Basi Di Dati

Info

Docente
Annalisa Franco -> annalisa.franco@unibo.it
Tutor
Giacomo Cavalieri -> giacomo.cavalieri3@unibo.it
Andrea Negri -> andrea.negri6@unibo.it
Ricevimento
Annalisa Franco -> Venerdì 14:30-16:30
Lab
Martedì -> 14:00-16:00

Elaborato
Singolo o gruppo (max 3)
Punteggio da 0 a 4 che vengono sommati al voto della prova scritta
Pubblica direttamente il voto finale su almaesami, **max 2 rifiuti** non cè bisogno di rifare l'elaborato
Validità **illimitata**
Realizzare elaborato significa:

- **progettare** il database e **documentare** tutte le fasi della progettazione in una relazione
- **creare** il database utilizzando DBMS relazionale
- realizzare **un'applicazione** che si interfacci con il database per lo svolgimento delle operazioni previste dal sistema

Più operazioni corpose come un po di statistica (numero di acquisti di un prodotto particolare in un periodo di tempo), meno operazioni banali come cancellazione e modifica.

Sistema Informativo

Complesso sistema di procedure, informatizzate e non, che permettono di gestire le informazioni utili ai processi aziendali. Richiede la conoscenza dei:

- Processi aziendali
- Informazioni necessarie ai processi
- Struttura aziendale

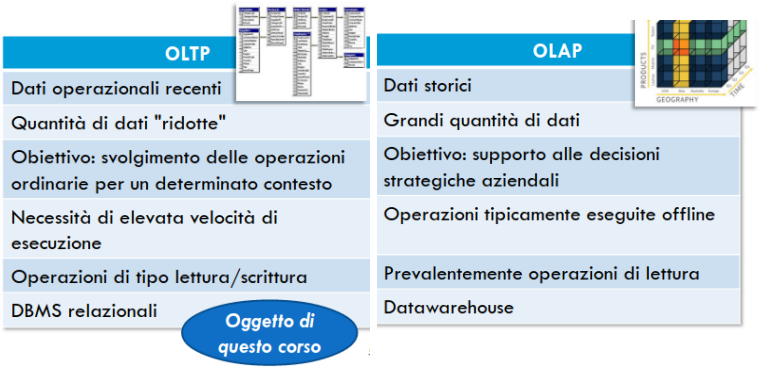
Progettare un Sistema Informativo richiede competenze anche in ambito di organizzazione aziendale, economia, psicologia ecc.

Base di Dati

Si tratta di un componente del Sistema Informativo atta alla memorizzazione strutturata delle informazioni. **Funzione:** fornire un **supporto informativo** per la memorizzazione dei dati.

Database: rappresenta una collezione di dati d'interesse per una o più applicazioni; nel contesto del corso è intesa come una collezione di dati gestita tramite un DBMS. I dati sono strutturati e collegati a livello logico, nel rispetto del **modello di rappresentazione** adottato dal DBMS e, a **livello fisico**, risiedono su dispositivi di memoria organizzati in particolari strutture.

DBMS (Data Base Management System): è un sistema software in grado di gestire **efficientemente** le informazioni necessarie a un SI, rappresentandone i dati in **forma integrata**, secondo un modello logico, garantendone la **persistenza**, **condivisione**, **l'affidabilità** e la **riservatezza**.



Applicazioni legate alla gestione dei **big data** o sistemi **NOSQL**

- Necessità di archiviare e gestire **grandi quantità di dati eterogenei** (immagini, testo, video)
- Principali caratteristiche:
  - **Scalabilità**: dati memorizzati in sistemi distribuiti, aggiunta di ulteriori nodi all'aumentare del volume dei dati
  - **Disponibilità:** possibilità di accedere ai dati in modo continuativo, anche a fronte di guasti di singoli nodi
  - **Sharding:** distribuzione del carico di lavoro su più nodi
  - Accesso ai dati ad **alte prestazioni:** utilizzo di tecniche di hashing o partizionamento per migliorare l'efficienza
- Rispetto ai sistemi relazionali:
  - Memorizzazione di dati **semi-strutturati** che si autodescrivono (JSON o XML)
  - Linguaggi di **interrogazione meno potenti**
  - **Versioning** per la storicizzazione dei dati

Data Base Administrator (DBA): installa, configura e gestisce il DBMS:

- crea gli oggetti logici necessari per le applicazioni
- crea gli utenti e concede loro i dovuti privilegi
- garantisce la sicurezza e l'integrità dei DB
- monitora e ottimizza le performance dei DB e delle applicazioni che li utilizzano
- pianifica strategie di backup e recovery

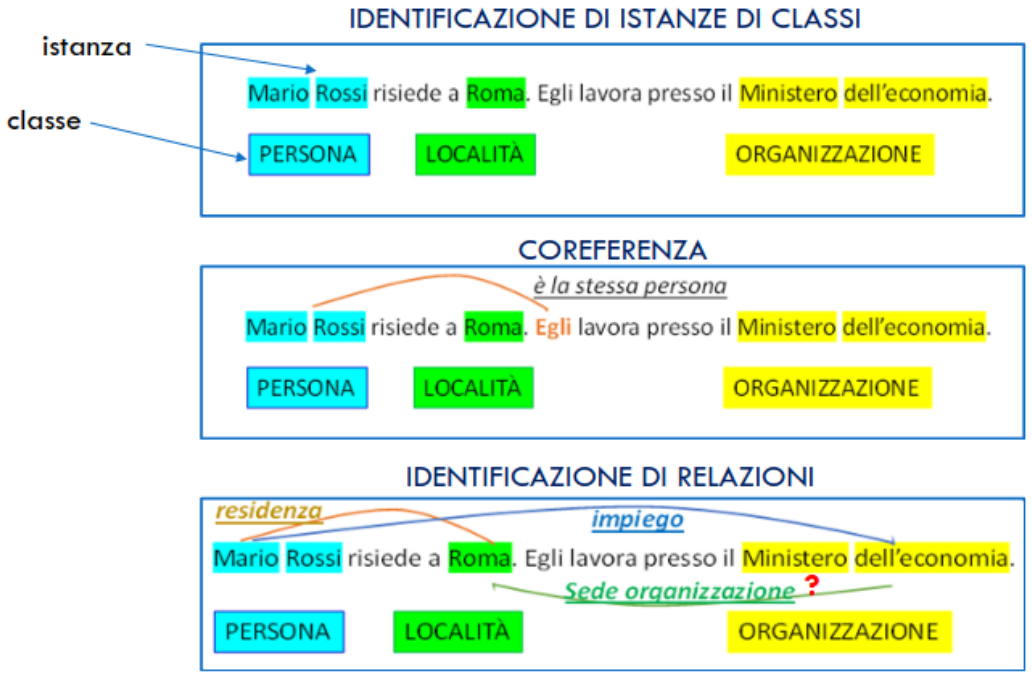
Data Base Designer: cura la progettazione di un modello dettagliato del DB da implementare; il modello esplicita tutte le scelte progettuali a livello concettuale, logico e fisico.

