

BASI DI DATI I

- Progettazione di una Base di Dati
- Modellazione del DB: ER vs UML
- Entità e Attributi (UML vs ER)
- Esempio: Il database *Company*
- Le Relazioni (ER) e le Associazioni (UML)
- Esempi

PROGETTAZIONE DI UNA BASE DI DATI



PROGETTAZIONE DI BASI DI DATI

- È una delle attività del processo di sviluppo dei sistemi informativi.
- Va quindi inquadrata in un contesto più generale:
 - il ciclo di vita dei sistemi informativi.



SISTEMA INFORMATIVO

- **Componente (sottosistema)**

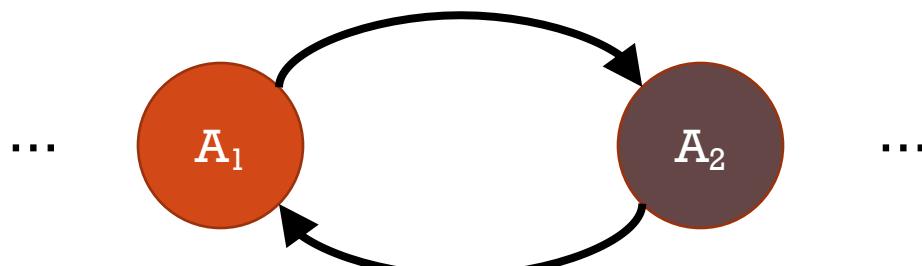
di una organizzazione che gestisce (acquisisce, elabora, conserva, produce)

le informazioni di interesse (cioè utilizzate per il perseguitamento degli scopi dell'organizzazione).



IL CICLO DI VITA DEI SISTEMI INFORMATIVI

- Insieme e sequenzializzazione delle attività svolte da analisti, progettisti, utenti, nello sviluppo e nell'uso dei sistemi informativi.
- È un'attività iterativa, quindi è rappresentata attraverso un **ciclo**.



IL CICLO DI VITA DEI SISTEMI INFORMATIVI



- Il ciclo di vita è una attività iterativa, rappresentata tramite un ciclo



FASI (TECNICHE) DEL CICLO DI VITA

■ Studio di fattibilità

- Si analizzano le potenziali aree di applicazione, si effettuano degli studi di *costi/benefici*, si determina la complessità di dati e processi, e si impostano le priorità tra le applicazioni.

■ Raccolta e analisi dei requisiti

- Comprende una raccolta dettagliata dei requisiti con interviste ai potenziali utenti, per definire le funzionalità del sistema.

■ Progettazione di dati e funzioni

■ Realizzazione

- Si implementa il sistema informativo, si carica il DB e si implementano e si testano le transazioni.

■ Validazione e collaudo

- Si verifica che il sistema soddisfi i requisiti e le performance richieste.

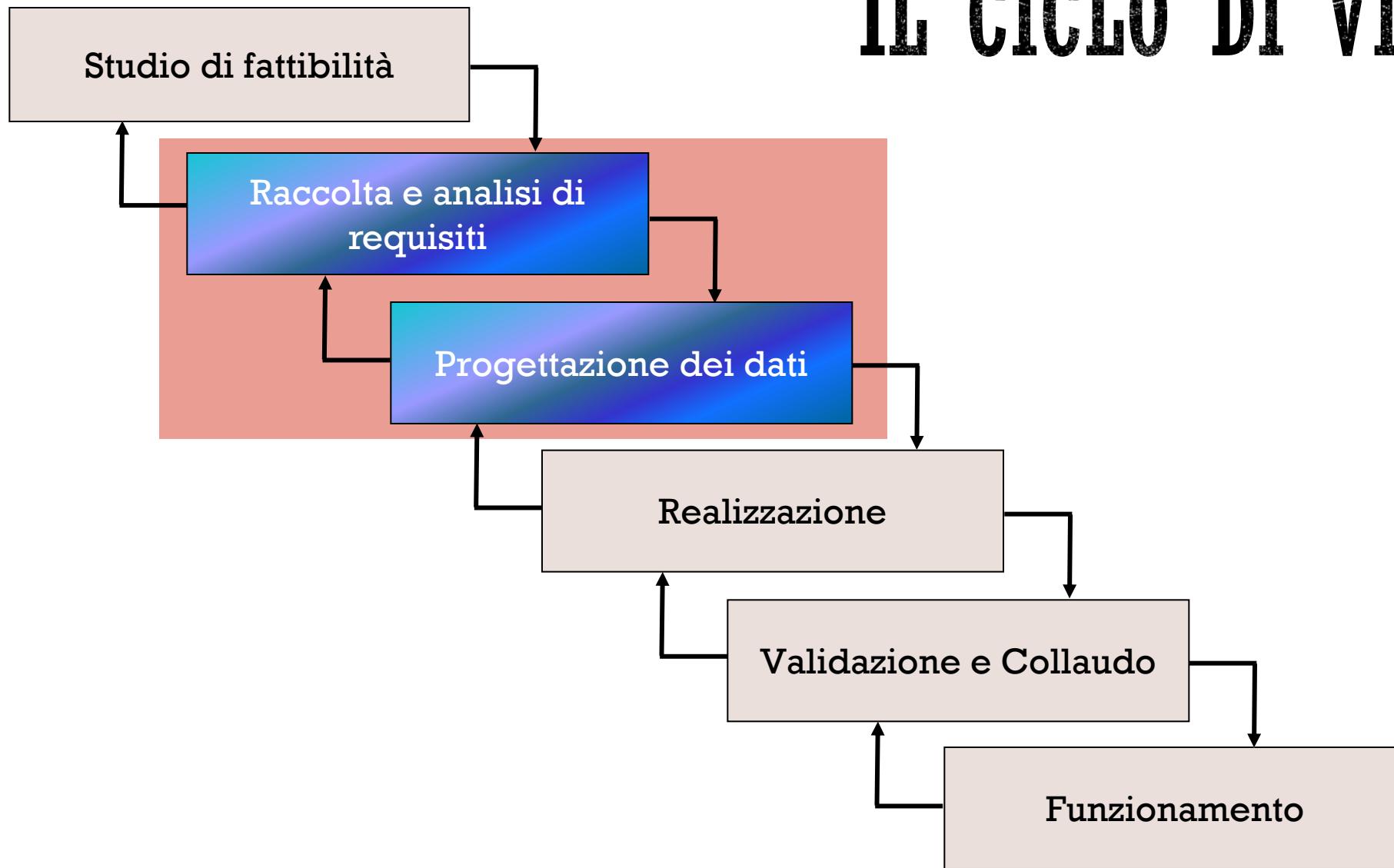
■ Funzionamento

- La fase operativa del nuovo sistema parte quando tutte le funzionalità sono state validate.

Il rilascio può essere preceduto da una fase di addestramento del personale al nuovo sistema.

Se emergono nuovi funzionalità da implementare, si ripetono i passi precedenti, per includerle nel sistema (*manutenzione*).

IL CICLO DI VITA



LA METODOLOGIA DI PROGETTO

- Per garantire prodotti di buona qualità è opportuno seguire una
 - metodologia di progetto, con:
 - articolazione delle attività in fasi indipendenti tra loro;
 - strategie da seguire nei vari passi e criteri di scelta (alternative),
 - modelli di rappresentazione per descrivere i dati in ingresso e uscita delle varie fasi.
 - proprietà:
 - generalità,
 - qualità del prodotto in termini di correttezza, completezza ed efficienza rispetto alle risorse impiegate,
 - facilità d'uso delle strategie e dei modelli.



MODELLO DEI DATI

- I prodotti della varie fasi sono schemi di alcuni modelli di dati:
 - Schema concettuale
 - Schema logico
 - Schema fisico



MODELLO DEI DATI

- È un insieme di costrutti utilizzati per organizzare i dati di interesse e descriverne la dinamica.
- Componente fondamentale: meccanismi di strutturazione (o costruttori di tipo).
 - Come nei linguaggi di programmazione esistono meccanismi che permettono di definire nuovi tipi, così ogni modello dei dati prevede alcuni costruttori .
- **Esempio:** il modello relazionale prevede il costruttore relazione, che permette di definire insiemi di record omogenei.



SCHEMI E ISTANZE

- In ogni base di dati esistono:
 - lo **schema**, sostanzialmente invariante nel tempo, che ne descrive la struttura (*aspetto intensionale*):
 - **Es:** nel modello relazionale, le intestazioni delle tabelle.
 - l'**istanza**, i valori attuali, che possono cambiare anche molto rapidamente (*aspetto estensionale*):
 - **Es:** nel modello relazionale, il “corpo” di ciascuna tabella.



DUE TIPI (PRINCIPALI) DI MODELLI

- **Modelli concettuali:** permettono di rappresentare i dati in modo indipendente da ogni sistema:
 - cercano di descrivere i concetti del mondo reale,
 - sono utilizzati nelle fasi preliminari di progettazione.
- **Modelli logici:** utilizzati nei DBMS esistenti per l'organizzazione dei dati:
 - utilizzati dai programmi,
 - indipendenti dalle strutture fisiche.

Esempi: **relazionale**, reticolare, gerarchico, a oggetti.



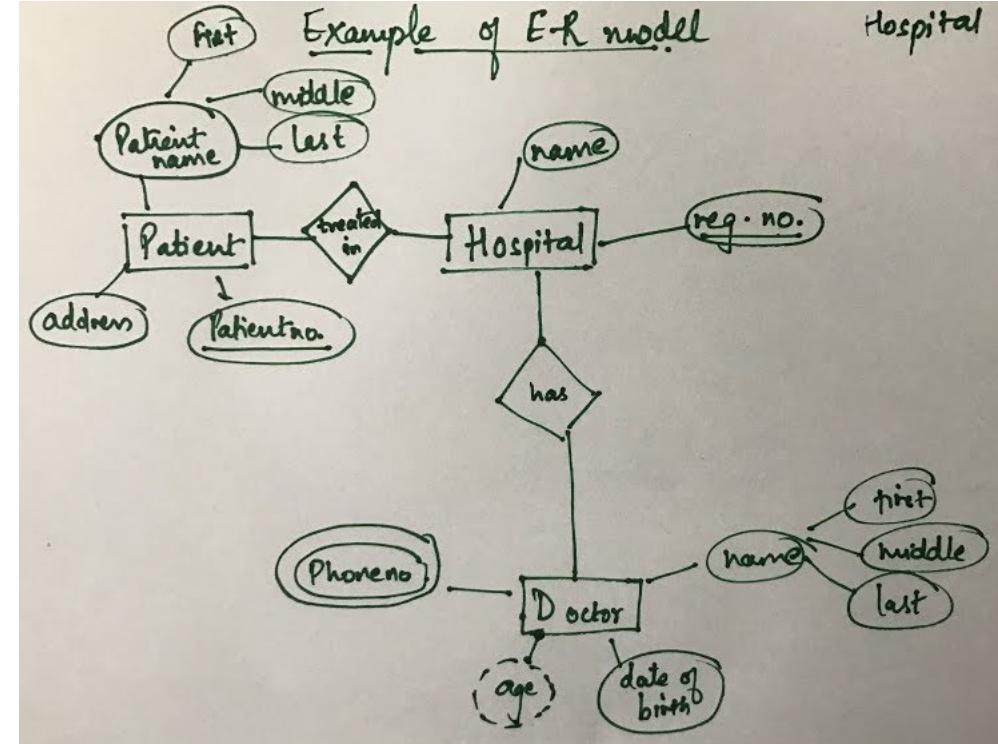
MODELLI CONCETTUALI, *PERCHÉ?*

- Proviamo a modellare una applicazione definendo direttamente lo schema logico della base di dati:
 - da dove cominciamo?
 - rischiamo di perderci subito nei dettagli;
 - dobbiamo pensare subito a come correlare le varie tabelle (chiavi, relazioni, etc.);
 - i modelli logici sono rigidi.



MODELLI CONCETTUALI, PERCHÉ? (2)

- Servono per ragionare sulla realtà di interesse, indipendentemente dagli aspetti realizzativi.
- Permettono di rappresentare le classi di dati di interesse e le loro correlazioni.



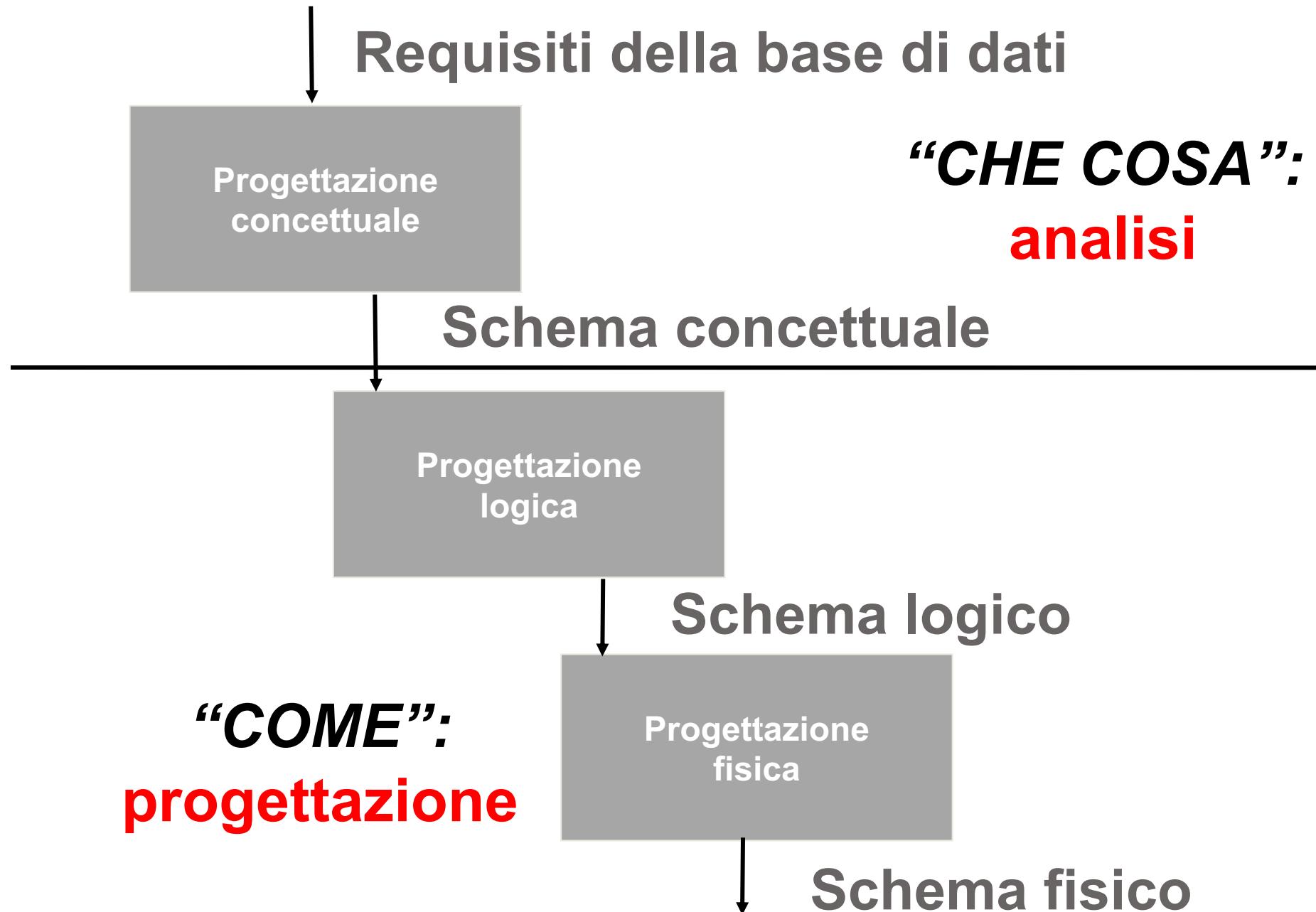
- Prevedono efficaci rappresentazioni grafiche (utili anche per documentazione e comunicazione).

