

# La Progettazione delle Basi di Dati

## Progettazione Concettuale - Modello Entità-Relazione

Prof. Fedeli Massimo - Tutti i diritti riservati

ITS Academy - Fabbrica Digitale

1 dicembre 2025

# Sommario

- 1 Introduzione alla Progettazione
- 2 Livelli di Astrazione
- 3 Analisi Preliminare
- 4 Modello Entità-Relazione
- 5 Entità e Attributi
- 6 Chiavi
- 7 Relazioni
- 8 Relazioni Gerarchiche

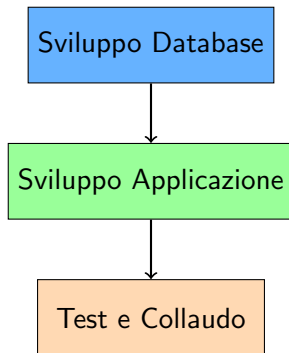
# La Progettazione del Database

## I passi principali della progettazione

- ➊ Analisi del problema
- ➋ Progettazione concettuale del database → **modello E-R**
- ➌ Progettazione logica del database → **schema logico**
- ➍ Progettazione fisica e implementazione
- ➎ Realizzazione delle applicazioni (Java, VB, PHP, etc.)

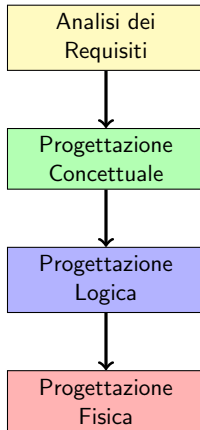
## Complessità

Ognuno di questi passi presenta criticità e implica operazioni complesse



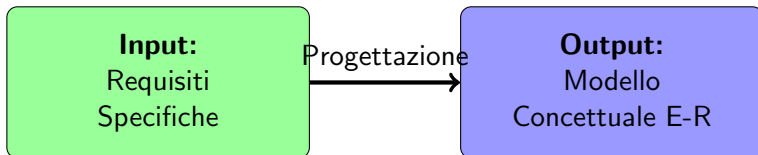
- Fase di sviluppo del database
- Fase di sviluppo dell'applicazione
- Test di funzionamento ed collaudo

# Le Fasi della Progettazione



## Progettazione Concettuale

L'output della progettazione concettuale è il **modello concettuale**

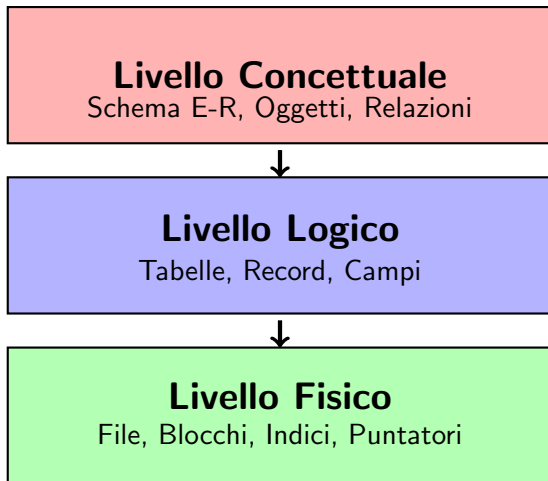


## Tre livelli di rappresentazione

La progettazione avviene a diversi livelli di astrazione:

- ① **Livello Concettuale** (livello di oggetti)
  - Rappresenta la realtà dei dati e le relazioni
  - Usa uno schema astratto
- ② **Livello Logico** (livello di record)
  - Organizzazione dei dati negli archivi
  - Descrive composizione e formato dei dati
- ③ **Livello Fisico**
  - Installazione fisica su disco
  - Ubicazione nelle memorie di massa

# Schema dei Livelli di Astrazione



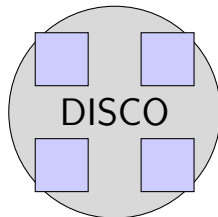


## Implementazione fisica dei dati

Il livello fisico è l'implementazione del livello logico sui supporti di memorizzazione fisica