Software Engineer: analisti di sistema e programmatori di applicazioni; i primi determinano le esigenze degli utenti finali e dettano specifiche per le transazioni che saranno realizzate a cura dei programmatori.

End User: sono persone che interagiscono a vari livelli con una o più basi di dati per lo svolgimento delle proprie attività lavorative o per esigenze di altra natura. Possono essere divisi in due classi:

- ➔ **Naive End User**
  - ◆ accede al DB tramite query preconfezionate all'interno di un'applicazione
  - ◆ non ha di norma nessuna conoscenza né del DBMS né del DB
- ➔ **Sophisticated End User**
  - ◆ ha un certo grado di conoscenza della struttura del DB e delle potenzialità del DBMS
  - ◆ è in grado di interagire direttamente con la base di dati, attraverso l'uso di un linguaggio d'interrogazione, o indirettamente attraverso l'uso di interfacce e/o strumenti avanzati

Dato: è ciò che è immediatamente presente alla conoscenza prima di ogni elaborazione, si presentano sotto varie forme; ad essi si deve attribuire un **significato** affinchè rappresentino una realtà di interesse.
Informazione: notizia, dato o elemento che consente di avere conoscenza più o meno esatta di fatti, situazioni, modi di essere.

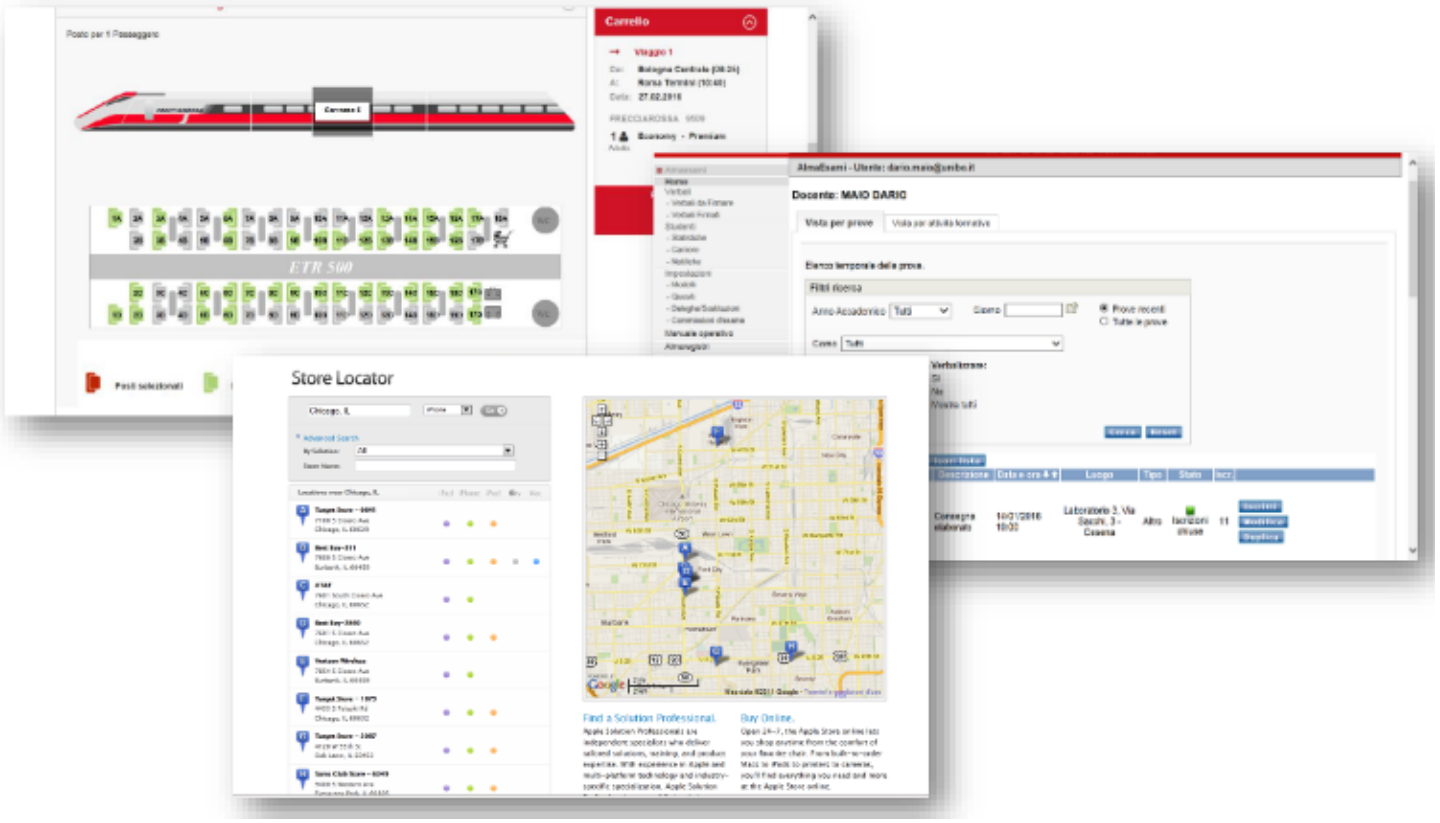


I **SI** affondano le proprie radici sulla correlazione tra **informazione** e **decisione**, e tra **informazione** e **controllo**.

Un SI deve provvedere alla **raccolta** e alla **classificazione** delle informazioni, da attuarsi con procedure **integrate** e **idonee**, al fine di produrre **in tempo utile** e **ai giusti livelli** le **sintesi** necessarie per i **processi decisionali**, nonchè per **gestire** e **controllare** le attività dell'ente nel suo complesso.