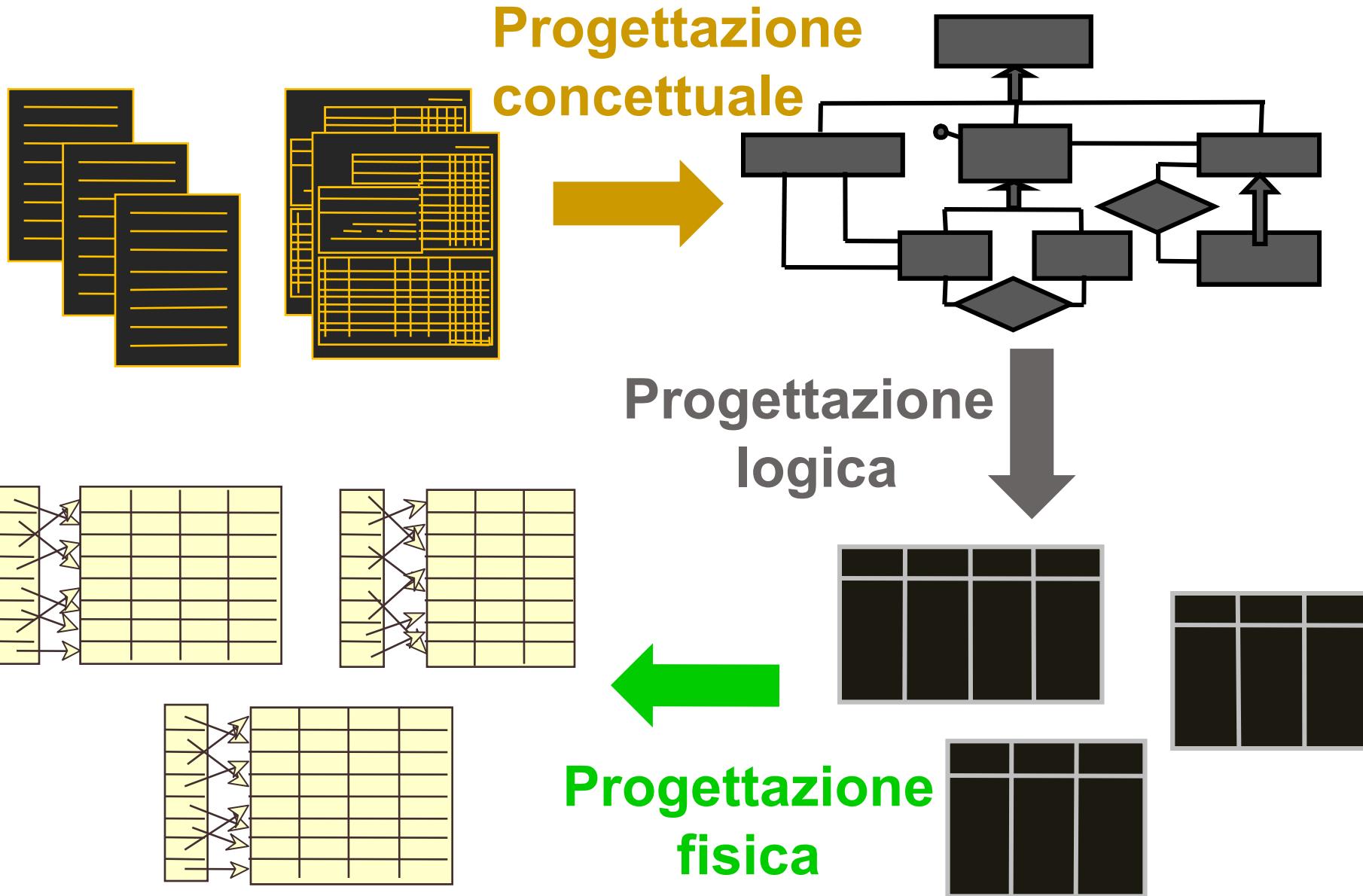
IL PROCESSO DI PROGETTAZIONE DEL DATABASE

Il processo consta di quattro fasi:

- 1. Raccolta e analisi dei requisiti**
- 2. Disegno del database concettuale (Progettazione Concettuale)**
- 3. Disegno del database logico (Progettazione Logica)**
- 4. Disegno del database fisico (Progettazione Fisica)**







PROGETTAZIONE CONCETTUALE

- Individuare l'informazione rilevante da memorizzare per soddisfare le richieste informative e funzionali del caso applicativo
- Interesse specifico per la parte statica del problema: i dati
- Individuazione nel problema degli aspetti informativi rilevanti:
 - Gli elementi informativi di base (Entità)
 - Il modo in cui sono interrelati gli elementi informativi di base (Associazioni)
 - I vincoli che i dati memorizzati nel database dovrebbero soddisfare per garantire la qualità e la coerenza dell'informazione
 - Le più importanti forme di interazione che gli utenti possono richiedere al database (ad esempio le interrogazioni della base di dati)
 - La frequenza con cui le interazioni avvengono
 - Il numero di utenti che simultaneamente possono interagire.



CARATTERISTICHE DELLA PROGETTAZIONE CONCETTUALE

- Dovrebbe essere indipendente da uno specifico modello dei dati e da uno specifico DBMS (scelta da adottare successivamente)
- Deve escludere dettagli implementativi legati a alla scelta del modello dei dati o alla scelta di uno specifico DBMS



DOCUMENTAZIONE ALLA PROGETTAZIONE CONCETTUALE

- Per la descrizione statica delle entità e delle loro associazioni si utilizzeranno i Class Diagram di UML;
- Tradizionalmente viene proposta un linguaggio di descrizione grafico chiamato Schemi E-R (Entità-Relazione);
 - E' uno schema facilmente esprimibile mediante Class Diagram UML
 - I Class Diagram di UML sono lo standard per la rappresentazione delle strutture dati nella programmazione ad oggetti
 - I Class Diagram sono indipendenti dal modello dei dati relazionale e agevolano le descrizioni per il modello relazionale
- Dizionari delle entità e delle associazioni
 - Descrivono e commentano integrando la presentazione del Class Diagram
- Dizionario dei vincoli
- Dizionario delle interrogazioni e indicazione della loro frequenza

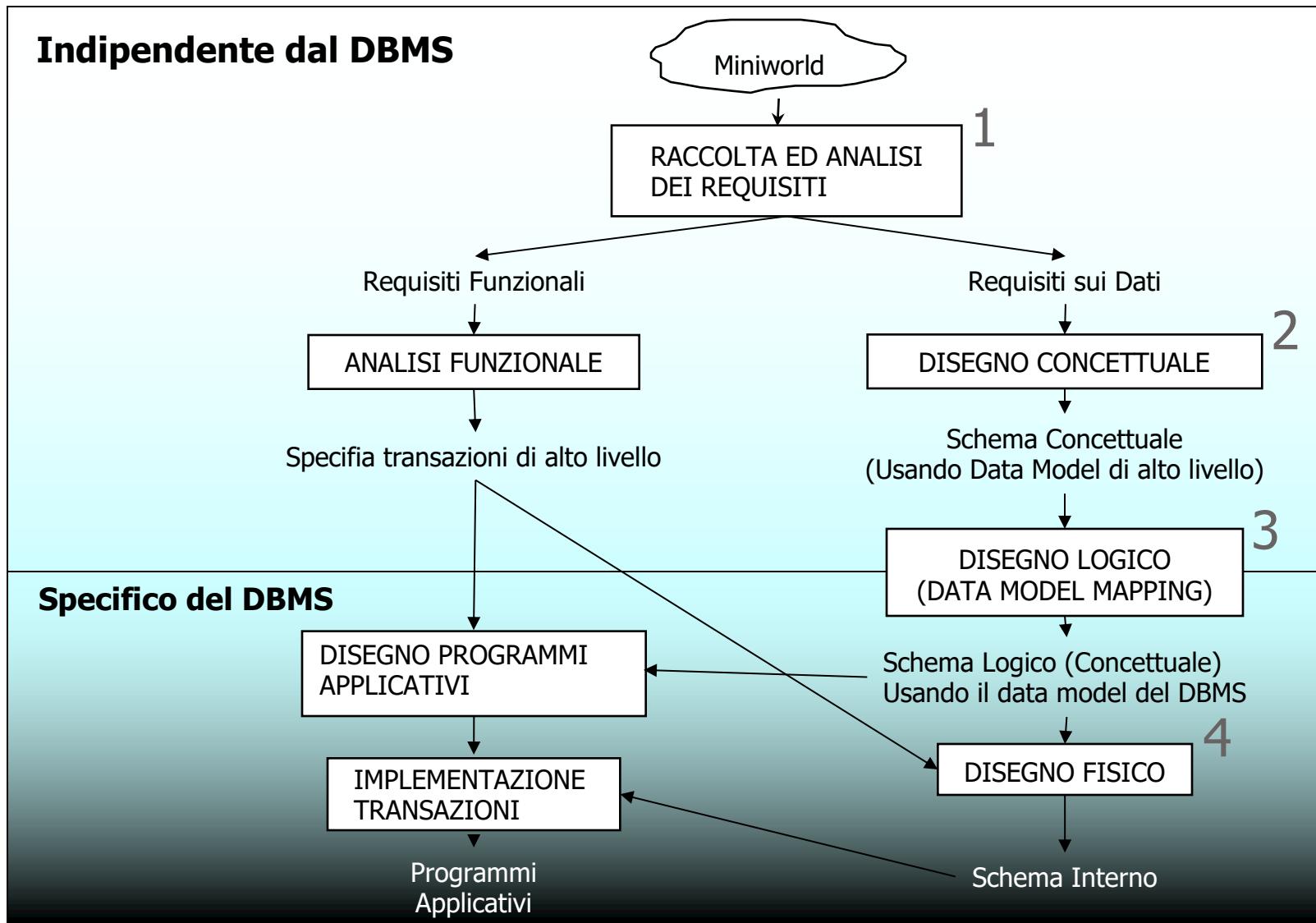


PASSI SUCCESSIVI ALLA PROGETTAZIONE CONCETTUALE

- Progettazione logica
 - Scelta del modello dei dati
 - Codifica delle entità ed associazioni nel modello dei dati
- Progettazione fisica
 - Scelta del DBMS
 - Definizione dei metadati utilizzando il DDL del DBMS
 - Implementazione dei vincoli di consistenza
 - Definizione dei dettagli implementativi utilizzando le funzionalità specifiche del DBMS scelto



LE FASI DI PROGETTAZIONE DI UN DATABASE



1. RACCOLTA E ANALISI DEI REQUISITI

- Il progettista del DB intervista i potenziali utenti dei DB per capire e documentare i requisiti utente.

Output:

- Requisiti sui dati
- Requisiti funzionali
 - Operazioni definite dall'utente (transazioni) che saranno applicate al DB.
 - *Es:* aggiornamenti, ricerche, ...
 - Specifica dei requisiti funzionali attraverso Data Flow Diagrams.



2. DISEGNO DEL DATABASE CONCETTUALE

- Creazione dello schema concettuale, usando un data model concettuale ad alto livello.

Output:

- Descrizione concisa dei requisiti utente dei dati inclusiva di una descrizione dettagliata di tipi di dati, relazioni e vincoli fatta usando i concetti del data model ad alto livello.
 - Poiché non vi sono dettagli implementativi, tale descrizione può essere usata per comunicare con utenti non tecnici.
 - Tale documentazione può essere usata anche come riferimento per assicurarsi di aver considerato tutti i requisiti dell'utente.
- Questo approccio abilita il progettista a concentrarsi sui dati ignorando i dettagli di memorizzazione.
- Dopo aver disegnato lo schema concettuale, le operazioni di base del data model possono essere usate per specificare le operazioni di alto livello individuate durante l'analisi funzionale (*passo 1*).



3. DISEGNO DEL DATABASE LOGICO

- Implementazione del database usando un DBMS commerciale.
 - Detto anche data model mapping.
 - Molti DBMS usano un data model di implementazione, in modo da trasformare facilmente lo schema concettuale da data model ad alto livello in data model di implementazione.

Output:

- Schema del database nel data model di implementazione specifico del DBMS scelto.



4. DISEGNO DEL DATABASE FISICO

- Specifica delle strutture di memorizzazione interne, degli access path e dell'organizzazione dei file.
- Progettazione dei programmi applicativi e implementazione delle transazioni.



IL PROCESSO DI PROGETTAZIONE DEL DATABASE

1. Raccolta e analisi dei requisiti
 2. Disegno del database concettuale
 3. Disegno del database logico
 4. Disegno del database fisico
-
- La prima fase mal si presta all'utilizzo di rigorosi formalismi, essendo essenzialmente un grosso documento di testo.
 - La seconda fase, invece, permette la definizione di modelli formali per supportare il lavoro del progettista.



MODELLAZIONE DEL DB: ER VS UML



MODELLAZIONE DEL DATABASE

- *La seconda fase, invece, permette la definizione di modelli formali per supportare il lavoro del progettista.*
- Abbiamo due tipologie di modelli che possiamo utilizzare
 - ER (Entity Relationship):
 - Il diagramma ER è definito proprio per la modellazione di basi di dati relazionali
 - I costrutti e le forme utilizzate sono adattate alla modellazione di un database
 - UML (Unified Modelling Language)
 - I diagrammi UML sono utilizzati per molte attività di modellazione nel ciclo di vita di un progetto software
 - Sequence Diagram, Activity Diagram, StateChart Diagram, Class Diagram...
 - Ma è anche esso un linguaggio di modellazione (UNIFICATO) e può essere facilmente adattato anche per la descrizione di una base di dati
 - All'interno del corso, utilizzeremo questo.
 - In particolare, utilizzeremo la notazione dei Class Diagram per la modellazione di una base di dati.