### Si occupa di:

- Partizioni
- Puntatori
- Blocchi fisici
- Cluster
- Indici



Blocchi fisici

# Analisi Preliminare alla Modellazione

## Obiettivi dell'analisi

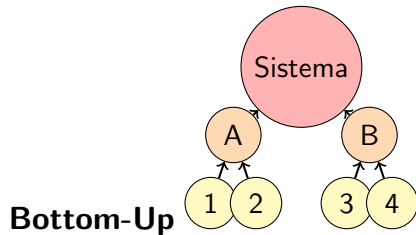
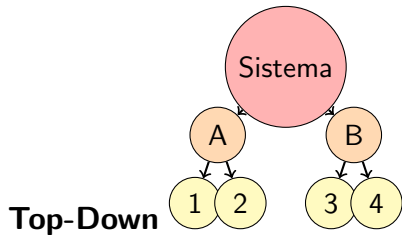
Individuare le esigenze del cliente o il dominio dell'applicazione

## Domande chiave

- Quali informazioni devono essere salvate?
- In che modo queste informazioni verranno manipolate?
- Chi sono gli utenti finali?

## Approcci:

- **Top-down**: dalla visione generale ai dettagli
- **Bottom-up**: dai dati ai concetti (preferibile per DB)



## Modelli disponibili

Al termine dell'analisi, si può ricorrere a:

### ① **Modello Entità-Relazione (E-R)**

- Approccio classico e più diffuso
- Ampiamente utilizzato nell'industria
- Standard de facto

### ② **Modello a Oggetti**

- Approccio più recente
- Interessante dal punto di vista accademico
- Meno diffuso commercialmente

## Scelta consigliata

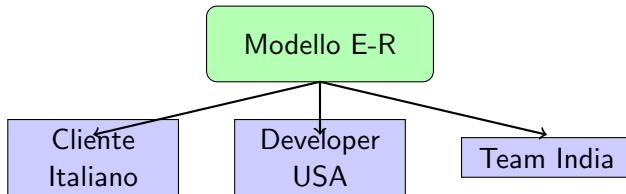
La stragrande maggioranza delle applicazioni usa l'**approccio E-R**

# Modello Entità-Relazione: Vantaggi

## Indipendenza dal linguaggio

Il modello E-R supera le barriere linguistiche essendo:

- Assolutamente indipendente dal linguaggio scritto o parlato
- Comprensibile a tutti gli stakeholder
- Rappresentazione grafica universale

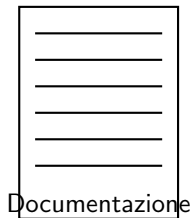


## Importante

Al modello E-R viene **sempre** affiancato un documento tecnico

### Il documento descrive:

- Dettagli degli oggetti
- Vincoli di integrità
- Regole di business
- Casi particolari
- Assunzioni fatte



# Progettazione Concettuale: Scopi

## Primo scopo

Fornire la **rappresentazione grafica** di tutti gli oggetti del database

- Visualizzazione completa della struttura
- Verifica del flusso delle informazioni
- Controllo prima dello sviluppo

## Secondo scopo

Base per la **creazione del database fisico**

- Gli sviluppatori lo usano come blueprint
- Guida per creare tabelle e relazioni
- Riferimento per tutti gli oggetti

# Principi di Qualità del Modello

Un buon modello concettuale deve rispettare quattro principi:

① **Correttezza**

- Uso appropriato degli strumenti di modellazione

② **Completezza**

- Rappresentazione di tutti gli aspetti rilevanti

③ **Chiarezza**

- Modello leggibile e comprensibile

④ **Indipendenza**

- Non dipendente da strumenti specifici



# Correttezza del Modello

## Utilizzo appropriato degli strumenti

Il modello deve usare gli strumenti secondo la loro finalità specifica

## Esempio

Un diagramma E-R serve a rappresentare **relazioni tra entità**, non flussi di processo

- Corretto: rappresentare studenti e corsi con la loro relazione
- Errato: usarlo per descrivere un workflow aziendale

## Coerenza semantica

Ogni simbolo deve rispettare le convenzioni:

- Rettangoli per le entità
- Rombi per le relazioni
- Ovali per gli attributi

# Completezza del Modello

## Modellazione completa

Rappresentare **tutti** gli aspetti significativi del dominio applicativo

- Tutte le entità rilevanti
- Tutte le relazioni importanti
- Tutti gli attributi necessari
- Tutti i vincoli di integrità



# Chiarezza del Modello

## Leggibilità e comprensibilità

Il modello deve essere facilmente leggibile e le informazioni comprensibili

### **Come ottenere chiarezza:**

- Nomi significativi per entità e attributi
- Layout ordinato e ben organizzato
- Evitare sovrapposizioni di linee
- Raggruppare concetti correlati
- Usare colori o evidenziazioni (quando possibile)
- Limitare la complessità di ogni diagramma

## Regola pratica

Se un diagramma non è comprensibile a prima vista, va semplificato!

# Indipendenza del Modello

## Astrazione tecnologica

Il modello concettuale deve essere **astratto e neutrale**

### NON dipendente da:

- DBMS specifici
- Linguaggi di programmazione
- Tecnologie particolari
- Piattaforme hardware

### Vantaggi:

- Longevità del modello
- Flessibilità implementativa
- Portabilità
- Manutenibilità

## Cosa descrivere

**Cosa** fare, non **come** implementarlo

## Dipendente

“Usare PostgreSQL con trigger per gestire cancellazioni”

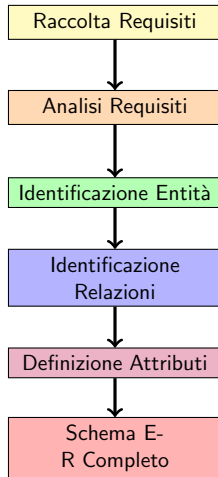
- Riferimento a tecnologia specifica
- Dettagli implementativi
- Non portabile

## Indipendente

“Una prenotazione cancellata deve aggiornare la disponibilità della risorsa”

- Regola di business
- Tecnicamente neutrale
- Implementabile ovunque

# Fasi della Progettazione Concettuale



# Caratteristiche della Progettazione Concettuale

## Rigorosa

Per non lasciare dubbi sulle caratteristiche della base di dati

- Precisione nelle definizioni
- Specifiche non ambigue
- Documentazione dettagliata

## Semplice nei formalismi

Per consentire lettura e comprensione anche agli utenti non tecnici

- Notazione grafica intuitiva
- Simboli standard
- Documentazione in linguaggio naturale

## Importante

Gli utenti devono essere certi che i progettisti abbiano compreso le loro esigenze

## Aspetti rilevanti

Concetti e formalismi utilizzati nella costruzione del modello entità/associazioni

### Caratteristiche:

- Molto utilizzato nella progettazione
- **Non ha rappresentazione standardizzata**
- Composto da: entità, associazioni e attributi
- Diversi modi di rappresentazione

### Notazioni principali:

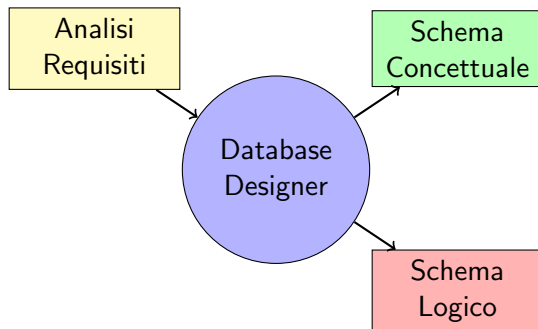
- ① Notazione classica (Chen)
- ② Notazione standard UML