Classificazione SI

Transaction Processing System



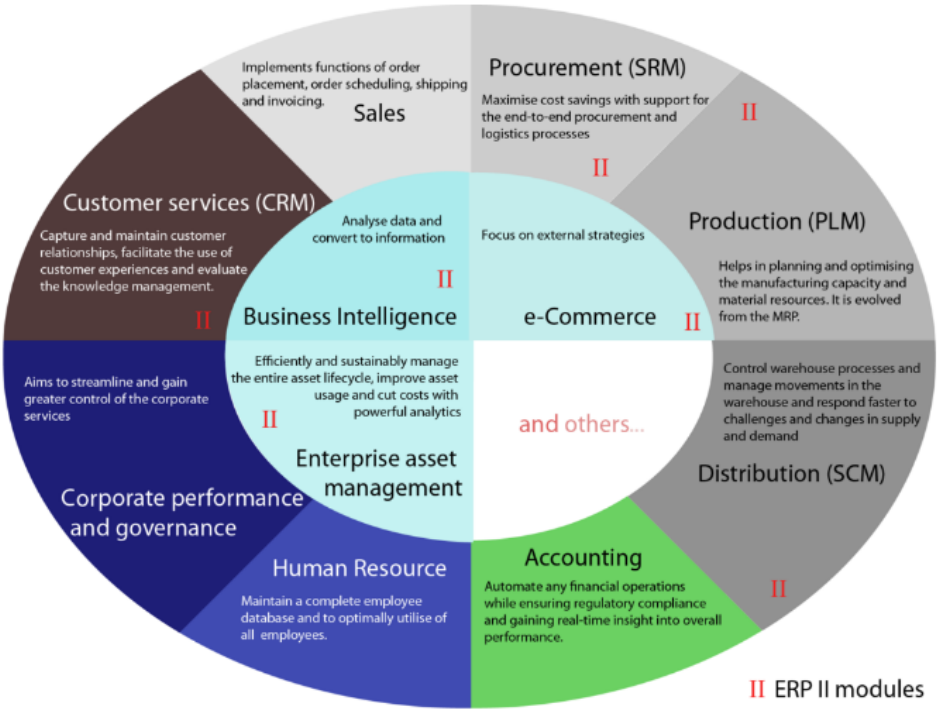
Management Information Systems

ERP (Enterprise Resource Planning)

Sistema di gestione che integra i processi di business rilevanti di un'azienda:

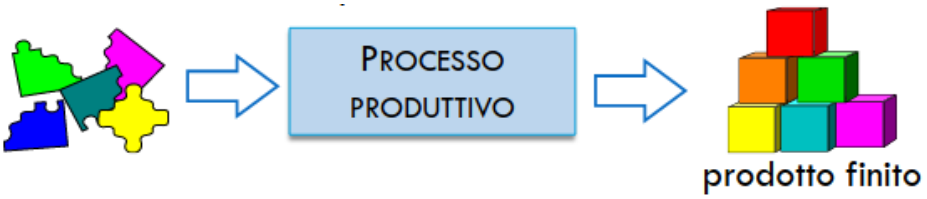
- I dati vengono raccolti in modo centralizzato nonostante provengano da molteplici parti dell'ente

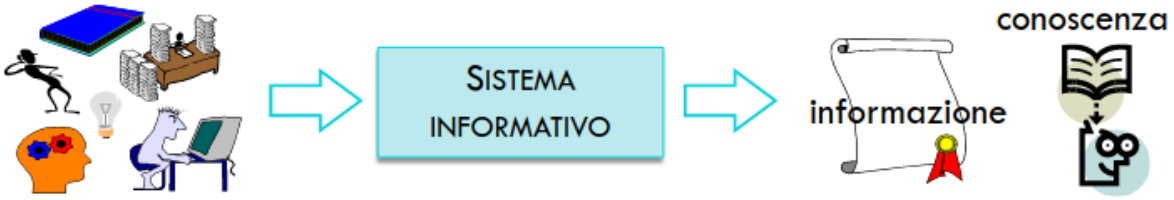
- Migliora l'efficienza dell'ente, riducendo costi e rischi, e aumenta il cotrollo sulla gestione delle varie risorse



La risorsa informazione

L'informazione è un bene, o merce, a valore crescente richiesto dalla direzione per pianificare e controllare con efficacia le attività di un'organizzazione. I SI trasformano dati in informazioni e conoscenza così come materie prime e semilavorati sono trasformati in prodotti finiti dai sistemi di produzione.





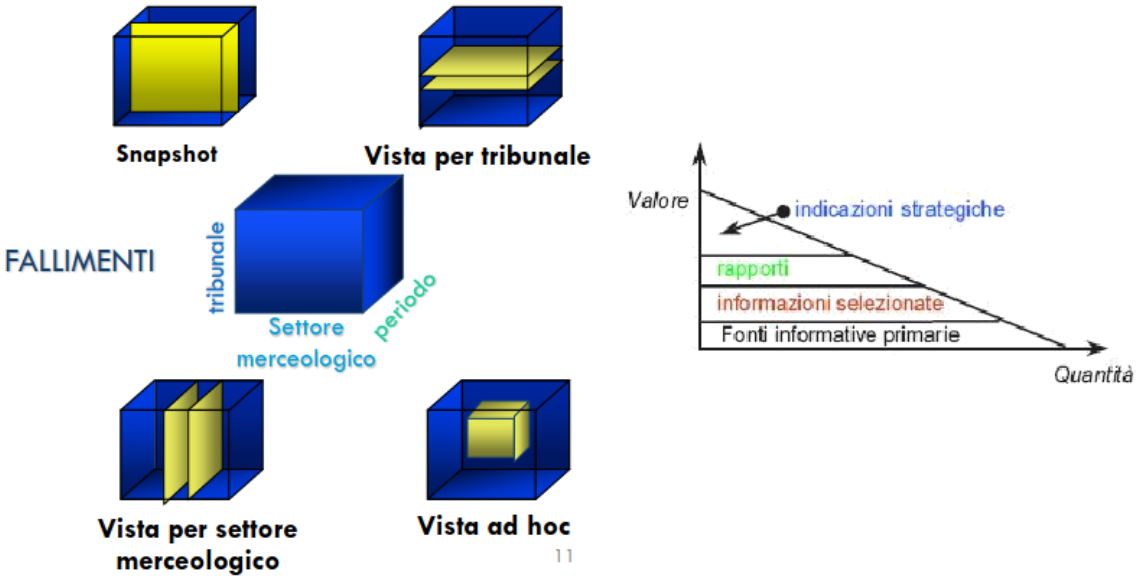
Che cosa rende un'informazione utile nel processo produttivo?

- **Soggettività:** il valore associato a un'informazione differisce da individuo a individuo e dipende dal tipo di decisione
- **Rilevanza:** l'informazione deve essere pertinente alla decisione da prendere
- **Tempestività:** l'informazione è utile alla decisione solo se è disponibile nel momento decisionale
- **Accuratezza:** le informazioni devono essere e precise