MODELLO ER

- Il modello Entità-Relazione (ER) è un diffusissimo data model di alto livello, estesamente utilizzato per definire lo schema concettuale di un database.
- È stato concepito per essere più vicino ai concetti “*umani*”, e quindi facilmente comprensibile anche ad utenti non tecnici.
- Il modello ER ha avuto una grandissima diffusione principalmente per i formalismi grafici semplici e chiari che incorpora.
- Il modello ER descrive i dati con tre concetti fondamentali:
 - Entità
 - Attributi
 - Relazioni



MODELLAZIONE DEI DATI IN UML

- UML (**Unified Modeling Language**) è un linguaggio grafico per la modellazione di applicazioni software basate sulla programmazione orientata agli oggetti.
 - Permette di rappresentare (attraverso una serie di diagrammi) tutti gli aspetti di un applicazione software: dati, operazioni, processi e architetture.
- UML può essere utilizzato in alternativa al modello ER:
 - I *diagrammi delle classi* sono usati per descrivere le classi di oggetti di interesse e le relazioni che intercorrono tra di esse.
- *Cambia la rappresentazione diagrammatica ma non l'approccio alla progettazione.*



UML VS. ER

- UML
 - modellazione di un'applicazione software
 - aspetti strutturali e comportamentali (dati, operazioni, processi e architetture)
 - formalismo ricco
 - diagramma delle classi, degli attori, di sequenza, di comunicazione, degli stati, ...
- ER/EER
 - modellazione di una base di dati
 - aspetti strutturali di un'applicazione
 - costrutti funzionali alla modellazione di basi di dati



UML VS. ER

- Principali differenze di UML rispetto ad ER
 - assenza di notazione standard per definire gli identificatori
 - possibilità di aggiungere note per commentare i diagrammi
 - possibilità di indicare il verso di navigazione di una associazione (non rilevante nella progettazione di una base di dati)
- Formalismi diversi
- Il diagramma delle classi di un'applicazione è diverso dallo schema ER della base di dati
- Il diagramma delle classi, anche se progettato per uso diverso, può essere adattato per la descrizione del progetto concettuale di una base di dati



UML: LE ORIGINI

- Proposto a metà degli anni '90 quale **formalismo unificante** per la modellazione o.o. di applicazioni software, è stato standardizzato sotto l'egida dell'Object Management Group (OMG).
- UML offre una molteplicità di diagrammi, corredati da una descrizione testuale della loro semantica: **molteplicità di viste** della medesima applicazione.



UML: IL MODELLO DELL'APPLICAZIONE

- L'insieme dei diagrammi definisce il **modello dell'applicazione** (controparte dello schema ER):
 - UML è un metamodello per la descrizione di modelli di applicazioni software.
- Nella sua versione corrente UML prevede un certo numero di **diagrammi fondamentali**:
 - diagramma delle classi, degli oggetti, dei casi d'uso, di sequenza, di comunicazione, delle attività, degli stati, dei componenti e di distribuzione dei componenti.



UML: I DIAGRAMMI PRINCIPALI (1)

- Diagramma delle classi:
 - descrive le caratteristiche statiche e dinamiche delle componenti (**classi**) e delle loro relazioni (**associazioni**).
- Diagramma degli oggetti:
 - rappresentazione delle possibili istanze delle classi (**oggetti**) e dei loro collegamenti.
- Diagramma dei casi d'uso:
 - modalità di utilizzo del sistema da parte di persone/altri sistemi (**attori**) e interazioni tra sistema e attori.



UML: I DIAGRAMMI PRINCIPALI (2)

- Diagramma di sequenza:
 - ordinamento temporale di messaggi (**invocazione di metodi**) scambiati tra i diversi oggetti dell'applicazione.
- Diagramma delle attività:
 - comportamento dinamico di un processo dell'applicazione in termini dei flussi di **attività** da svolgere.
- Diagramma degli stati:
 - descrive il ciclo di vita di un oggetto dell'applicazione attraverso gli **stati** che può assumere.

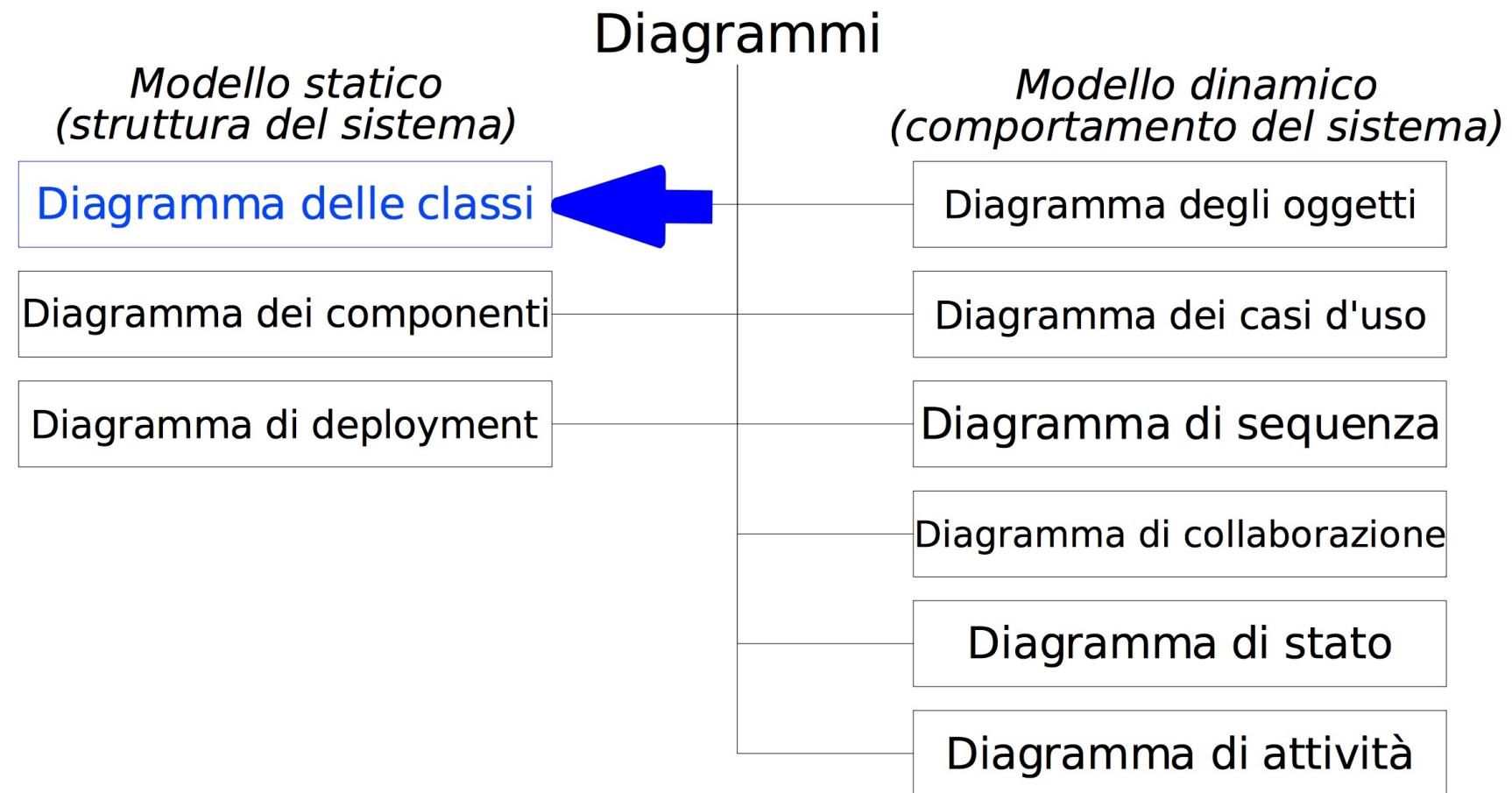


UML: I DIAGRAMMI PRINCIPALI (3)

- Diagramma di collaborazione (comunicazione):
 - descrive lo scambio di **messaggi** tra gli oggetti, ma utilizzando notazione e prospettiva diverse.
- Diagramma dei componenti:
 - descrive l'organizzazione delle **componenti** fisiche del sistema (file, moduli, ..) e le loro dipendenze.
- Diagramma di distribuzione dei componenti (deployment):
 - descrivere la **dislocazione** dei nodi hardware del sistema e le loro associazioni.



UML: I DIAGRAMMI PRINCIPALI (4)



MODELLAZIONE STRUTTURALE

- Modellazione strutturale:
 - Rappresenta una vista di un sistema software che pone l'accento sulla struttura degli oggetti (classi di appartenenza, relazioni, attributi, operazioni)
- Il **diagramma delle classi** descrive il tipo degli oggetti che compongono il sistema e le relazioni statiche esistenti tra loro



ENTITÀ E ATTRIBUTI (UML E ER)

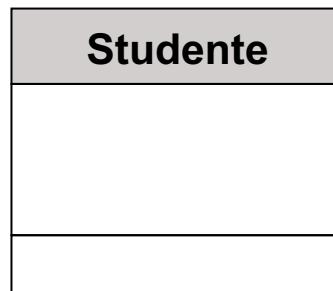


ENTITÀ

- Le entità corrispondono a classi di oggetti del mondo reale (fatti, persone, ...) che hanno proprietà omogenee, comuni ed esistenza “autonoma” ai fini dell'applicazione di interesse.
- Un'entità può essere un oggetto fisico (casa, impiegato, ...) o un oggetto concettuale (un lavoro, una società, ...).



Rappresentazione ER
dell'entità "*Studente*"



Rappresentazione UML
Della classe "*Studente*"



RAPPRESENTAZIONE DI DATI CON I DIAGRAMMI DELLE CLASSI

- **Classi:** Sono le componenti principali dei diagrammi delle classi.
 - Corrispondono alle entità del modello ER.
 - È possibile aggiungere i metodi (i.e., le operazioni ammissibili sugli oggetti della classe).
 - I dati vengono descritti *insieme* alle operazioni da svolgere su di essi.

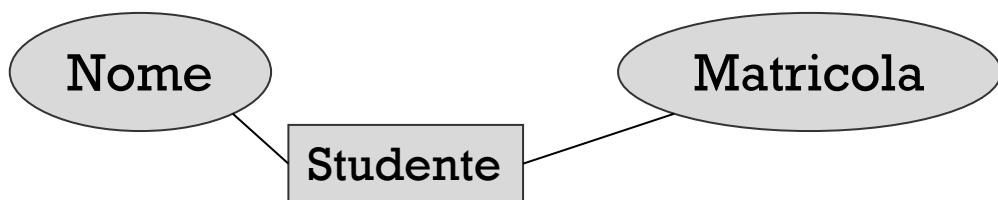
Impiegato
Codice
Cognome
Stipendio
Età
<i>SetStipendio()</i>

Progetto
Nome
Budget
Data consegna

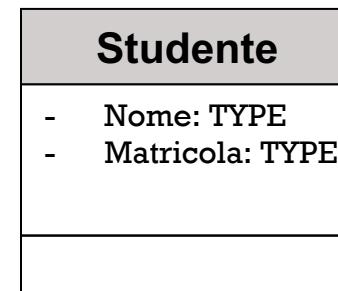


ATTRIBUTI

- Ogni entità ha delle proprietà dette attributi.
 - *Es:* l'entità *Impiegato* ha attributi **nome**, **età**, **indirizzo**, **salario**, **telefono**, ...
- Ogni entità è caratterizzata da un valore per i suoi attributi.



Rappresentazione ER
dell'entità "*Studente*" con gli
attributi "*Nome*" e "*Matricola*"

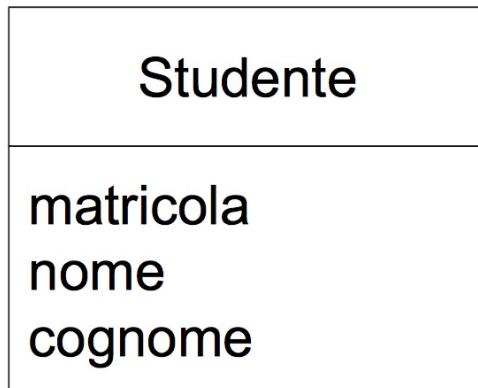


Rappresentazior **UML**
della classe "*Studente*" con gli
attributi "*Nome*" e "*Matricola*"



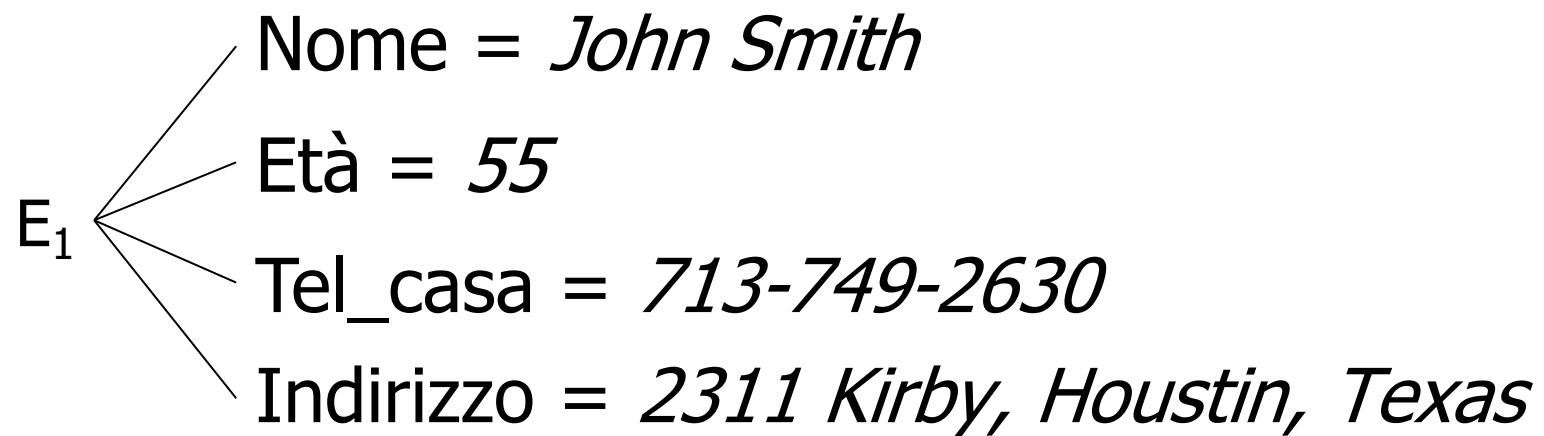
ATTRIBUTI IN ER E UML

- ER:
 - un attributo descrive una proprietà elementare di un'entità o di una relazione.
- UML:
 - un attributo rappresenta una proprietà elementare degli oggetti di una classe.



ENTITÀ E ATTRIBUTI: *ESEMPIO*

- ***Esempio:*** entità *Impiegato*



TIPO DI ENTITÀ

- Entità con gli stessi attributi di base sono raggruppati in un tipo di entità.
- Esempio:** Tutte le persone che lavorano per un dipartimento possono essere definite con l'entità *Impiegato*.

Nome del Tipo di Entità:
(Schema o Intenzione)

Impiegato

Nome, Età, Stipendio

e_1	■	
(John Smith, 55, 80k)		
e_2	■	
(Fred Brown, 40, 30k)		
e_3	■	
(Judy Clark, 25, 20k)		
	:	

Insieme di Entità
(Estensione)

Un tipo di entità descrive lo schema (o intenzione) per un insieme di entità

Le entità individuali di un particolare tipo di entità sono raggruppate in una collezione o insieme detta estensione del tipo di entità.