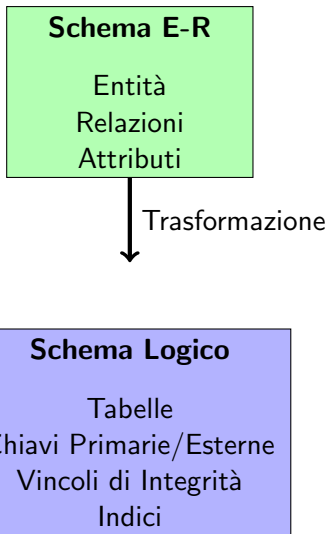


# Chi Realizza il Modello? Il Database Designer

## Ruolo del Database Designer

Responsabile dell'astrazione dei dati dal mondo reale





# I Compiti del Database Designer

## Primo compito

Analizzare le informazioni raccolte durante l'analisi dei requisiti

## Obiettivo principale

Costruire il modello di base, da raffinare fino al completamento

### Prime operazioni:

- 1 Classificare gli oggetti come **entità** o **attributi**
- 2 Partire dalla documentazione del progetto
  - Raccolta e analisi della documentazione
  - Definizione del glossario dei termini

## Tipi di documentazione

### **Documentazione specifica del progetto:**

- Note delle riunioni tecniche
- Richieste del cliente
- Appunti dalle interviste agli utenti
- Documentazione scritta ad hoc

### **Documentazione esistente:**

- Normative generali e del settore
- Regolamenti interni
- Procedure aziendali

### **Sistema esistente:**

- Sistema da rimpiazzare
- Specifiche di integrazione con sistemi esistenti

# Identificare le Entità

## Che cos'è un'entità?

Un'entità rappresenta un **oggetto** (concreto o astratto) del mondo reale che ha un'esistenza autonoma

**STUDENTE**

Mario Rossi

**PROFESSORE**

Prof. Bianchi

**CORSO**

Basi di Dati

**AULA**

Aula 101

# Esempi di Entità

## Entità Concrete:

- Persona
- Automobile
- Edificio
- Prodotto
- Computer

## Entità Astratte:

- Corso universitario
- Prenotazione
- Transazione
- Progetto
- Evento

## Caratteristica comune

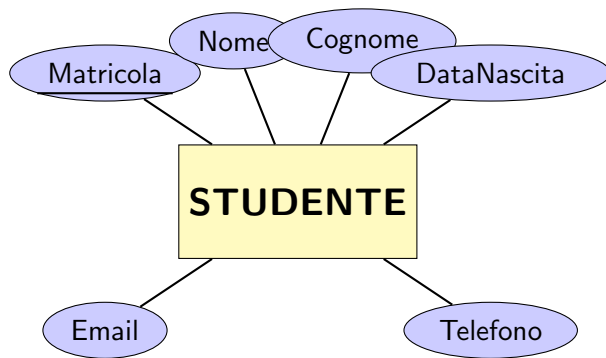
Tutte hanno proprietà (attributi) che le descrivono e un'identità univoca

## Che cos'è un attributo?

Gli attributi **descrivono** un'entità: corrispondono ai campi dei record

### Regole per individuare gli attributi:

- ① Gli attributi devono essere **atomici**
  - Non ulteriormente scomponibili
- ② Gli attributi **derivati** non dovrebbero essere memorizzati
  - Esempio: età (derivabile dalla data di nascita)
- ③ Utilizzare **codici** per classificare gli attributi
  - Quando si presenta l'opportunità





## Esempio CORRETTO

**PERSONA:** Nome, Cognome, Via, Città, CAP

## Esempio ERRATO

**PERSONA:** NomeCompleto, Indirizzo

- NomeCompleto non è atomico (contiene nome e cognome)
- Indirizzo non è atomico (contiene via, città, CAP)

## Perché l'atomicità è importante:

- Facilita le ricerche specifiche
- Migliora l'ordinamento
- Evita ridondanza
- Semplifica aggiornamenti

## Regole per i nomi

### **I nomi degli oggetti devono essere unici**

- Evitare ambiguità
- Nessuna duplicazione nel modello

### **Devono avere significato per l'utente finale**

- Usare terminologia del dominio
- Comprensibili senza spiegazioni

### **Contenere il minimo numero di parole necessarie**

- Descrizione univoca e accurata
- Bilanciare brevità e chiarezza

## Nomi CORRETTI

- Studente
- DataIscrizione
- CorsoDiLaurea
- NumeroMatricola
- IndirizzoEmail

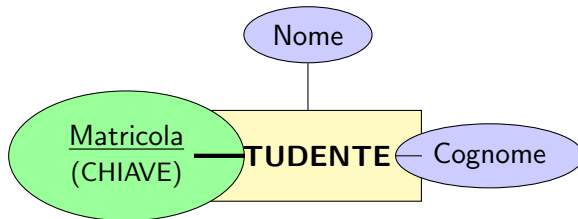
## Nomi da EVITARE

- S (troppo generico)
- Data1 (non significativo)
- CorsoDiLaureaTriennaleInInformatica (troppo lungo)
- Num (abbreviazione oscura)
- x\_email (non user-friendly)