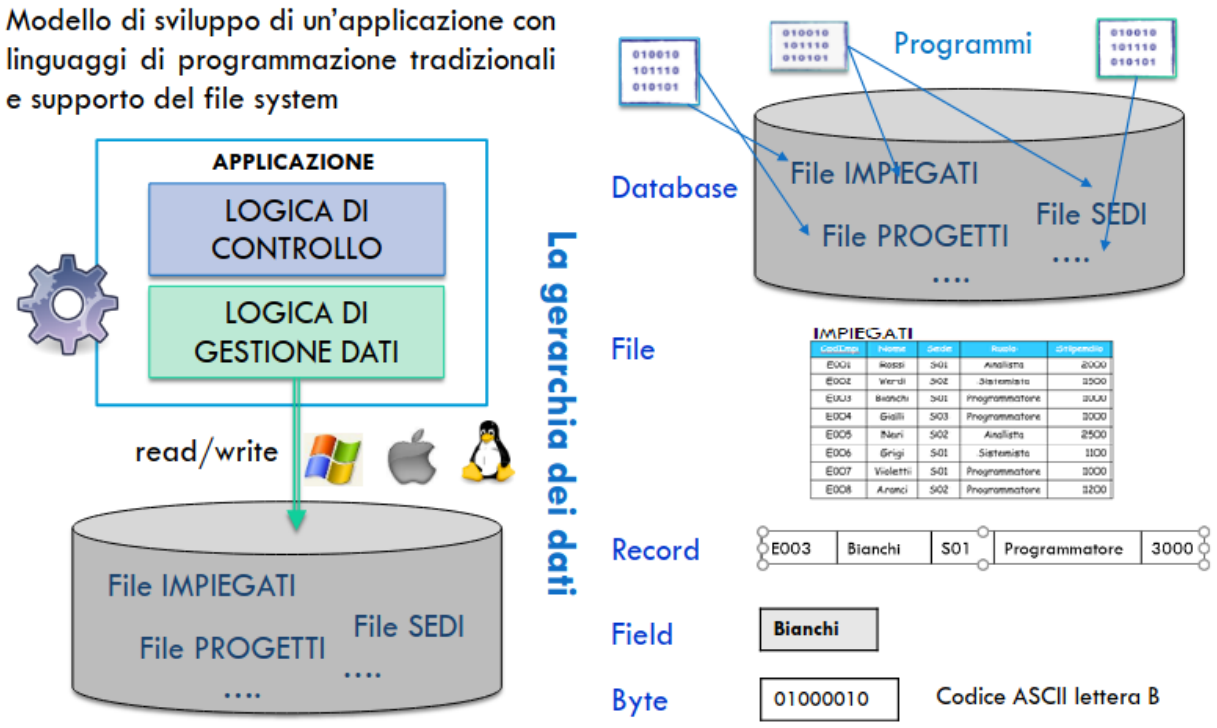
- **Presentazione:** l'informazione deve essere utilizzabile direttamente per la decisione senza ulteriori elaborazioni
- **Accessibilità:** le informazioni devono essere disponibili appena necessarie a chi le richiede
- **Completezza:** il **decisore** deve avere a disposizione tutte le informazioni necessarie per prendere una decisione corretta

Valore dell'informazione

L'informazione è una risorsa alla stessa stregua del capitale, delle materie prime, degli impianti e delle persone.



Gestione dei dati con file system



Sistemi informatici settoriali

Un'**organizzazione** è solitamente articolata in diversi **settori di competenza**, il **flusso di informazioni intra-settoriale è più intenso rispetto al flusso inter-settoriale.**
Questa considerazione, unitamente ad aspetti di natura tecnologica, ha fatto sì che prima dell'avvento dei **DBMS** la gestione fosse realizzata con programmi applicativi ad hoc ciascuno pensato per i dati specifici di un particolare settore, con **file condivisi con altri settori**.
Nei **sistemi settoriali** del passato il flusso di dati da un settore all'altro era di norma gestito tramite la **creazione e trasmissione di copie** in formato elettronico e/o cartaceo

Problemi dei sistemi informatici settoriali

- La **progettazione** degli archivi di settore è effettuata sulla base di **considerazioni locali**.
    - La mancanza di standard a livello globale complica la gestione dei flussi inter-settoriali e può creare problemi di incompatibilità nelle rappresentazioni adottate
- I dati sono soggetti a diversi **vincoli di integrità**, che riflettono la conoscenza della realtà specifica rappresentata.
    - Se i vincoli emersi in fase di analisi dipendono solo da specifiche considerazioni settoriali, è evidente che si possono generare **inconsistenze**.
- La presenza di **copie dello stesso dato** dà luogo a **ridondanze**:
    - Si ha uno spreco di memoria
    - Inoltre, la ridondanza può dar luogo a problemi d'inconsistenza delle copie e comporta la necessità di propagare le modifiche, con ulteriore spreco di risorse
- L'uso di **file non condivisi** e il mero ricorso ai servizi del file system presenta molteplici problemi in termini di **facilità d'accesso ai dati**, di sviluppo delle applicazioni, di flessibilità, di controllo degli accessi concorrenti, di sicurezza ecc.

- Oggi un approccio alla gestione dei dati basato su file può essere accettabile **solo nel caso di**:
    - **sistemi di piccole dimensioni**
    - prevalentemente **a uso personale**
    - con **scarsa necessità di condivisione dei dati**

Perché non adottare un file system?

Necessità: gestire grandi quantità di dati, potenzialmente scalabili in dimensione, in modo persistente e condiviso, e consentire di soddisfare esigenze applicative che mutano nel tempo
Soluzione con **file system:**

- **Povertà dell'astrazione**
- **Difficoltà per l'accesso alle informazioni**
- I meccanismi di **condivisione** sono in genere limitati
- Non sempre sufficiente i meccanismi forniti per la **protezione a fronte dei guasti**
- **Vincoli d'integrità**
- Non sono disponibili i vari **servizi aggiuntivi** offerti da un **DBMS**

Peculiarità di un DBMS

Un **DBMS** è un sistema software che:

- gestire **grandi quantità di dati persistenti e condivisi**
- offre un supporto per almeno un **modello dei dati** in grado di fornire agli utenti un'estrazione di alto livello attraverso cui definire strutture di dati complesse a interagire con il DB
- attua **indipendenza tra programmi e dati, tra programmi e applicazioni**
- garantisce **efficienza** nella gestione di grandi quantità di dati e persegue obbiettivi di efficacia, nel senso di supporto soddisfacente alla produttività degli utenti

La **persistenza** e la **condivisione** richiedono che un DBMS fornisca meccanismi per garantire l'affidabilità dei dati (**fault tolerance**), per il controllo degli accessi, nel senso di supporto soddisfacente alla produttività degli utenti

Ciclo di vita di un Sistema Informativo

- Definizione strategica, **strategic study**
- Pianificazione, **information system planning**
- Analisi dell'organizzazione, **business analysis**
- Progettazione del sistema, **system design**
- Progettazione esecutiva, **construction design**
- Realizzazione e collaudo in fabbrica, **construction and workbench test**
- Installazione, **installation**
- Collaudo del sistema in fabbrica, **test of installed system**
- Esercizio, **operation**
- Evoluzione, **evolution**
- Messa fuori servizio, **phase out**
- Post mortem

Progettazione di DB e di applicazioni

Di grande rilevanza è l'adozione di **corrette metodologie** per l'analisi e per la specificazione dei requisiti. Il processo di analisi è incrementale e porta per passi successivi alla stesura di un **insieme di documenti** in grado di rappresentare un modello dell'organizzazione e comunicare, in modo non ambiguo, una **descrizione esauriente, coerente e realizzabile** dei vari aspetti **statici, dinamici** e **funzionali**.

La progettazione di una base di dati e delle relative applicazioni è una delle **attività principali del processo di sviluppo** di un sistema informativo.