TIPI DI ATTRIBUTI

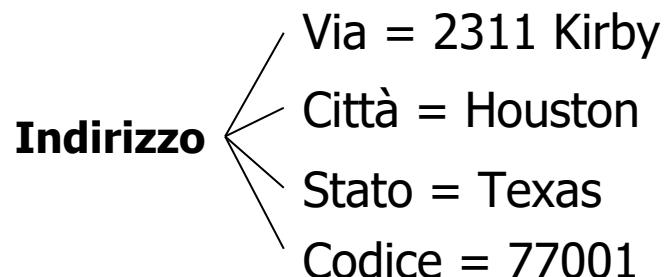
- Gli attributi possono essere di diversi tipi

Divisibile?	Più valori?	Calcolabile?
Semplice	Single-valued	Memorizzato
Composto	Multivalued	Derivato



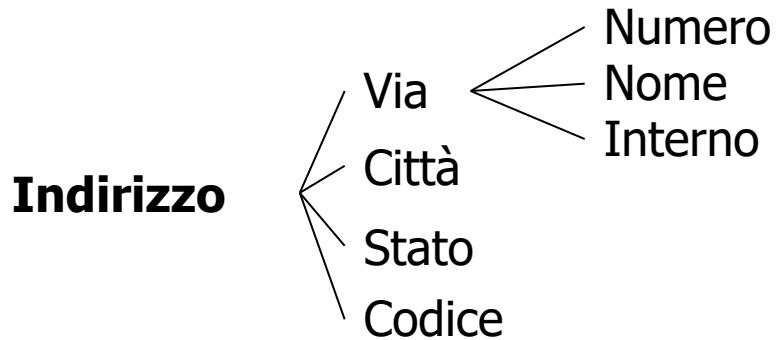
ATTRIBUTI SEMPLICI E COMPOSTI

- *Attributi semplici*: ogni entità ha un valore singolo (atomico) per tale attributo.
- *Attributi composti*: possono essere divisi in sottoparti, che rappresentano informazioni di base con loro significati indipendenti.
 - **Esempio:** l'attributo **Indirizzo**:



ATTRIBUTI SEMPLICI E COMPOSTI (2)

- Gli attributi composti possono formare una gerarchia:



- L'utilizzo di un attributo semplice o di uno composto dipende dalla necessità o meno di trattare separatamente le sottoparti.



SINGLE-VALUED E MULTIVALUED

■ *Attributi single-valued:*

- hanno un solo valore per ciascuna entità.
- **Esempio:** l'età di un impiegato.

■ *Attributi multivalued:*

- può avere un insieme di valori per la stessa entità.
- **Esempio:** l'attributo **Colore** per un'entità auto. Un'auto può avere più colori.
- Può essere determinato un limite inferiore e un limite superiore al numero di valori per un'entità.



MEMORIZZATI E DERIVATI

- *Attributi derivati:* alcuni attributi possono essere in relazione tra loro.
 - **Esempio:** l'età e la data di nascita:

■ Età	attributo derivato
■ Data_nascita	attributo memorizzato
 - Alcuni valori di attributi possono essere derivati da entità correlate.
Esempio:
Numero_di_Impiegati di una entità “*dipartimento*” può essere derivato contando il numero di impiegati relati al (che lavorano per) dipartimento.
- Alcuni attributi possono avere valore **null** col significato di “*non noto*” o “*mancante*” o “*non applicabile*”.



PARTICOLARITÀ DEGLI ATTRIBUTI IN UML

- Non è possibile definire attributi composti.
- È possibile associare agli attributi i rispettivi domini
 - Interi, reali, stringhe, etc.

Impiegato
String Codice
String Cognome
Float Stipendio
int Età
<i>SetStipendio()</i>



ATTRIBUTI CHIAVE DI UN TIPO DI ENTITÀ

- Un importante vincolo sulle entità di un tipo di entità è la chiave o vincolo di unicità.
 - L'attributo chiave di un tipo di entità è un attributo che deve avere un valore univoco per cui ogni entità.
 - **Esempio:** Il codice fiscale di una persona.
- Talvolta più attributi insieme formano una chiave:
 - In tal caso tali attributi possono essere raggruppati in un attributo composto che diventa chiave.
- Alcuni tipi di entità possono avere più di un attributo chiave.
- In notazione ER l'attributo chiave è rappresentato **sottolineato** nell'ovale.



ESEMPIO: ENTITÀ AUTO

AUTO

Targa(Numero,Provincia), Telaio, Marca, Modello, Anno_imm, {Colore}

Car 1 .

((ASC 123, TEXAS), tk629, Ford Mustang, convertible, 1989, {red, black})

Car 2 .

((ABO 123, NEW YORK), WPS872, N~san sontra 2~cor, 1992, {blue})

Car3 .

((VYS 720, TEXAS), TD729, Chrysler LeBaron, 4dcor, 1993, {white, blue})

Gli attributi multivalued sono mostrati tra parentesi {}.
Le componenti di un attributo composto sono mostrate tra parentesi ().



DOMINIO DI UN ATTRIBUTO

- Ciascun attributo semplice è associato a un dominio o insieme di valori che rappresenta l'insieme dei valori che l'attributo può assumere.
- **Esempio:** l'età dell'impiegato può variare nel dominio (16, 70).
- *Matematicamente:*
 - Un attributo semplice A , del tipo di entità E , avente dominio V , è una funzione:
 $A: E \rightarrow P(V)$ $P(V)$ insieme potenza dei sottoinsiemi di V :
Un valore **null** è rappresentato con \emptyset .
 - $A(e)$ il valore dell'attributo A per l'entità e ,
 - $A(e) = \text{singleton}$ per attributi single-valued.



DOMINIO DI UN ATTRIBUTO (2)

- Per un attributo composto A, del tipo di entità E, il dominio V è il prodotto cartesiano

$$V = P(V_1) \times P(V_2) \times \dots \times P(V_n)$$

dove V_i è il dominio dell'attributo semplice i -mo di A,

- **Esempio:** l'indirizzo telefonico di una persona con più residenze, dove ogni residenza può avere più telefoni, è specificato come segue:

{ Indirizzo ({Telefono (Prefisso, Numero)}, Indirizzo (Via (Numero, Nome, Interno), Città, Stato, Codice) }



ESEMPIO: IL DATABASE COMPANY



REQUISITI PER IL DATABASE COMPANY

- La compagnia è organizzata in DIPARTIMENTI. Ogni Dipartimento ha un nome, un numero ed un impiegato che lo gestisce. Bisogna tener traccia della data di insediamento del manager. Un dipartimento può avere più locazioni.
- Ogni dipartimento controlla una serie di PROGETTI. Ogni progetto ha un nome, un numero ed una singola locazione.
- Per IMPIEGATO si tiene traccia di: nome, SSN, indirizzo, salario, sesso e data di nascita. Ogni impiegato lavora per un dipartimento e può lavorare su più progetti. Teniamo traccia del numero di ore settimanali che un impiegato spende su un progetto e del supervisore di ogni impiegato.
- Ogni impiegato ha una serie di PERSONE A CARICO. Per ogni persona a carico, registriamo: nome, sesso, data di nascita e parentela con l'impiegato.



DISEGNO CONCETTUALE DEL DATABASE COMPANY

- Descriviamo i tipi di entità per il database COMPANY.
- In accordo ai requirement possiamo identificare quattro tipi di entità:

1.DIPARTIMENTO

Nome, Numero, {Sedi}, Manager, Datains_manager

Nome e Numero sono entrambi attributi chiave.

2.PROGETTO

Nome, Numero, Luogo, Dip_controllo

Nome e Numero sono entrambi attributi chiave.



DISEGNO CONCETTUALE DEL DATABASE COMPANY (2)

3. IMPIEGATO

Nome, SSN, Sesso, Indirizzo, Stipendio, DataNascita, Dipartimento, Supervisore

SSN è un attributo chiave.

Nome e Indirizzo sono attributi composti (occorrerebbe verificare con l'utente se ha bisogno di riferire ai componenti individuali).

4. PERS_A_CARICO

Sesso, DataNascita, Impiegato, Nome_pers_carico, Parentela



DISEGNO CONCETTUALE DEL DATABASE COMPANY (3)

- Dobbiamo però rappresentare:
 - Il fatto che un impiegato può lavorare su più progetti.
 - Il numero di ore settimanali di un impiegato su ciascun progetto.
- Si può aggiungere un attributo a IMPIEGATO “*Lavora_su*” composto di due componenti semplici (**Progetto, Ore**):

IMPIEGATO

Nome (FName, Minit, LName), SSN, Sesso, Indirizzo, Stipendio, DataNascita, Dipartimento, Supervisore, {Lavora_su(Progetto, Ore)}

- In alternativa, le stesse informazioni si potrebbero mantenere nel tipo di entità PROGETTO con un attributo composto:
 - **Addetti (Impiegato, Ore)**

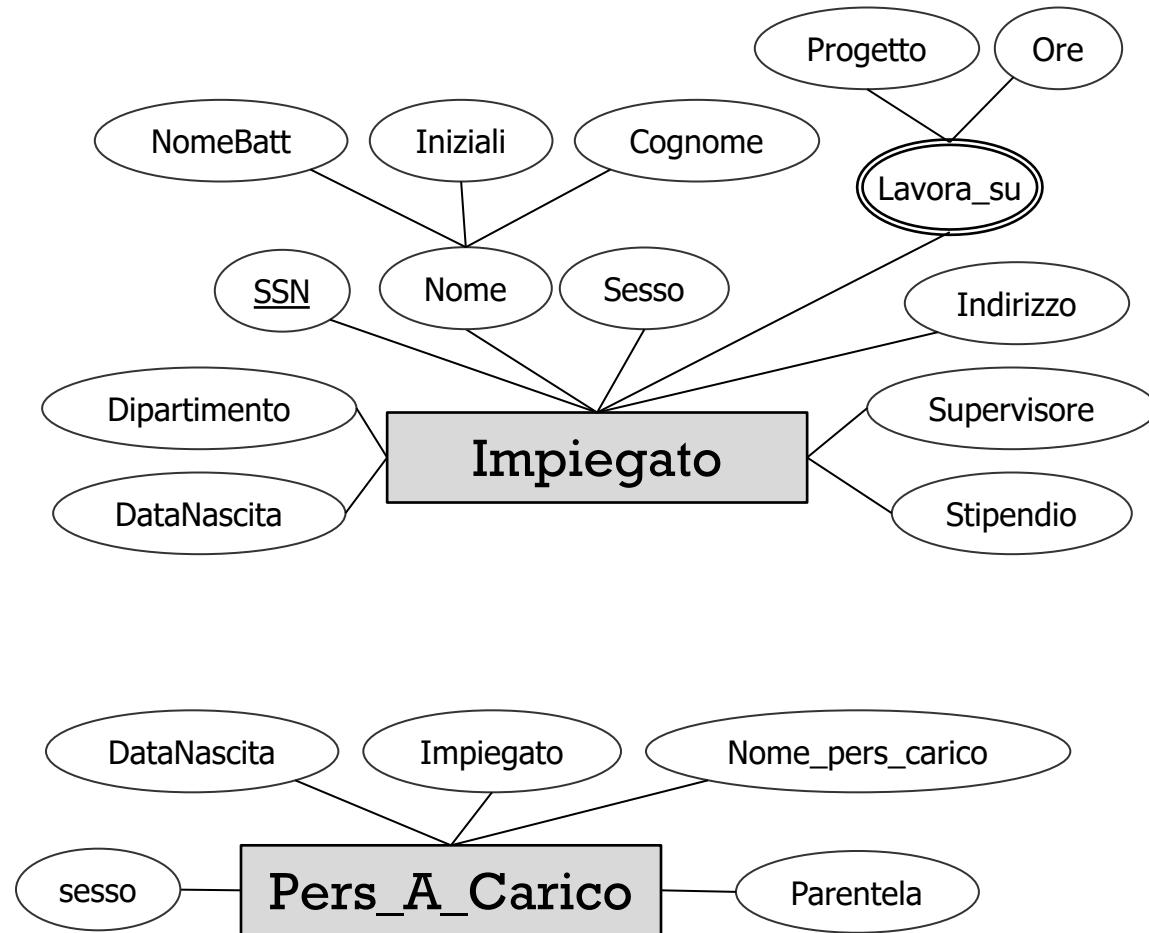
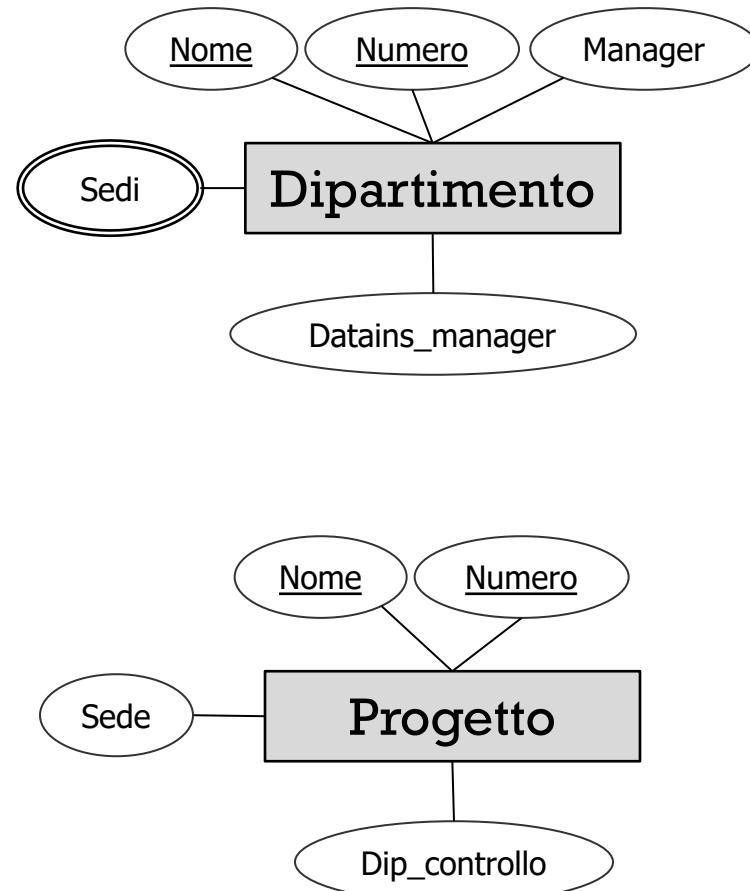


DISEGNO CONCETTUALE DEL DATABASE COMPANY (4)

- **Esistono varie relazioni implicite:**
 - L'attributo Manager di DIPARTIMENTO si riferisce a un impiegato che gestisce il dipartimento;
 - L'attributo Dip_controllo di PROGETTO si riferisce al dipartimento che controlla il progetto;
 - L'attributo Dipartimento di IMPIEGATO si riferisce al dipartimento per cui lavora l'impiegato;
 - ...
- Nel disegno iniziale queste associazioni tra entità sono rappresentabili come attributi, ma durante il processo di raffinamento nel modello ER questi riferimenti dovrebbero essere rappresentati come relazioni.