# Il Concetto di Chiave

## Definizione

Una **chiave** è un attributo (o insieme di attributi) che identifica univocamente un'istanza di un'entità



## Proprietà della chiave:

- **Univocità**: nessun duplicato
- **Minimalità**: minimo numero di attributi necessari

## ① Chiave Primaria (Primary Key - PK)

- Identifica univocamente ogni istanza
- Non può contenere valori NULL
- Unica per ogni entità

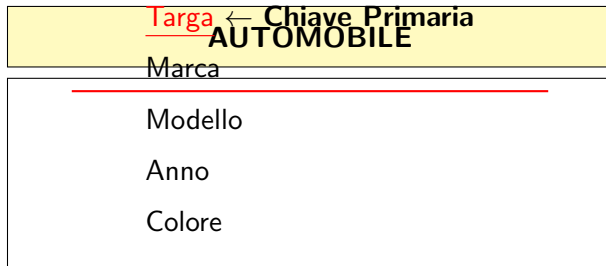
## ② Chiave Candidata

- Potenziale chiave primaria
- Soddisfa requisiti di univocità

## ③ Chiave Esterna (Foreign Key - FK)

- Riferisce la chiave primaria di un'altra entità
- Stabilisce relazioni tra entità

# Esempio di Chiave Primaria



La **Targa** identifica univocamente ogni automobile

# Chiavi Composte

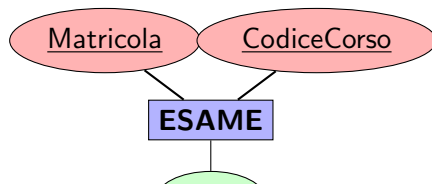
## Definizione

Una chiave composta è formata da **più attributi** che insieme identificano univocamente un'istanza

## Esempio

Entità **ESAME**:

- Matricola (da sola non univoca)
- CodiceCorso (da solo non univoco)
- **(Matricola, CodiceCorso)** → Chiave composta univoca



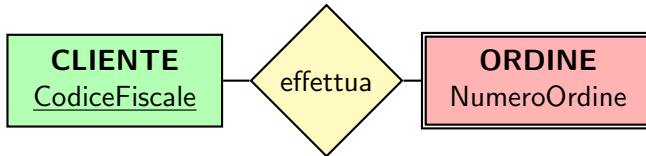
# Entità Forti e Deboli

## Entità Forti

Hanno una chiave primaria propria e esistenza autonoma

## Entità Deboli

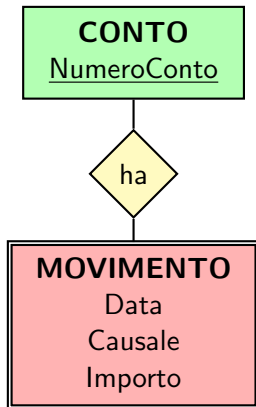
Non hanno chiave primaria propria, dipendono da altre entità



Nota: il doppio bordo indica un'entità debole



## Esempio: Conto Corrente



Movimento non  
ha chiave primaria  
propria: dipende da  
Conto

# Soluzione: Aggiunta di Contatore

## Problema

Entità deboli possono creare complessità nella gestione

## Soluzione

Aggiungere un attributo **numero progressivo** auto-generato

### MOVIMENTO

IdMovimento

Data

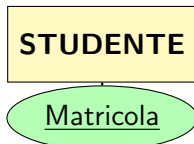
Causale

Importo

### Vantaggi:

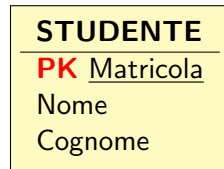
- Entità diventa forte
- Ordinamento cronologico
- Identificazione univoca