Fasi in cui interviene la progettazione del DB

- **Definizione strategica**
  - si assumono decisioni sulle aree aziendali che devono essere oggetto di automazione
- **Pianificazione**
  - si definiscono gli obiettivi e si evidenziano i fabbisogni, viene condotto uno **studio di fattibilità**, per individuare possibili strategie d'attuazione e avere una prima idea dei costi, dei benefici e dei tempi

- **Analisi dei requisiti**
  - si **formalizzano i requisiti**, avvalendosi di tecniche di modellazione della realtà e si producono macro-specifiche per la fase di progettazione
- **Progettazione del sistema**
  - si interpretano i requisiti in una **soluzione architetturale di massima**. Sono prodotte specifiche formali indipendenti in linea teorica dai particolari strumenti che saranno usati per la costruzione del sistema
- **Progettazione esecutiva**
  - le specifiche del passo precedente sono rese **vincolanti** per lo staff addetto alla realizzazione, descrivendo la struttura dei componenti dell'architettura hardware, software e di rete. Queste specifiche devono essere tali da poter dar luogo, attraverso il ricorso a strumenti di sviluppo opportuni, a un **prodotto funzionante**.

Un'organizzazione con il proprio SI opera in base a un modello della realtà.

Progettazione guidata dai dati

Due aspetti di primaria importanza nella progettazione di un SI:

- progettazione della **base di dati**
- progettazione delle **applicazioni**

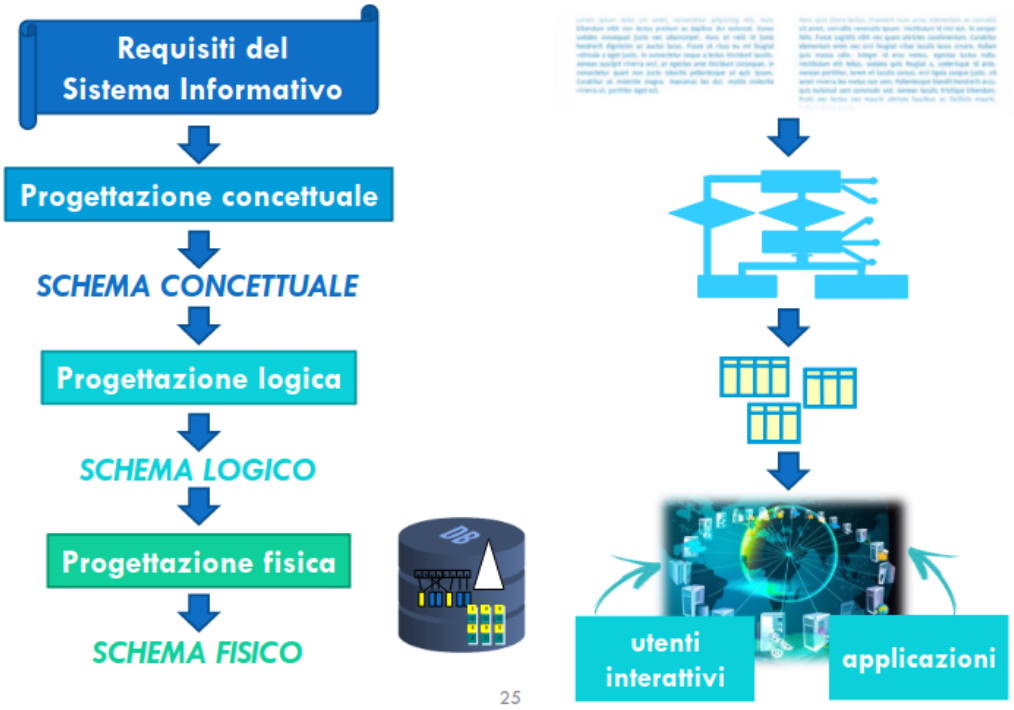
Il ruolo primario viene molto spesso svolto dai **dati**, in quanto:

- sono strutturalmente **più stabili nel tempo**
- sono **condivisi** da più applicazioni

Necessità di una **metodologia di applicabilità generale**, di facile uso e con supporto di strumenti automatici, in grado di:

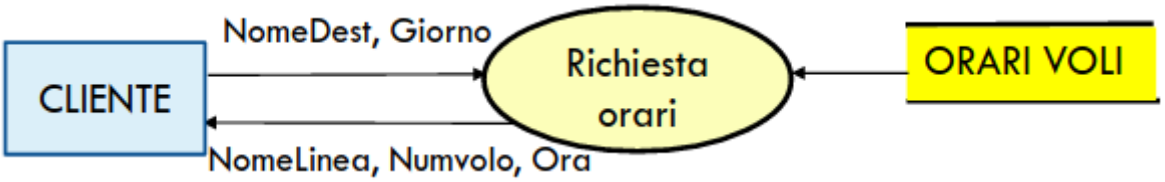
- definire le fasi in cui l'attività di progettazione si articola
- fornire criteri per scegliere tra diverse alternative
- rendere disponibili adeguati modelli di rappresentazione
- fornire supporto per lo sviluppo del software

Progettazione di una base dati



Progettazione dell'applicazione

Obiettivo -> realizzare **moduli software** per le funzioni e per l'interfaccia utente



Progettazione di un database

Scenari per la progettazione DB

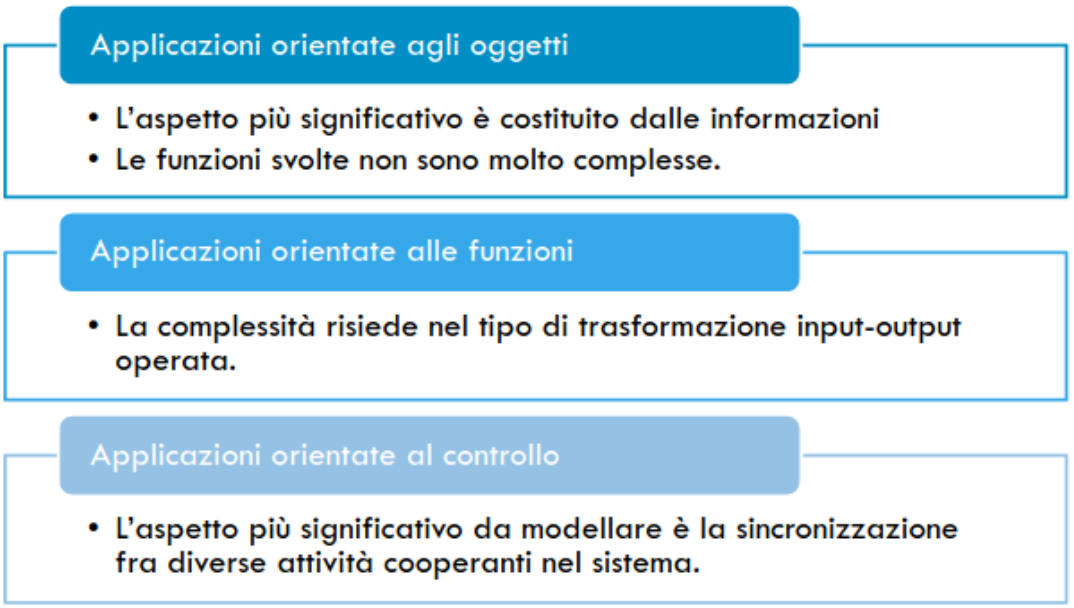
La progettazione è svolta nell'ambito della più ampia attività di sviluppo **ex novo** di un sistema informativo:

- in questo scenario i task specifici relativi alle basi di dati devono essere coordinati con vari altri aspetti di design del SI e possono condizionare anche le scelte delle configurazioni architetturali

Si richiede di **sviluppare un nuovo DB** con le relative applicazioni, o **reingegnerizzare un SB esistente** eventualmente aggiungendo funzionalità:

- in questo scenario l'attività di progettazione è svolta come un processo a sé stante
- non si esclude che si rendano necessarie modifiche e/o estensioni dell'architettura del sistema informatico

Tipologia applicazioni



Aspetti

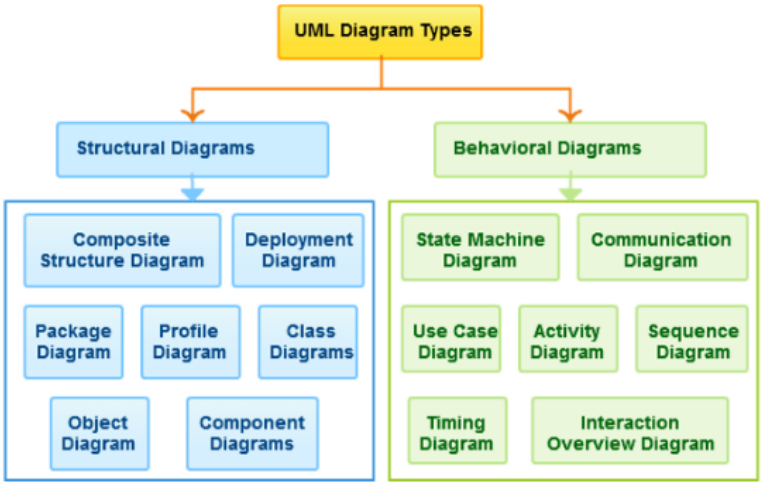
Oggetti (statici): possono essere descritti a partire da termini molto generici fino ad arrivare a livello di dettaglio specifici

Funzioni (funzionali): possono essere espresse inizialmente in modo vago e successivamente precisate

Stati (dinamici): possono essere descritti a un elevato livello di astrazione o specificati in maggior dettaglio

Tassonomia dei metodi di analisi

L'orientamento di un metodo è determinato sia dalla tipologia di SI sia dall'approccio seguito dal team che pone un diverso avvento nella modellazione della realtà verso un aspetto ritenuto predominante
La tendenza attuale consiste nell'integrare modelli di rappresentazione nati per finalità diverse; ad esempio il linguaggio di modellazione **UML** (**Unified Modeling Language**).

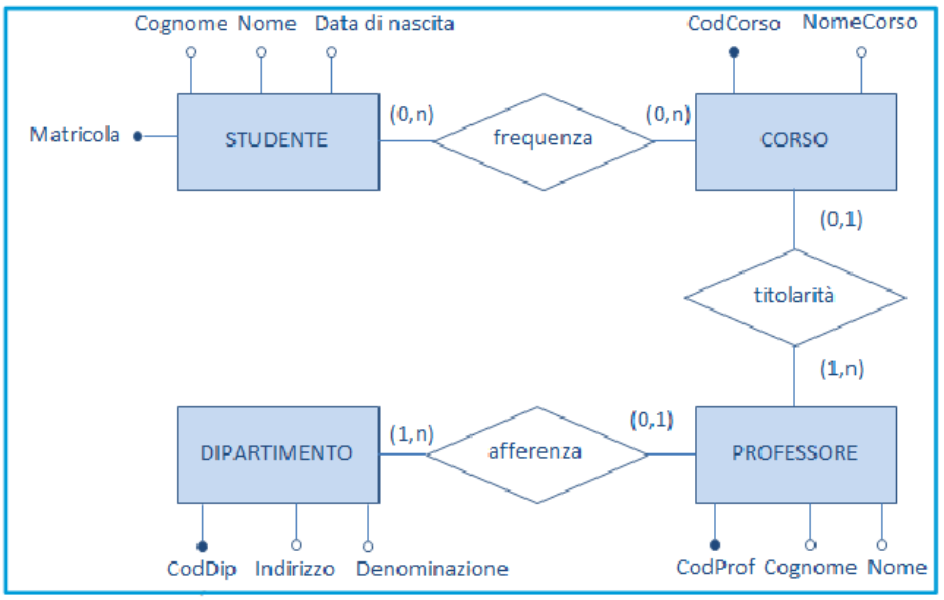


Analisi orientata agli oggetti

L'enfasi è posta:

- sull'**identificazione degli oggetti e sulla loro classificazione**
- sulle **interrelazioni tra oggetti**

Esempio modellazione E/R

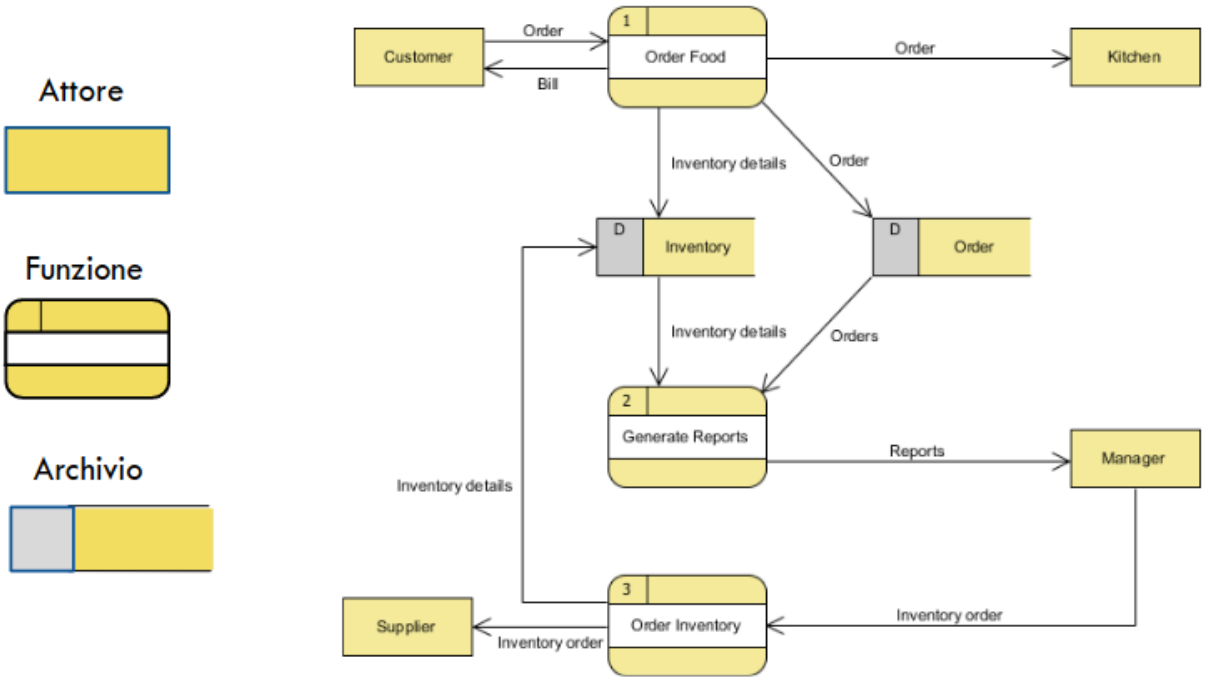


Analisi orientata alle funzioni

L'obiettivo è rappresentare un sistema come:

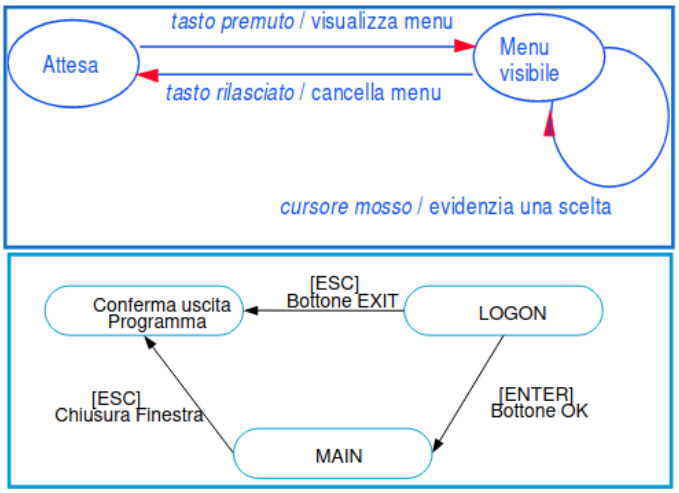
- una **rete di processi**
- un **insieme di flussi informativi tra processi**

Ciò corrisponde alla progressiva costruzione di una gerarchia funzionale; spesso in passato nella modellazione funzionale si è fatto ricorso alla rappresentazione **DFD** (**Data Flow Diagram**)



Analisi orientata agli stati

Per alcune categorie di applicazioni può essere utile pensare fin dall'inizio in termini di **stati** operativi, in cui si può trovare il sistema allo studio, e **transizioni di stato**.



Meccanismi di astrazione

Molteplici sono le relazioni in gioco fra oggetti, funzioni e stati e molteplici livelli di possibile dettaglio

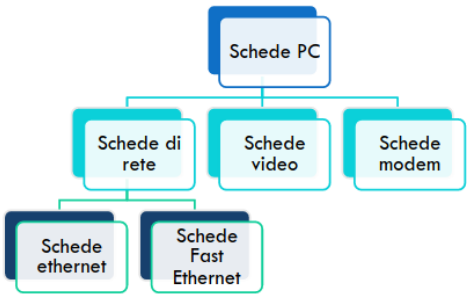
- L'analisi deve far ricorso a tecniche che gli consentano di **organizzare e interrogare la conoscenza** sul problema via via acquisita
- I principali meccanismi di astrazione usati durante il processo di analisi per costruire una base di conoscenza sul problema sono:
    - **classificazione**
    - **generalizzazione**
    - **aggregazione**
    - **proiezione**

Classificazione

Raggruppa gli oggetti in classi in base alle loro proprietà

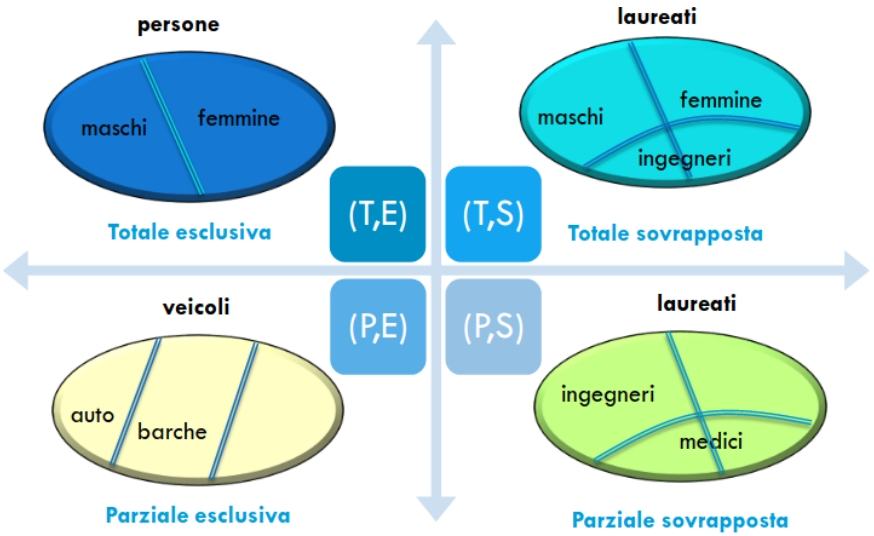
Generalizzazione

Cattura le relazioni di tipo << **è un** >> ovvero permette di astrarre le caratteristiche comuni fra più classi definendo super classi.



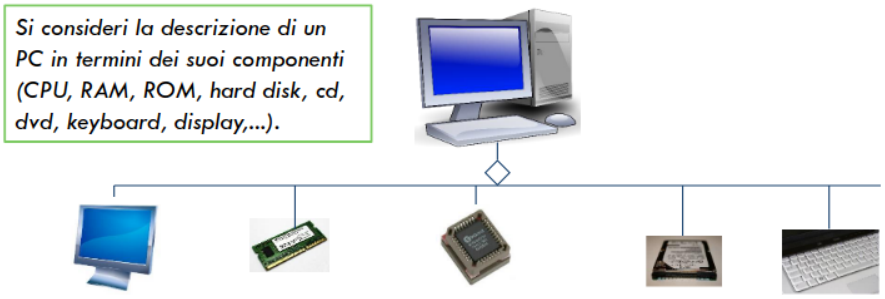
La **specializzazione** è il processo inverso della generalizzazione.

COPERTURA



Aggregazione

L'aggregazione esprime le relazioni parte di che sussistono tra oggetti, tra funzioni, o fra stati



Proiezione

Cattura la vista delle relazioni strutturali fra gli oggetti, le funzioni, gli stati

Associazioni

Le **associazioni** sono di fatto aggregazioni di cui le classi sono le componenti, dal punto di vista della modellazione, è importante caratterizzare queste corrispondenze in termini di vincoli di cardinalità.