PROGETTAZIONE ER PRELIMINARE PER IL DB COMPANY



PROGETTAZIONE UML (CLASS DIAGRAM) PRELIMINARE PER IL DB COMPANY

Dipartimento
Nome
Numero
Manager
Datains_Manager
Sedi

Progetto
Nome
Numero
Sede
Dip_controllo

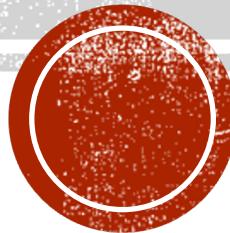
Impiegato
SSN
Dipartimento
DataNascita
{Nome}
Sesso
Indirizzo
Supervisore
Stipendio
{Lavora_Su}

Pers_a_carico
Nome
Parentela
Impiegato
Sesso
DataNascita





LE RELAZIONI (ER)



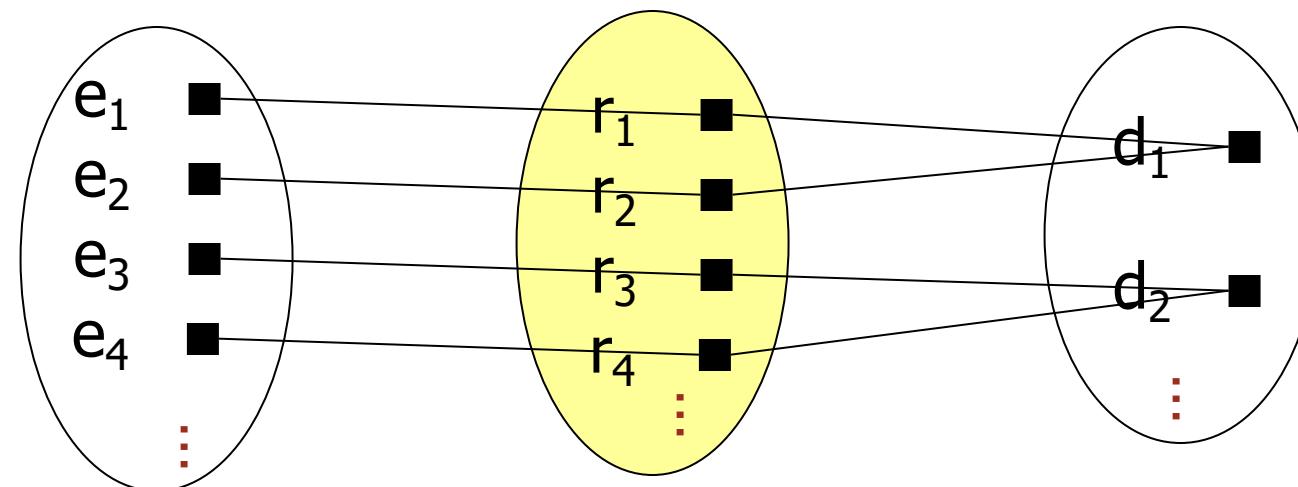
TIPI E Istanze di Relazioni

- Le relazioni corrispondono a legami logici tra entità, significativi ai fini dell'applicazione di interesse.
- Un tipo di relazione è un'associazione tra n tipi di entità E_1, E_2, \dots, E_n .
- Le occorrenze o istanze di relazione associano n entità dei tipi di relazione richiesti.
- Ogni tipo di entità è detto partecipare al tipo di relazione.
- Il grado di un tipo di relazione è il numero di entità che vi partecipano. Se il grado è 2, la relazione è detta binaria.



TIPI E Istanze DI RELAZIONI (2)

- **Esempio:** vogliamo rappresentare il fatto che ogni impiegato e_i lavora per un dipartimento d_j .
- Definiamo il tipo di relazione LAVORA_PER tra i due tipi di entità IMPIEGATO e DIPARTIMENTO: ogni relazione r_i associa una entità IMPIEGATO e_i con una entità DIPARTIMENTO d_j .



Impiegato:
Tipo di Entità

Lavora_Per:
Relazione

Dipartimento:
Tipo di Entità



TIPI E ISTANZE DI RELAZIONI (3)

Matematicamente:

R è un insieme di istanze di relazione r_i dove ogni r_i associa n entità (e_1, e_2, \dots, e_n), e ciascuna entità e_j in r_i è un membro del tipo di entità E_j , con $1 \leq j \leq n$.

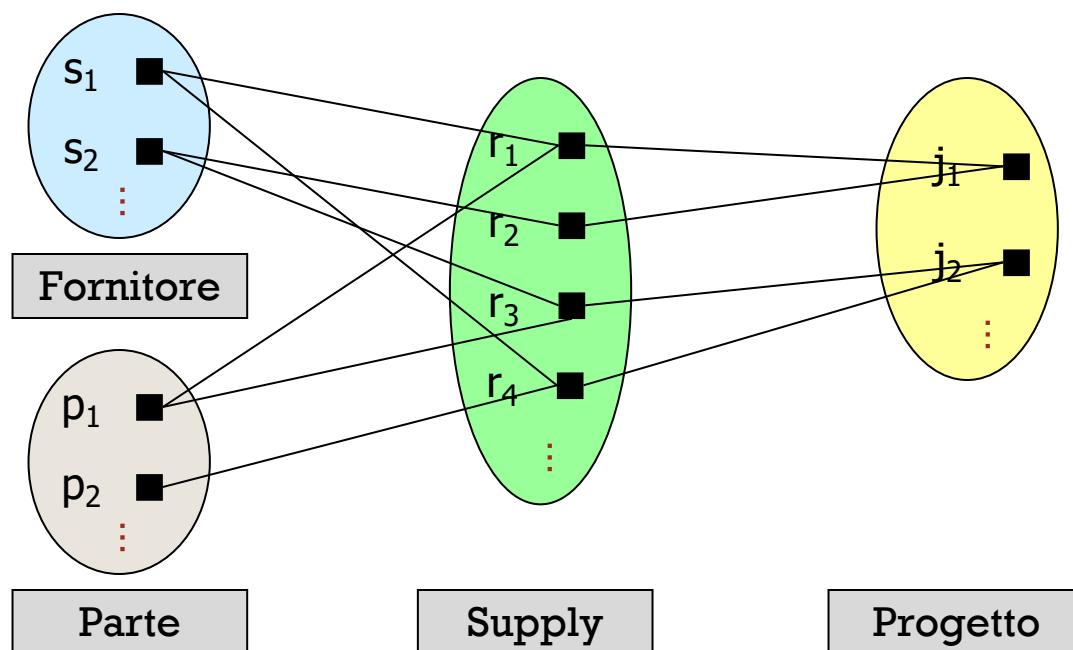
Quindi un tipo di relazione è una funzione matematica su E_1, E_2, \dots, E_n o alternativamente può essere definita come un sottoinsieme del prodotto cartesiano $E_1 \times E_2 \times \dots \times E_n$.

Ciascun E_j è detto partecipare al tipo di relazione R e analogamente ogni entità individuale e_j è detta partecipare all'istanza di relazione $r_i = (e_1, e_2, \dots, e_n)$.



GRADO DI UN TIPO DI RELAZIONE

- Il grado di un tipo di relazione è il numero di tipi di entità partecipanti.
- Esempio:**
 - LAVORA_PER è di grado 2 (binaria).
 - La relazione SUPPLY è un tipo di relazione ternaria, dove ogni istanza di relazione r_i associa tre entità, un fornitore s , una parte p e un progetto j ogni volta che s fornisce la parte p al progetto j .



Le relazioni possono essere di qualsiasi grado ma le più ricorrenti sono quelle binarie.



RAPPORTO DI CARDINALITÀ

- Deve essere indicato per ciascun tipo di entità che partecipa ad una relazione, e permette di specificare il numero minimo e massimo di istanze di relazione a cui le occorrenze delle entità coinvolte possono partecipare.



- Rappresentazione ER della relazione “*Impiegato lavora per Dipartimento*”, con rapporto di cardinalità N:1 – N impiegati lavorano per un dipartimento:
 - MAX:** Ogni dipartimento può avere numerosi impiegati, e ciascun impiegato lavora per un solo dipartimento.
 - MIN:** Un dipartimento potrebbe non avere impiegati, mentre un impiegato deve sempre essere assegnato ad un dipartimento.



ESEMPIO: RELAZIONE LAVORA_PER

Esempio: Impiegato dipartimento

(rossi, ricerca)

(bianchi, ricerca)

(neri, amministrazione)

(verdi, ricerca)

- Rossi è presente una volta nella relazione.
- Ricerca è presente 3 volte.



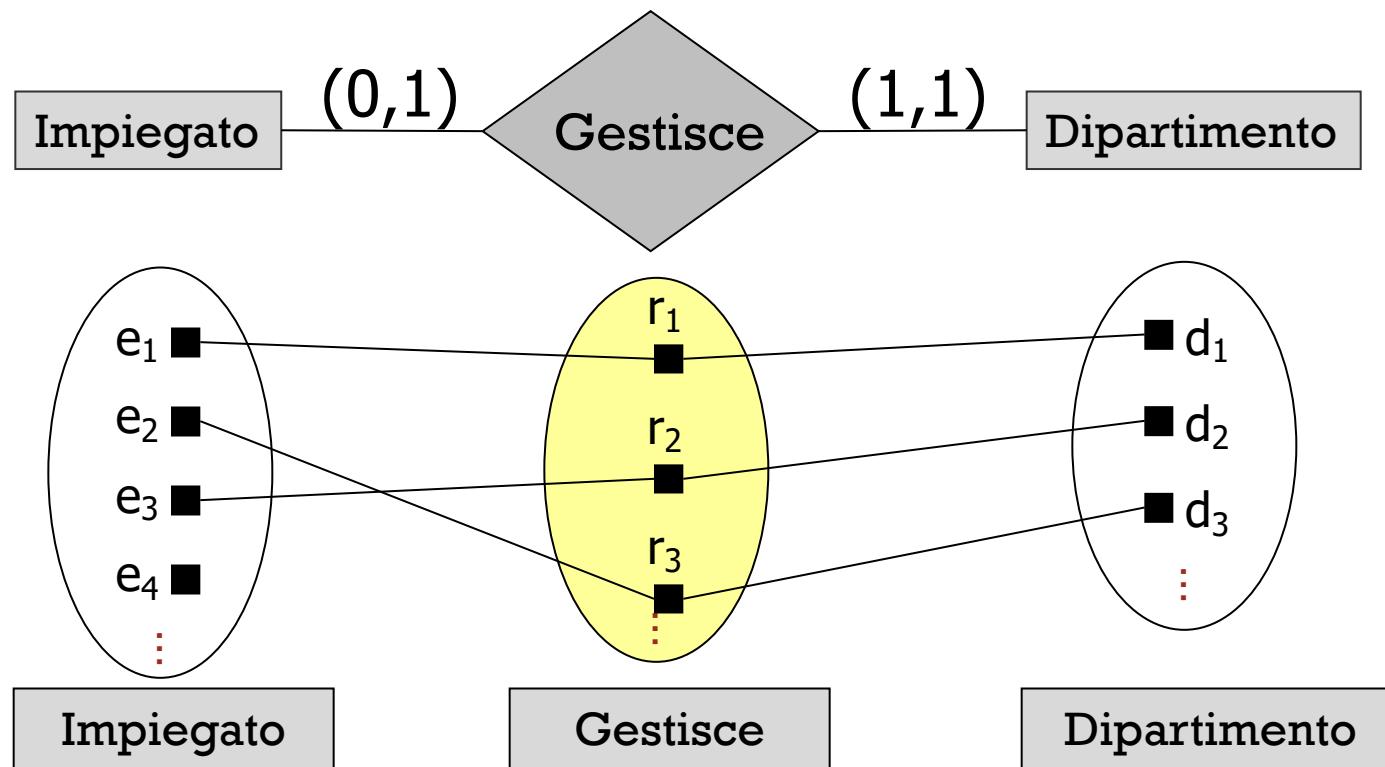
RAPPORTO DI CARDINALITÀ (2)

- È possibile assegnare un qualunque intero non negativo a un rapporto di cardinalità, con l'ovvio vincolo che la cardinalità minima deve essere minore o uguale alla cardinalità massima.
- Nella maggior parte dei casi si utilizzano solo tre valori: 0, 1 e N:
 - Il valore 0 per la cardinalità minima indica una partecipazione opzionale del tipo di entità alla relazione.
 - Il valore 1 per la cardinalità minima indica una partecipazione obbligatoria del tipo di entità alla relazione.
- La cardinalità minima può eventualmente essere omessa, quella massima deve essere sempre specificata.



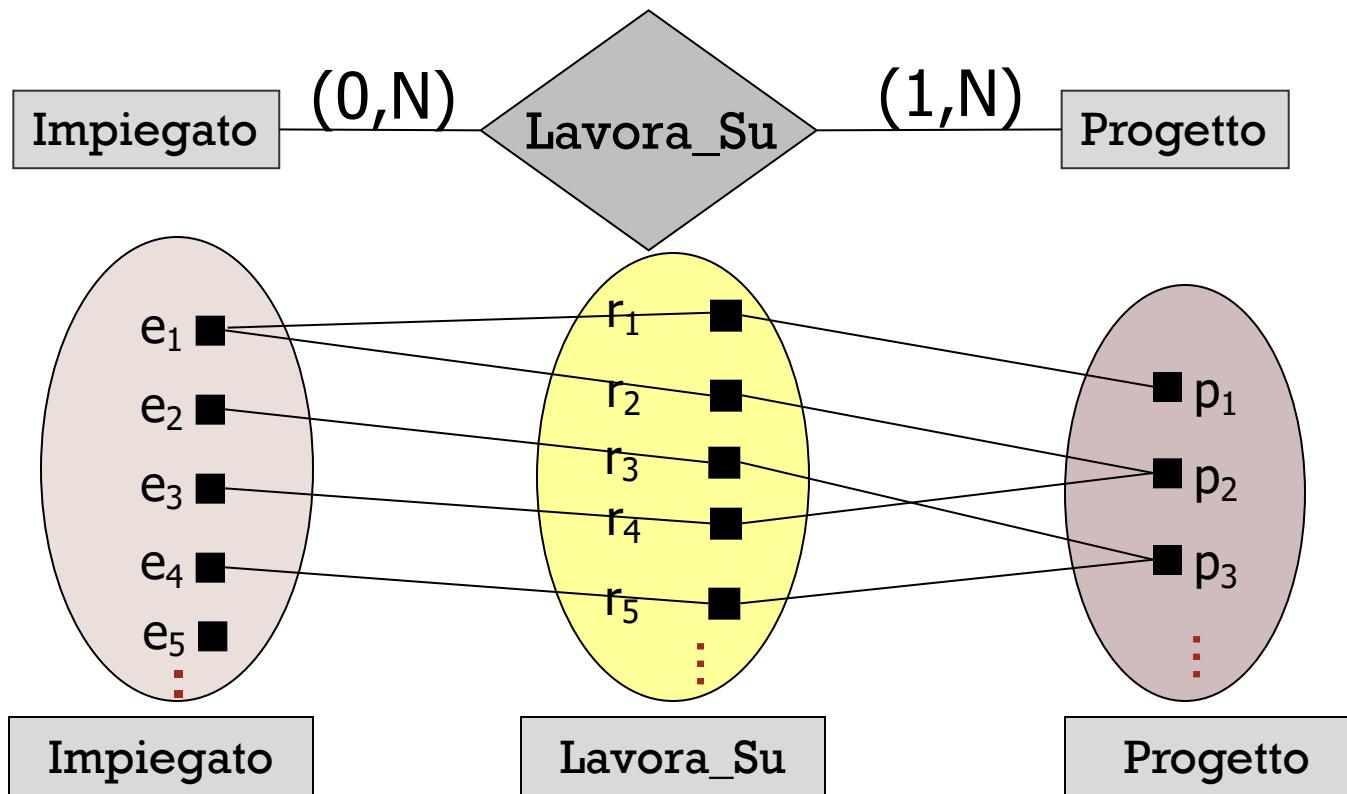
RAPPORTO DI CARDINALITÀ: ESEMPIO

- La relazione binaria GESTISCE tra IMPIEGATO e DIPARTIMENTO è di rapporto di cardinalità 1:1 (un impiegato può gestire al più un dipartimento, ed un dipartimento deve sempre avere un solo manager).