# Chiavi: Notazione Classica vs UML



## Notazione Classica

Chiave sottolineata nell'ovale



## Notazione UML

Chiave con marcatore PK

## Nota

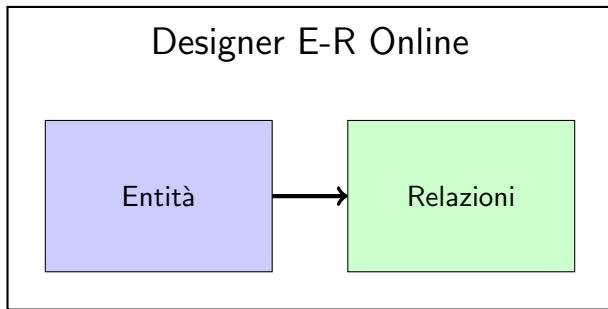
Entrambe le notazioni sono valide e ampiamente usate

# Tool per Diagrammi E-R

## Designer Online

Disponibile gratuitamente:

<https://designer-basic.polito.it/>



# Molteplicità delle Relazioni

## Definizione

La molteplicità indica il numero di possibili istanze di un'entità associate ad un'istanza dell'altra entità

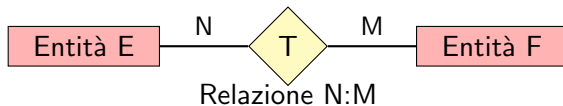
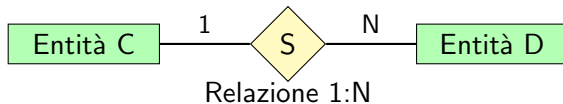
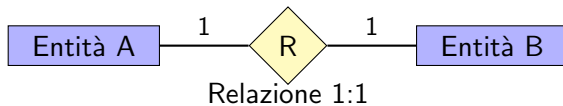
**Rappresentazione:** coppia di valori min..max

- 1..1 - esattamente uno
- 0..1 - zero o uno (opzionale)
- 1..N - uno o molti
- 0..N - zero o molti

## Concetti importanti

- **Valore minimo** → Obbligatorietà (0=facoltativo, 1=obbligatorio)
- **Valore massimo** → Cardinalità (1=uno, N=molti)

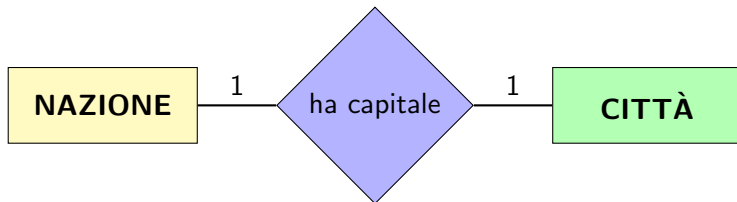
## Cardinalità - Notazione Classica



# Le Relazioni di Tipo 1:1

## Definizione

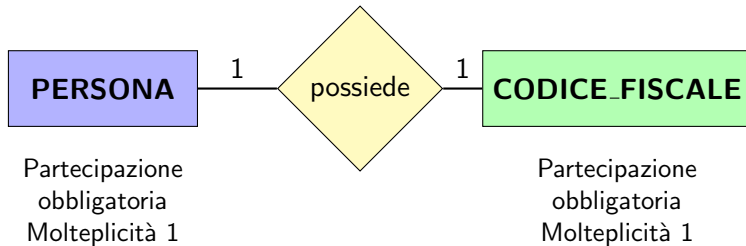
Ad ogni istanza della prima entità corrisponde **al più** un'istanza della seconda, e viceversa



## Esempio

- Ogni nazione ha **una** capitale ( $1 \rightarrow 1$ )
- Ogni capitale appartiene a **una** nazione ( $1 \leftarrow 1$ )

