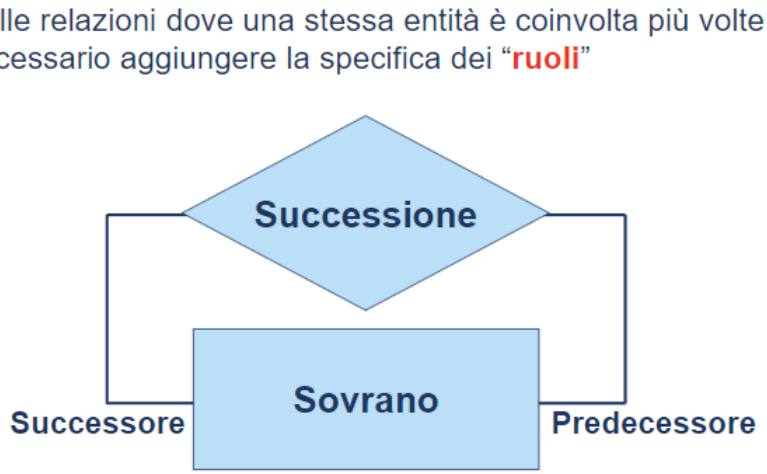
- **Attributi**: un attributo è definito su un dominio di valori. Esso è una funzione che associa ad ogni occorrenza **un** particolare valore (non di più!) Gli attributi possono essere composti e possono essere **qualsiasi cosa**.
  A meno che non venga definito il contrario, gli attributi sono **obbligatori**.

- **Relazione**: fatto che descrive un'azione o una situazione e che *stabilisce legami logici tra istanze di entità*.
  I legami possono essere fra più di due entità e il numero delle entià coinvolte ne determina il **grado**.
  Ogni relazione ha un nome che la identifica.
  Inoltre, le relazioni possono avere degli attributi, chiamati appunto **attributi delle relazioni** che modellano il legame tra le entità.

  Le relazioni vengono rappresentate nel modello ER tramite dei *rombi*.

### Associazioni ad anello

Un'associazione può coinvolgere "due o più volte" la stessa entità (associazione ricorisiva o ad anello).




## 2 Cardinalità delle Relazioni

La cardinalità delle relazioni non è altro che una coppia di valori che si associa a ogni entità che partecipa ad una relazione. Esprimono un **limite minimo** (cardinalità minima) e un **limite massimo** (cardinalità massima) di istanze della **relazione R** a cui può partecipare ogni istanza dell'**entità E**.
E' molto importante non sbagliare la *cardinalità massima*. E' la più importante fra le due. Se sbagliamo la cardinalità, sbagliamo il Database!

- **Molti a Molti** (3 tabelle). Quando zero o più istanze della prima entità si possono associare a zero o più istanze della seconda entità.
- **Uno a Molti** (2 tabelle). Quando ogni istanza della prima entità si può associare a una o più istanze della seconda entià.
- **Uno a Uno** (2 tabelle). Quando ogni istanza della prima entità si può associare a una e una sola istanza della seconda entià.

## 2.1 Identificatori di una entità

E' uno strumento per l'identificazione univoca delle occorrenze di una entità. Ci sono due tipi di identificatori:

- **Identificatore primario** (Primary Key), che serve per distinguere in modo univoco una istanza di una entità.
- **Identificatore esterno** (Foreign Key), che serve per fare riferimento ad una istanza di un'altra entità avente come identificatore primario l'identificatore esterno dell'oggetto che stiamo prendendo in considerazione.

Ogni entità deve possedere almeno un identificatore (primario), ma può averne in generale più di uno (esterno).

## 2.2 Relazione IS-A

Può accadere che tra due classi rappresentate da due entità nello schema concettuale sussista la **relazione IS-A**, cioè che **ogni istanza di una sia anche istanza dell'altra**.
La relazione IS-A si può definire tra *due entità*: **entità padre** e **entità figlio**.

### 2.2.1 Principio di ereditarietà

Ogni proprietà del padre è anche una proprietà del figlio, che però non viene esplicitamente riportata nello schema concettuale.
L'entità figlio può avere degli attributi in più rispetto all'entità padre.

### 2.2.2 Generalizzazione

Può capitare però, che l'entità padre può generalizzare diverse sottoentità rispetto ad un unico criterio. In questo caso si parla di **generalizzazione**. Una generalizzazione può essere di *due tipi*:

- **Completa**: l'unione delle istanze delle sottoentità è uguale all'insieme delle istanze dell'entità padre.
- **Non completa**.

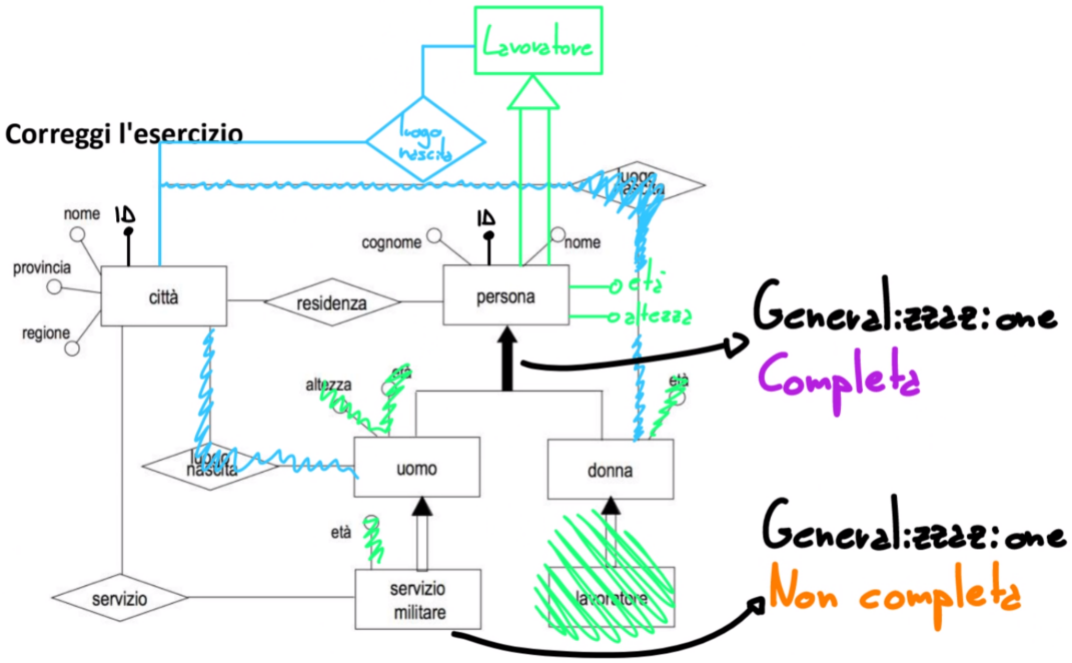



Figure 1: Mancano le *cardinalità*