RAPPORTO DI CARDINALITÀ: ESEMPIO

- La relazione binaria LAVORA_SU tra IMPIEGATO e PROGETTO è di rapporto di cardinalità M:N, (poiché un impiegato può lavorare su più progetti, e più impiegati possono lavorare sullo stesso progetto).

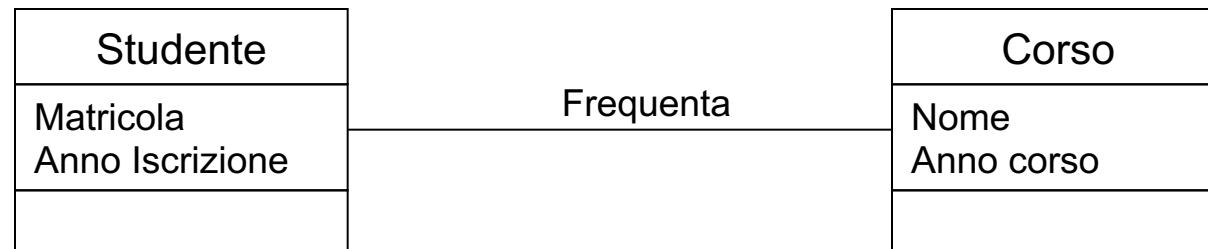
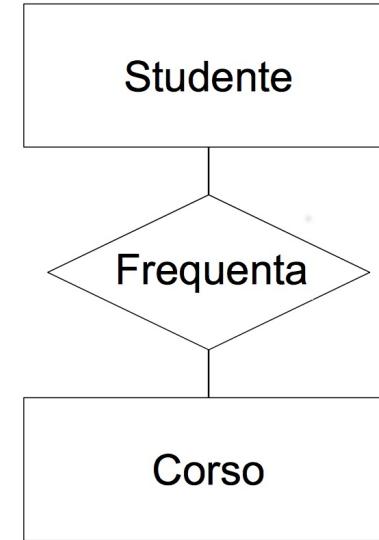


LE ASSOCIAZIONI (UML)



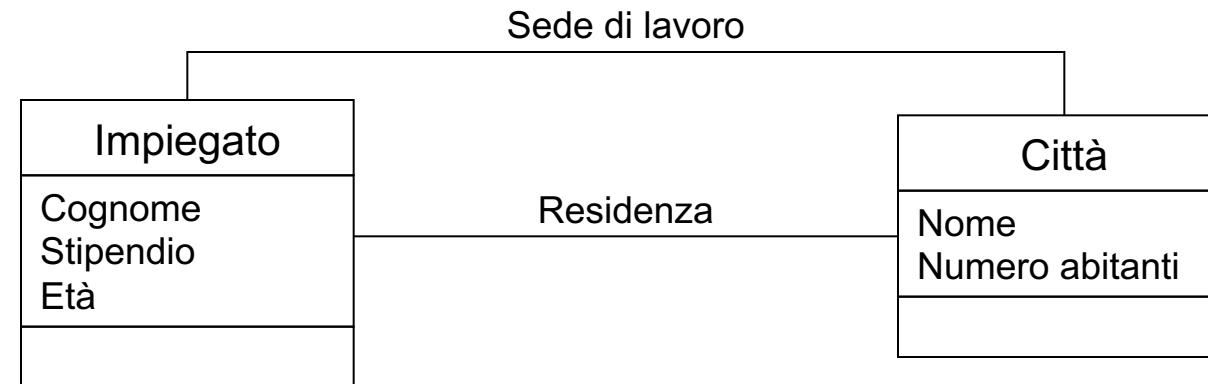
LE ASSOCIAZIONI

- **Associazioni:** Corrispondono alle relazioni del modello ER.
 - Le associazioni binarie si rappresentano con semplici linee che congiungono le classi coinvolte.
 - Il nome della relazione viene posto sulla linea.
 - *Non è obbligatorio:* infatti, in UML possono esistere relazioni senza nome.



LE ASSOCIAZIONI (2)

- Si possono definire più associazioni tra le medesime classi.

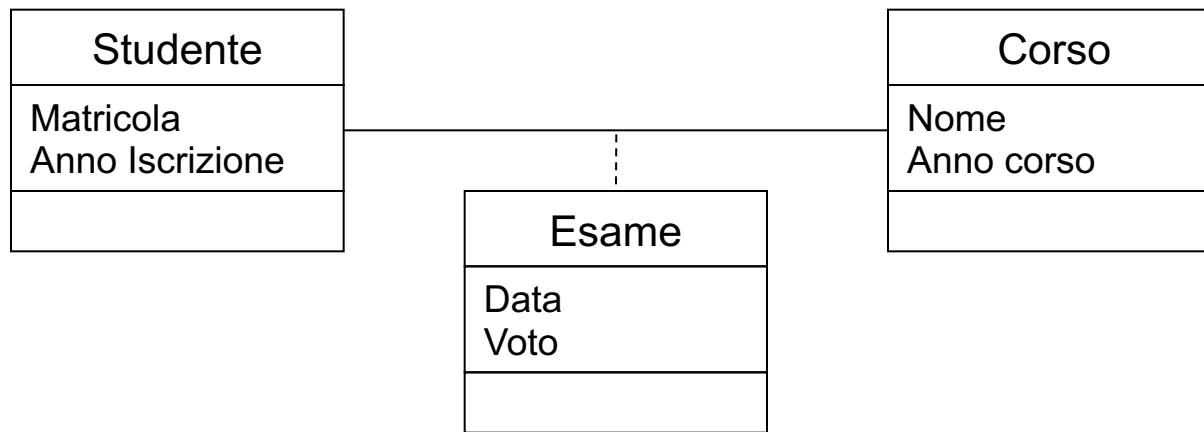


- È possibile associare *ruoli* alle classi coinvolte nelle associazioni.
- Non è possibile assegnare attributi alle associazioni.



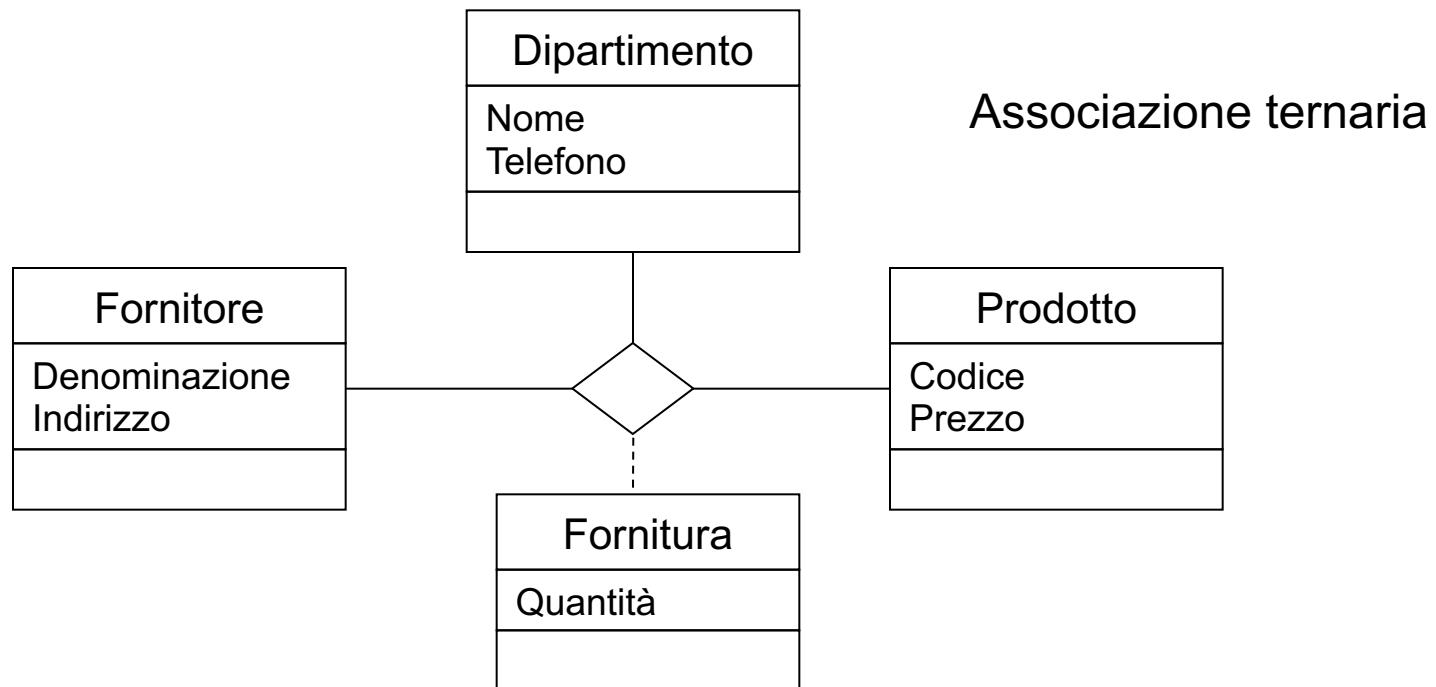
LE ASSOCIAZIONI (3)

- Per assegnare attributi ad una associazione si fa uso delle *classi di associazione* per descrivere le proprietà di una associazione.
 - Queste classi vengono collegate all'associazione da descrivere mediante una linea tratteggiata.



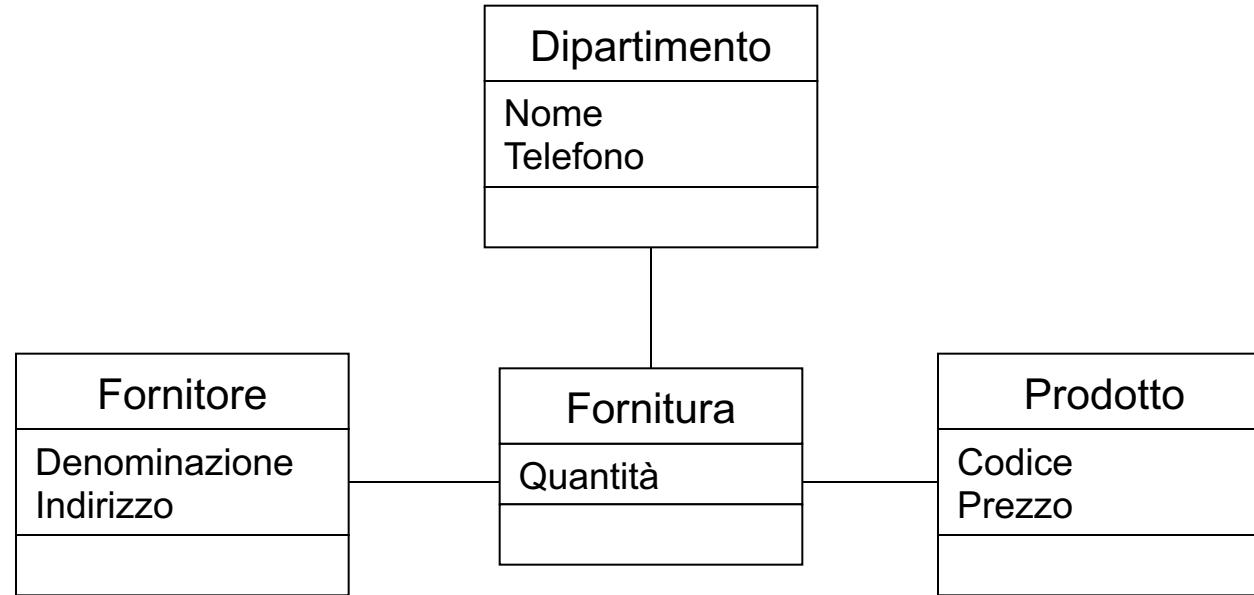
ASSOCIAZIONI N-ARIE

- L'associazione viene rappresentata da un *rombo* collegato con le classi che partecipano all'associazione.
 - Si può usare una *classe di associazione* per assegnare attributi all'associazione n-aria.



ASSOCIAZIONI N-ARIE (2)

- L'uso di associazioni n-arie è raro nel diagramma delle classi.
 - Si può applicare un processo di “*reificazione*”: ovvero trasformare l'associazione in una classe legata alle altre tramite associazioni binarie.



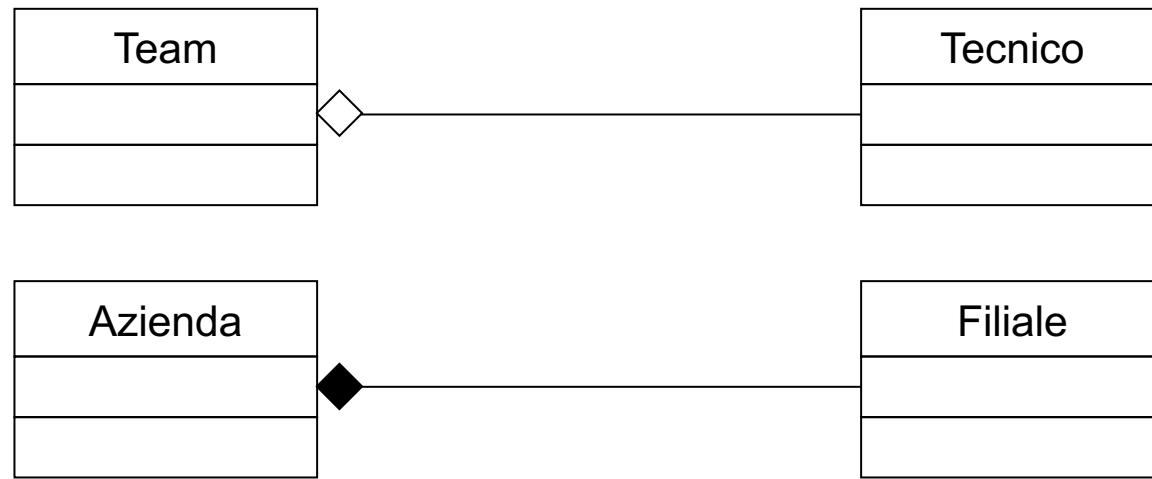
PROPRIETÀ DELLE ASSOCIAZIONI

- Con una freccia è possibile indicare un verso privilegiato di *navigabilità* di un'associazione.
 - Si possono specificare *aggregazioni* di concetti: relazioni tra un oggetto composito e uno o più concetti che ne costituiscono una sua parte.
 - Sono indicate da linee avente un rombo dal lato del concetto “aggregante”.