## Relazione 1:1 - Altro Esempio

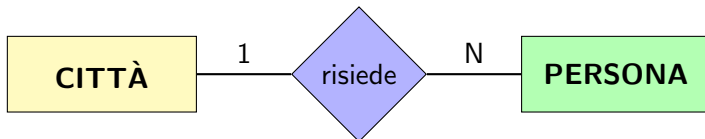




# Relazioni Uno a Molti (1:N)

## Definizione

Ad un'istanza della prima entità corrispondono **più istanze** della seconda, ma ad ogni istanza della seconda corrisponde **al più una** istanza della prima



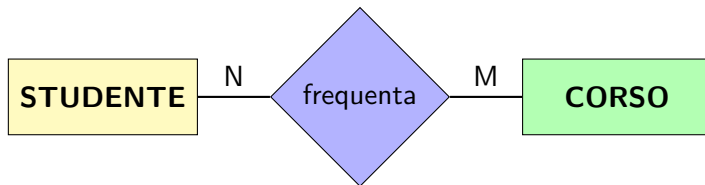
## Interpretazione

- Una città ha **molte** persone residenti
- Una persona risiede in **una sola** città

# Relazioni Molti a Molti (N:M)

## Definizione

Ad un'istanza della prima entità corrispondono **più istanze** della seconda, e viceversa

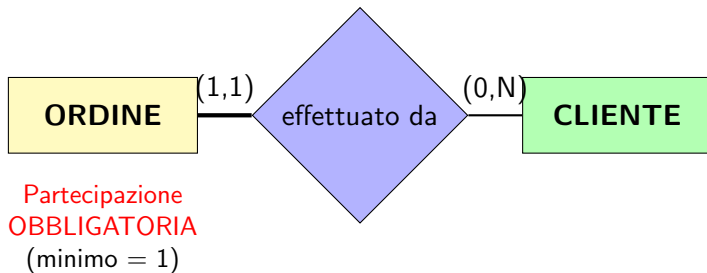


## Interpretazione

- Uno studente frequenta **più corsi**
- Un corso è frequentato da **più studenti**

## Definizione

Un'istanza di un'entità **deve necessariamente** partecipare alla relazione

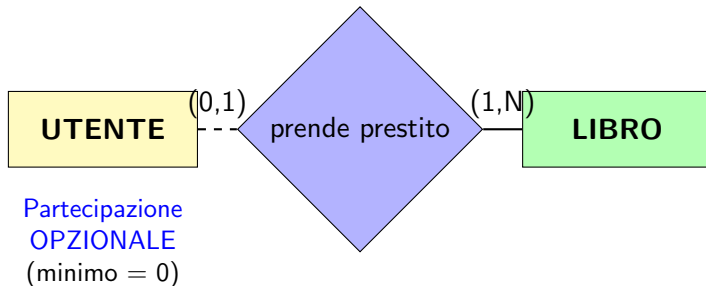


Un ordine DEVE avere un cliente

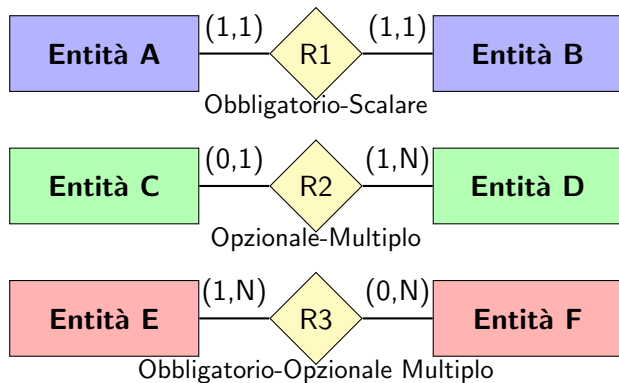
# Esistenza Opzionale

## Definizione

Un'istanza di un'entità **può** partecipare facoltativamente alla relazione



Un utente PUÒ prendere in prestito un libro (o anche nessuno)



# Esempi di Vincoli di Cardinalità

## Esempio 1: Persona - Codice Fiscale

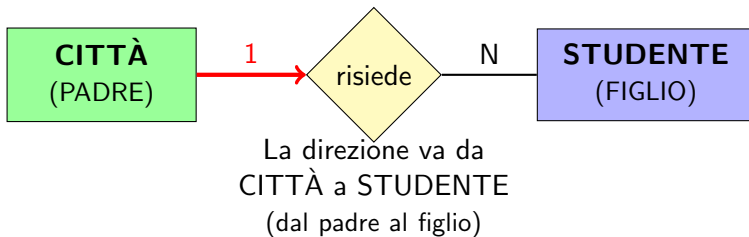
- Una persona possiede **almeno 1 e massimo 1** codice fiscale ( $\rightarrow$ )
- Un codice fiscale appartiene a **almeno 1 e massimo 1** persona ( $\leftarrow$ )

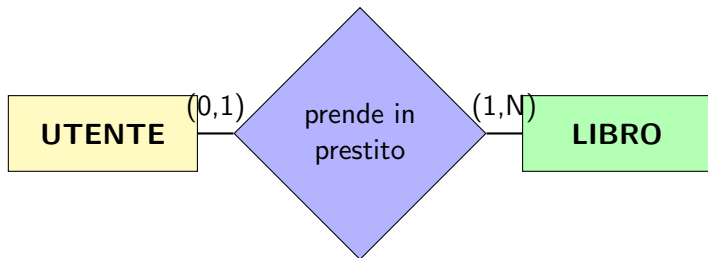
## Esempio 2: Persona - Città

- Una persona risiede in **almeno 1 e al massimo 1** città ( $\rightarrow$ )
- In una città risiedono **almeno 1 e al massimo N** persone ( $\leftarrow$ )

## Concetto di "padre"

Nella relazione uno-a-molti, l'entità con cardinalità 1 è detta **padre**





## Lettura da sinistra (UTENTE):

- $(0,$  → può prendere in prestito
- $,1)$  → un solo libro

## Lettura da destra (LIBRO):

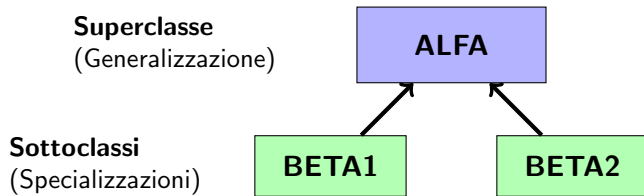
- $(1,$  → un libro deve essere prestato
- $,N)$  → a uno o più utenti (nel tempo)



# Relazione Gerarchica tra Entità

## Definizione

Situazioni in cui tra le entità può essere stabilita una gerarchia, simile alle classi nella programmazione OOP



## Due vincoli fondamentali:

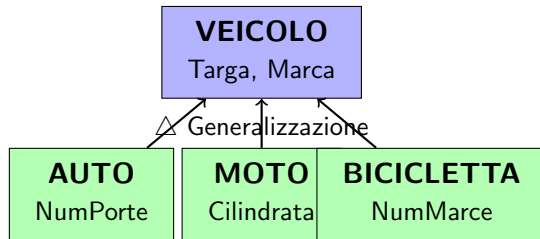
### 1. Vincolo di Struttura (Ereditarietà)

- La sottoclasse eredita tutti gli attributi della superclasse
- La sottoclasse partecipa a tutte le associazioni della superclasse
- La sottoclasse può avere attributi e associazioni aggiuntive

### 2. Vincolo di Insieme (Subset)

- Ogni oggetto della sottoclasse è anche un oggetto della superclasse
- La sottoclasse è un **sottoinsieme** della superclasse

## Esempio: Gerarchia Veicoli



Auto, Moto e Bicicletta **specializzano** Veicolo

## Concetto di Copertura

Le sottoentità non sempre contemplano tutti gli elementi della classe padre

### Due dimensioni indipendenti:

#### ① Confronto con l'unione:

- **Totale**: la superclasse è l'unione delle sottoclassi
- **Parziale**: la superclasse contiene l'unione delle sottoclassi

#### ② Confronto tra sottoclassi:

- **Esclusiva**: le sottoclassi sono disgiunte
- **Sovrapposta**: può esistere intersezione tra sottoclassi

# Grazie per l'attenzione!

Domande?

Prof. Fedeli Massimo  
ITS Academy - Fabbrica Digitale