PROPRIETÀ DELLE ASSOCIAZIONI (2)

- Il rombo bianco indica che un oggetto della classe “parte” può esistere senza dover appartenere a un oggetto della classe “aggregante”. **(Aggregazione)**
- Il rombo nero indica bianco indica che un oggetto della classe “parte” **non** può esistere senza appartenere a un oggetto della classe “aggregante”. **(Associazione)**

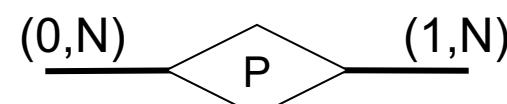
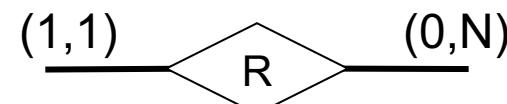


ASSOCIAZIONI CON MOLTEPLICITÀ

- La molteplicità di default è 1 (che denota la coppia di cardinalità 1..1).
 - Le cardinalità di default possono essere omesse.
- La cardinalità “*molti*” viene rappresentata dal simbolo * (dove si intende 0..*, ovvero (0, N) dello schema ER).
- La cardinalità di default per l’aggregazione è * (i.e., 0..*).



*Corrispondenza (invertita)
con le molteplicità dell'ER:*



IDENTIFICATORI (INTERNI)

- In UML non esiste una notazione per esprimere identificatori di classi.
- Si possono definire vincoli di integrità su associazioni e attributi specificandoli tra parentesi graffe (*vincolo utente*).
 - **Es.** $\{id\}$ indica che l'attributo è un identificatore.
 - **Es.** assegnare $\{id\}$ a più attributi indica un identificatore composto.

Automobile
Targa $\{id\}$
Modello
Colore

Persona
Data Nascita $\{id\}$
Cognome $\{id\}$
Nome $\{id\}$
Indirizzo



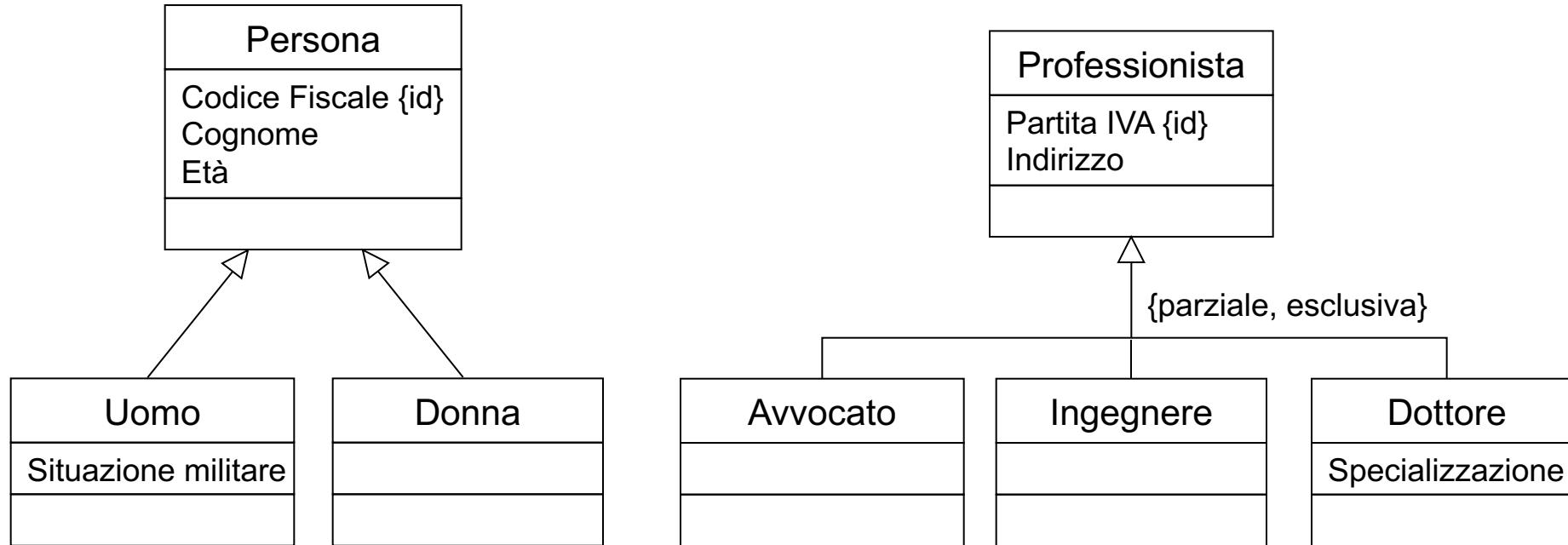
IDENTIFICATORE ESTERNO

- Per gli identificatori esterni si usa uno *stereotipo*.
 - Si usano per estendere i costrutti base dell'UML:
 - quando si vuole modellare un concetto che non può essere modellato con i costrutti di base.
 - Vengono indicati da un nome racchiuso tra i simboli << e >>.
 - **Es.** Lo stereotipo <<identificante>> insieme alla *Matricola* identifica univocamente uno *Studente* rispetto una particolare *Università*.



GENERALIZZAZIONI

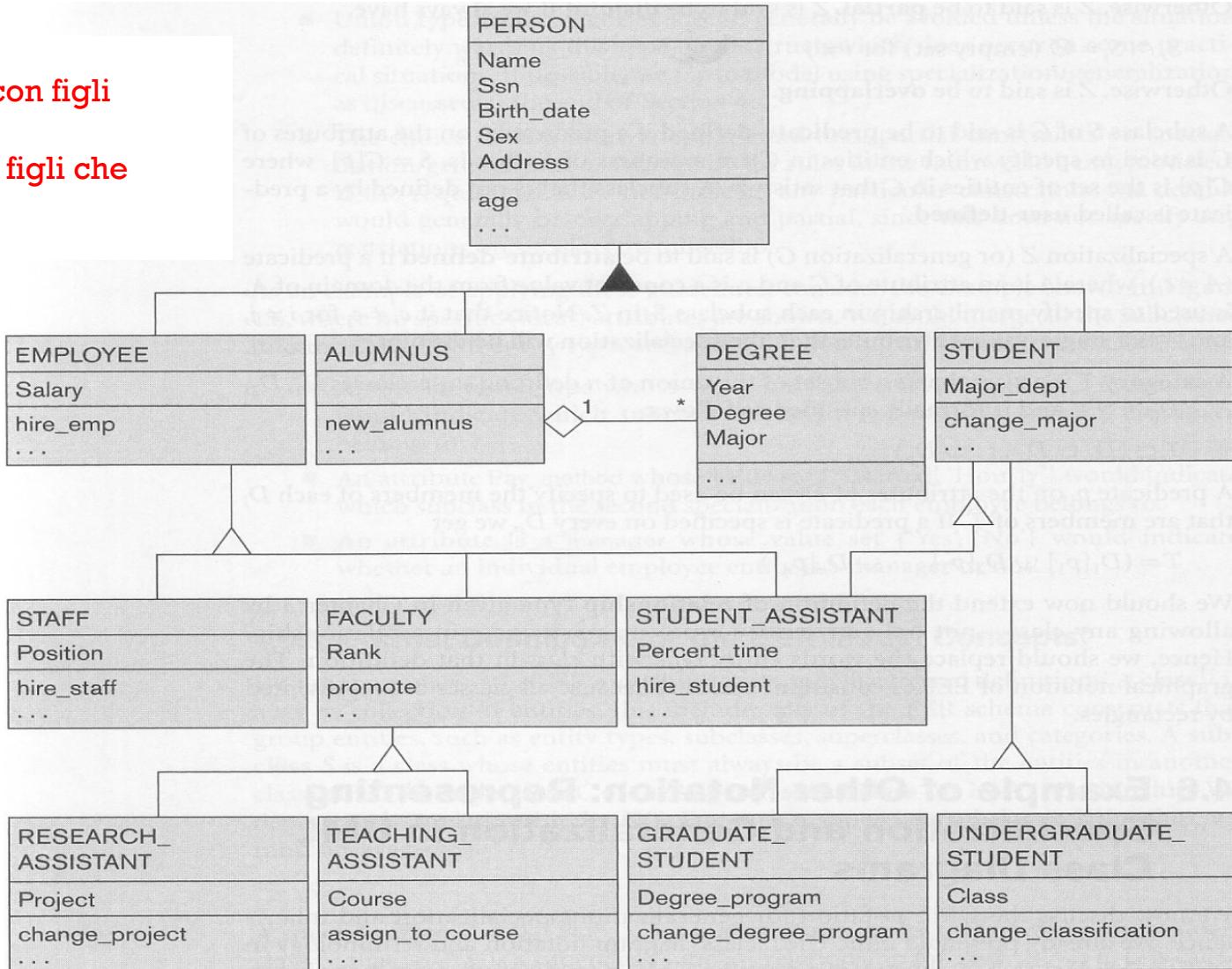
- Le generalizzazioni sono molto simili a quelle del modello ER.
- Eventuali proprietà possono essere rappresentate con vincoli racchiusi tra le parentesi { e }.



UNA GERARCHIA DI GENERALIZZAZIONI

Triangolo **bianco** = generalizzazione con figli disgiunti

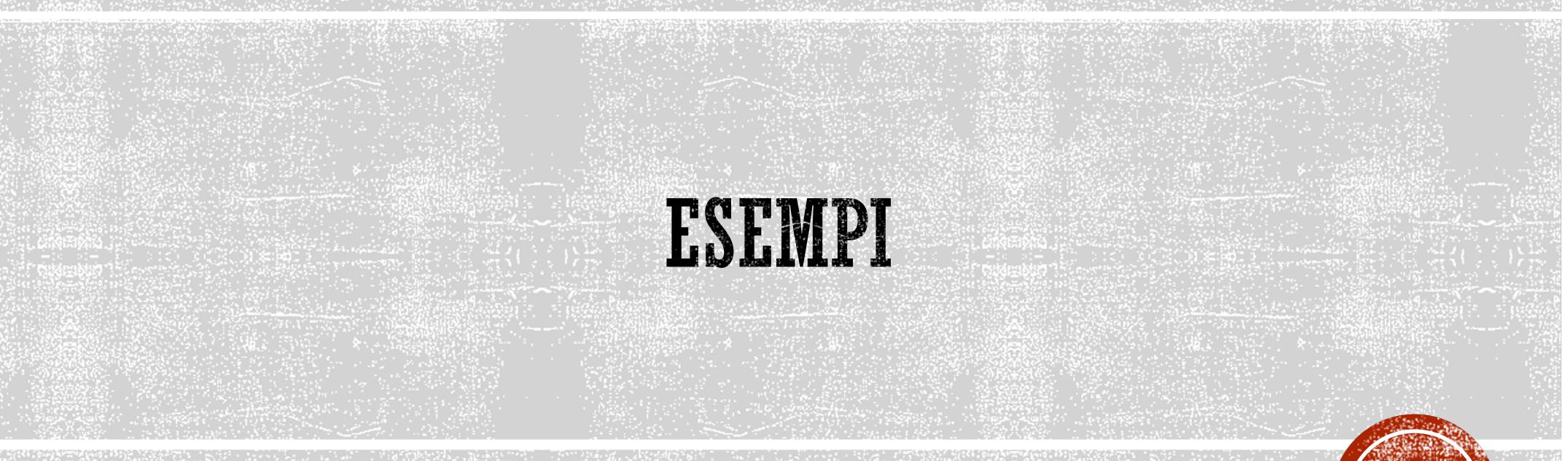
Triangolo **nero** = generalizzazioni con figli che possono sovrapporsi



LE NOTE

- Un diagramma delle classi può essere documentato con delle *note*.
 - Consistono in semplici commenti testuali.
 - Sono riportate sul diagramma in un rettangolo con l'angolo superiore destro ripiegato.
 - Una nota può essere associata ad un particolare elemento del diagramma attraverso una linea tratteggiata.





ESEMPI

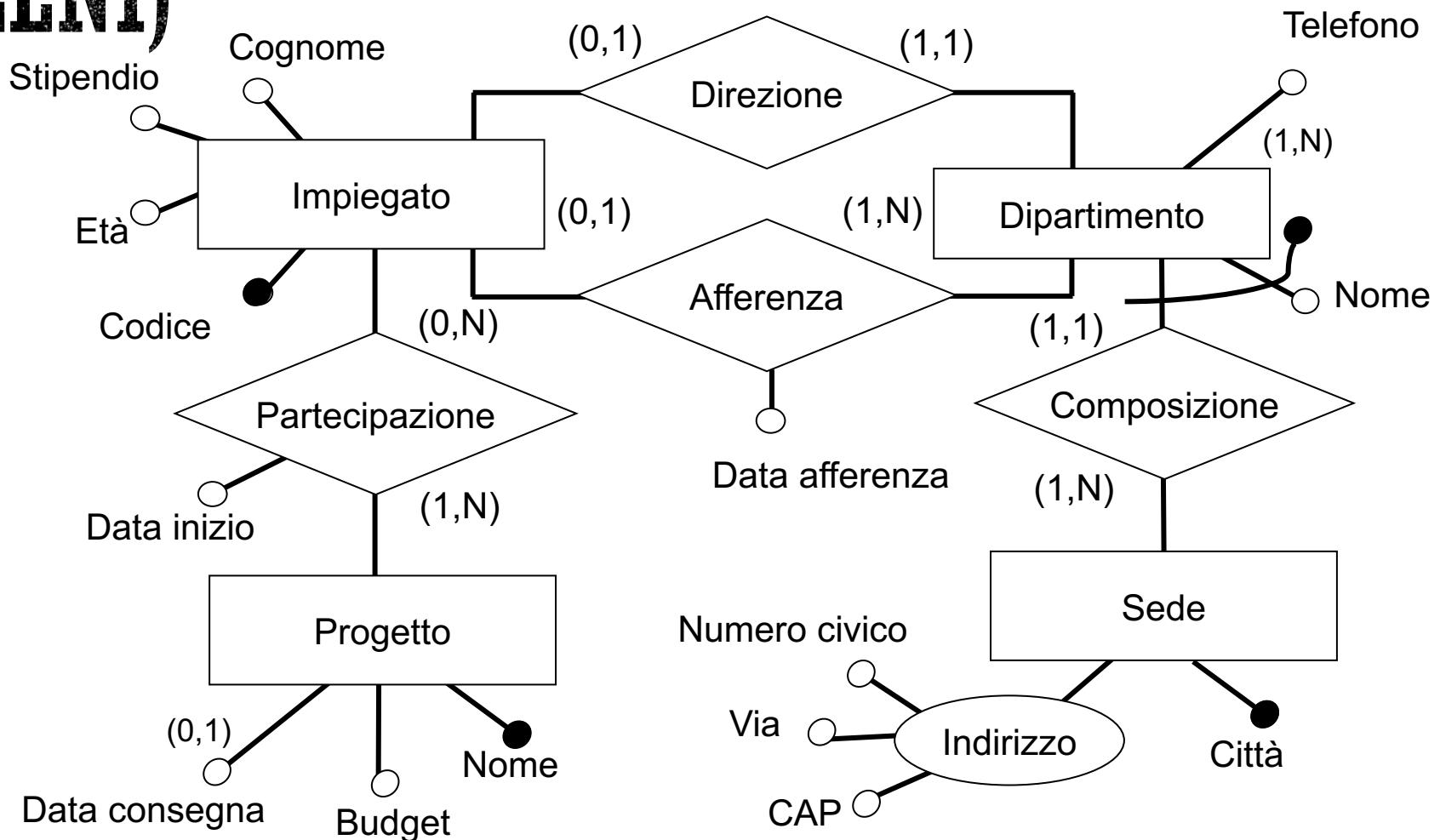


ESEMPIO

- *Modellare uno schema riguardanti le informazioni di carattere organizzativo relative ad una azienda con diverse sedi.*
 - Una **sede** dell'azienda è identificata dalla città ed è composta da una serie di dipartimenti che non possono essere definiti al di fuori della sede. Una sede ha anche un indirizzo.
 - Un **dipartimento** è identificato dal nome e dalla sede di appartenenza e possiede diversi numeri di telefono.
 - Ai dipartimenti afferiscono (a partire da una certa data) uno o più impiegati; e un impiegato li dirige.
 - Gli **impiegati** vengono rappresentati dal cognome, stipendio, età e un codice che serve per identificarli.
 - Gli impiegati lavorano su zero o più progetti a partire da una certa data.
 - Ogni **progetto** ha un nome, un budget e una data di consegna che può non essere specificata.
 - Una **nota** associata alla classe progetto ne descrive il significato.



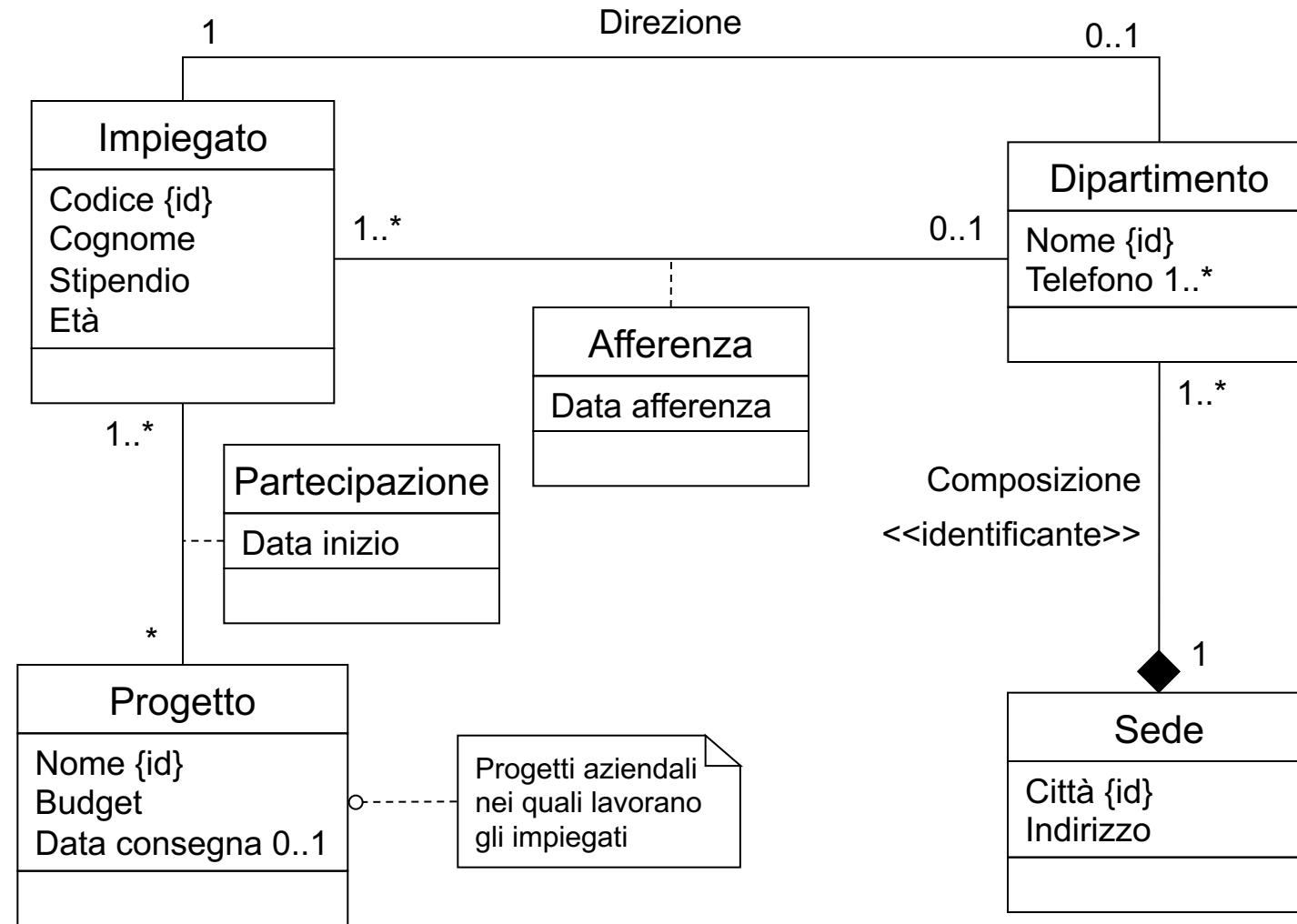
LO SCHEMA CONCETTUALE (NOTAZIONE ATZENI)



Indirizzo è stato modellato come attributo composto: Via, Numero civico, CAP



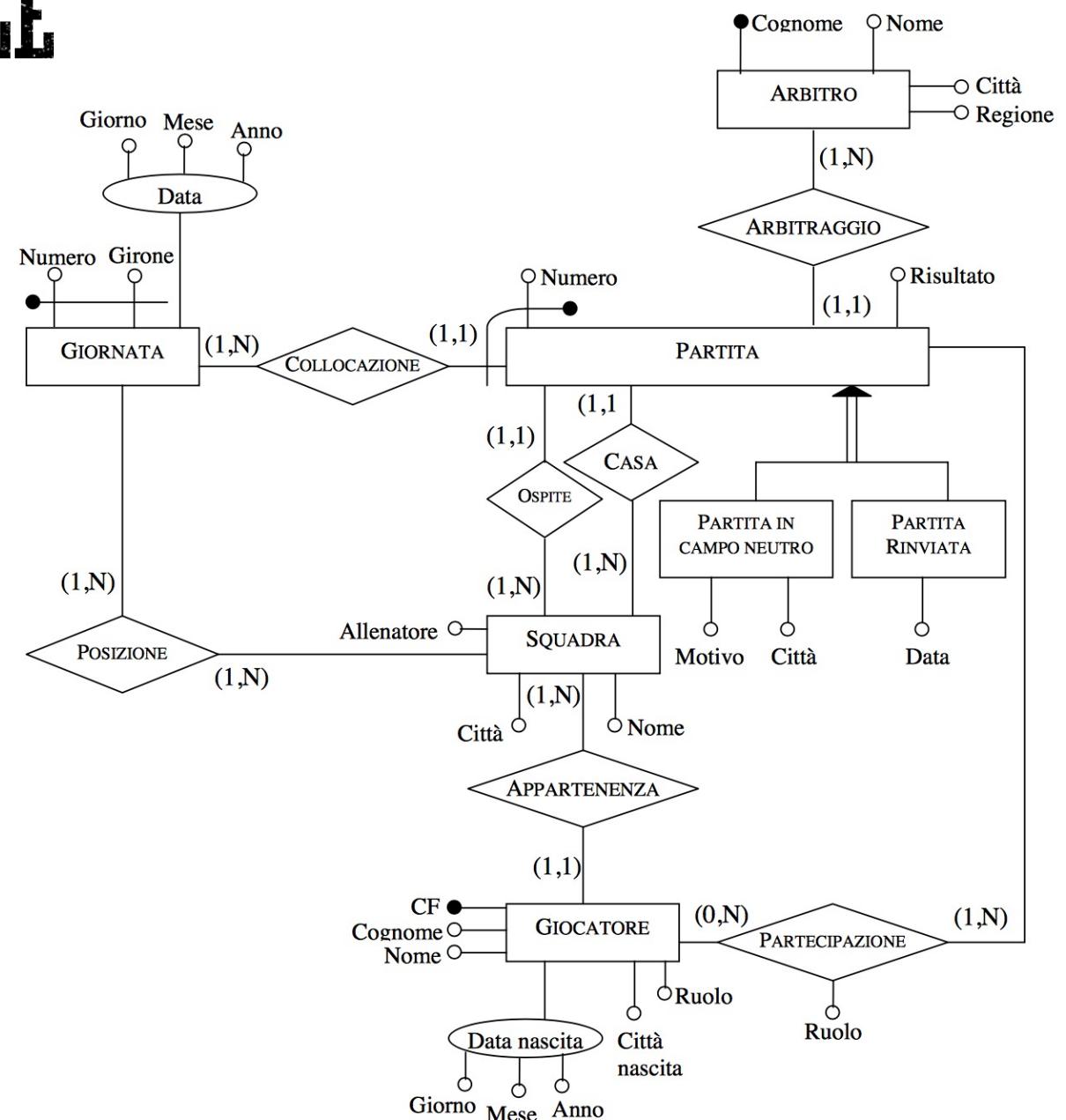
LO SCHEMA CONCETTUALE IN UML



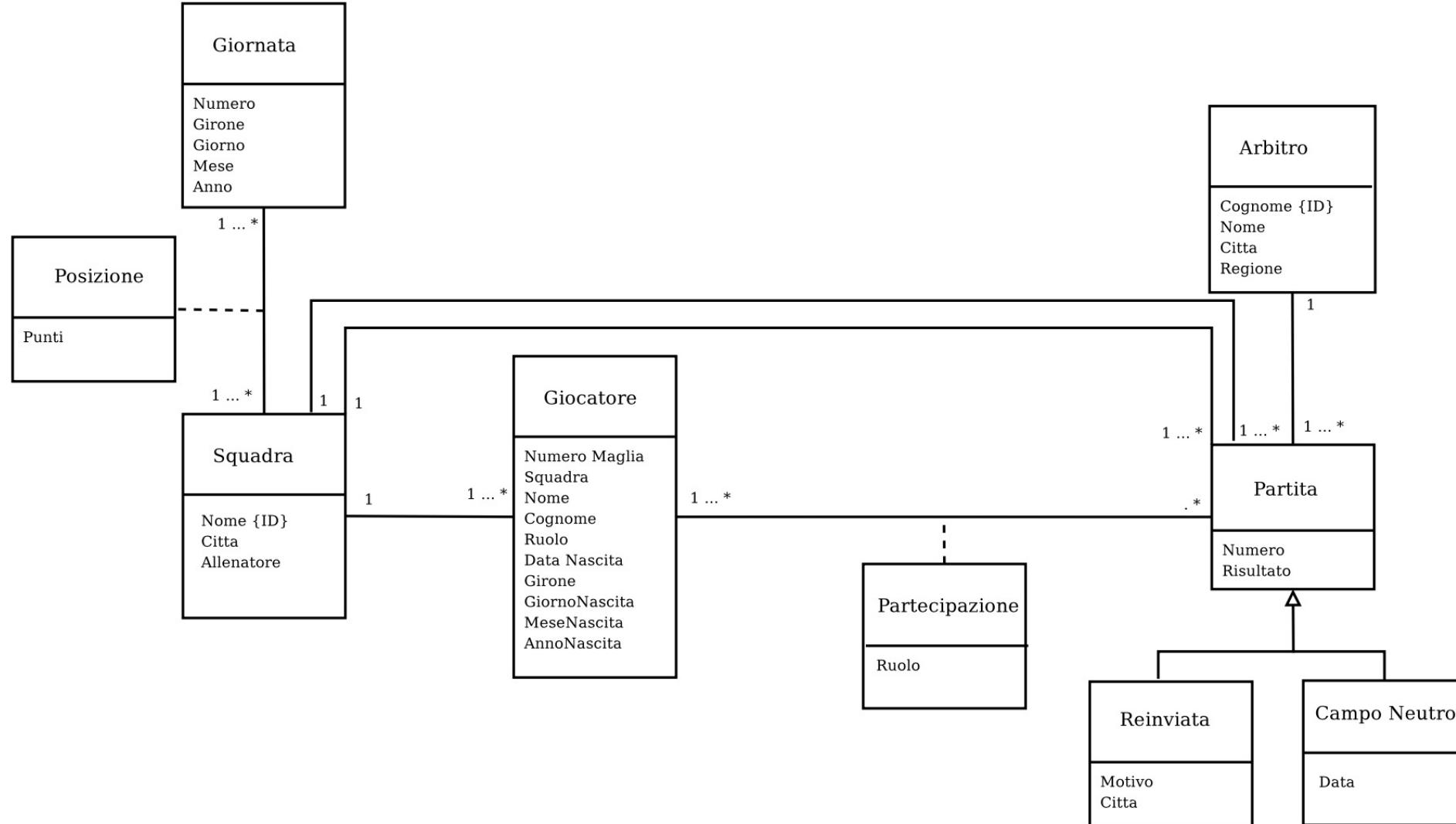
ESEMPIO 2

- *Modellare uno schema riguardanti le informazioni di un campionato di calcio:*
 - L'entità **SQUADRA** rappresenta tutte le squadre del campionato, indicando per ognuna di esse il nome, la città e il nome dell'allenatore.
 - L'entità **GIOCATORE** rappresenta i giocatori delle squadre: ogni giocatore ha un contratto con una sola squadra e ogni squadra ha più giocatori. I giocatori sono identificati dal loro Codice Fiscale e per ognuno di essi è indicato il nome, il cognome, il ruolo nella squadra, la città di nascita e la data di nascita.
 - Lo schema contiene anche informazioni sulle partite. Una **PARTITA** è identificata con un numero (che deve essere differente per tutte le partite dello stesso giorno) e con un riferimento al giorno (attraverso la relazione COLLOCAZIONE e l'entità giornata).
 - Le relazioni CASA e OSPITE rappresentano le due squadre che giocano la partita: per ogni partita è indicato il risultato e l'**ARBITRO**, con la relazione ARBITRAGGIO tra partita e arbitro; questa entità rappresenta tutti gli arbitri del campionato e per ognuno di essi è indicato il Nome, il Cognome, la Città e la Regione. Un arbitro è rappresentato solo se ha arbitrato almeno una partita.
 - Una partita può essere giocata su campo neutrale o può essere rinviata ad un'altra data (ma questi due eventi non sono ammessi contemporaneamente nello schema).
 - La relazione Partecipazione rappresenta il fatto che un giocatore abbia giocato in una partita, la sua posizione (che può essere diversa dalla sua solita). Lo schema non esprime la condizione che i giocatori che giocano una partita devono avere un contratto con una delle due squadre.
 - L'entità **GIORNATA** rappresenta la giornata del campionato. Sono identificate con Numero e Girone. La relazione Posizione dà il punteggio di ogni squadra in ogni giornata.

LO SCHEMA CONCETTUALE (NOTAZIONE ATZENI)



LO SCHEMA CONCETTUALE IN UML





FINE

Per eventuali domande: (in ordine di preferenza personale)

- Ora.
- Chat di Teams
- Mail: silvio.barra@unina.it