Vincoli di cardinalità

Sia A un'associazione fra C1 e C2:

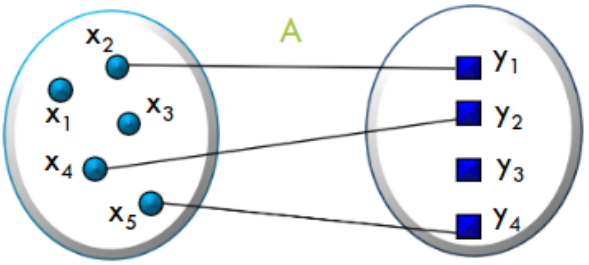
- **min-card (C1, A):** cardinalità minima di C1 in A
- **max-card (C1, A):** cardinalità massima di C1 in A

Vincoli di cardinalità minima

- **partecipazione opzionale:** min-card(C1, A) = 0
  - alcuni elementi di C1 possono non essere associati tramite A a elementi di C2
- **partecipazione obbligatoria (totale):** min-card(C1, A) > 0
  - a ogni elemento di C1 **deve essere associato**, tramite A, almeno un elemento di C2

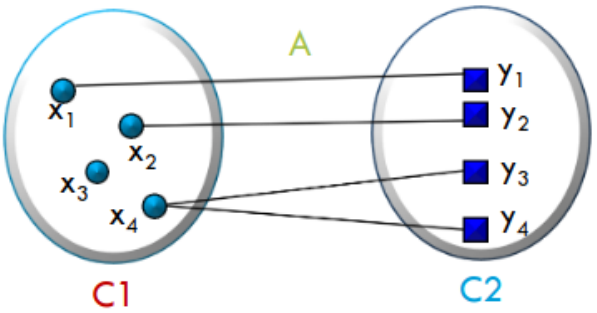
Associazioni binarie uno a uno

uno a uno (one-to-one)
max-card(C1, A) = 1
max-card(C2, A) = 1



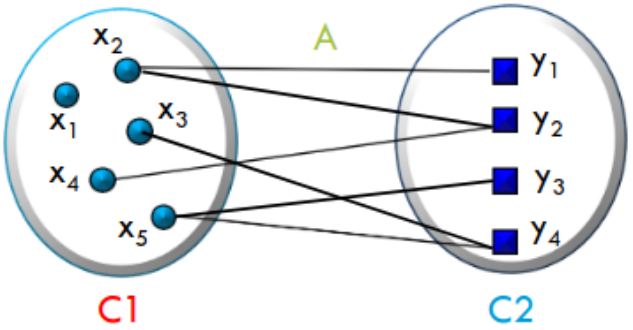
Associazioni binarie uno a molti

uno a molti (one-to-many)
max-card(C1, A) = N
max-card(C2, A) = 1
oppure
max-card(C1, A) = 1
max-card(C2, A) = N



Associazioni binarie uno a molti

molti a molti (many-to-many)
max-card(C1, A) = N
max-card(C2, A) = M



Tutte queste associazioni soddisfano i vincoli

Modello Entity-Relationship

Modelli dei dati: logici - concettuali

Un **modello dei dati** è una collezione di concetti che sono utilizzati per descrivere dati, associazioni e vincoli che devono essere rispettati.

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

Modelli concettuali dei dati

Lo scopo è pervenire a uno schema che rappresenti la realtà di interesse in modo indipendente dal DBMS.
Si cerca un livello di astrazione intermedio tra sistema e utenti, che sia al tempo stesso:

- **flessibile**
- **intuitivo**
- **espressivo**

Prevedono una rappresentazione grafica, il più noto è il **modello E/R**.

Modello Entity-Relationship

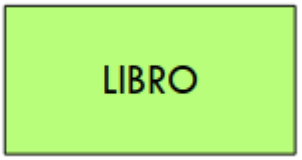
Proposto da Peter **Pin-Shan Chen** nel 1976, rappresenta oggi uno standard per la progettazione concettuale di una base di dati. Si usano anche termini EER o ERD intendendo una versione estesa rispetto al modello originario.
Caratterizzato da una rappresentazione grafica intuitiva che consente di disegnare schemi E/R facilitando la comprensione e l'interpretazione dei requisiti informativi modellati.

Concetti fondamentali del modello E/R

- **Entità**
- **Associazione**
- **Attributo**
- **Vincolo di cardinalità**
- **Identificatore**
- **Gerarchia di generalizzazione**

Entità

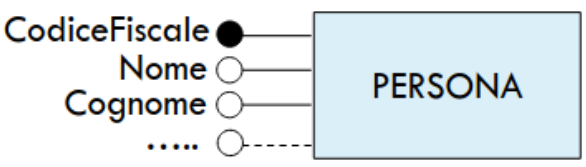
Rappresenta una classe di oggetti della realtà di interesse che possiedono caratteristiche comuni, e che hanno un'esistenza "autonoma".
Graficamente si rappresenta con un rettangolo.



Sconsigliato usare sigle in luogo di denominazione; non può avere elementi ripetuti (non possono esserci entità con lo stesso nome).

Livello intensionale

Lo schema che rappresenta un'entità ne descrive la natura, cioè l'aspetto intensionale. A **livello intensionale** un'entità **non è rappresentata** facendo esplicito riferimento alle sue singole istanze in un certo stato osservabile; si fa invece ricorso alla **classificazione** riconoscendo che le possibili istanze dell'entità sono caratterizzate da proprietà comuni.



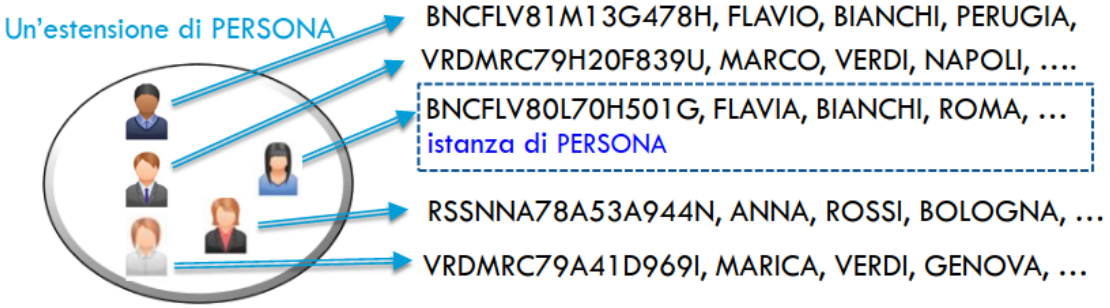
A questo **livello intensionale** si può parlare di **prototipo** d'istanza di entità.

Istanza

Un'**istanza** (elemento, occorrenza) **si un'entità** è uno specifico oggetto appartenente all'insieme che quella entità rappresenta.

Estensione di un'entità

A **livello estensionale** un'entità è un insieme di istanze ammissibili per quella entità, dunque un'estensione di un'entità a un certo tempo è un insieme di **specifici oggetti**.



Associazione

Un'**associazione** rappresenta un **legame logico tra entità**, rilevante nella realtà che si sta considerando.

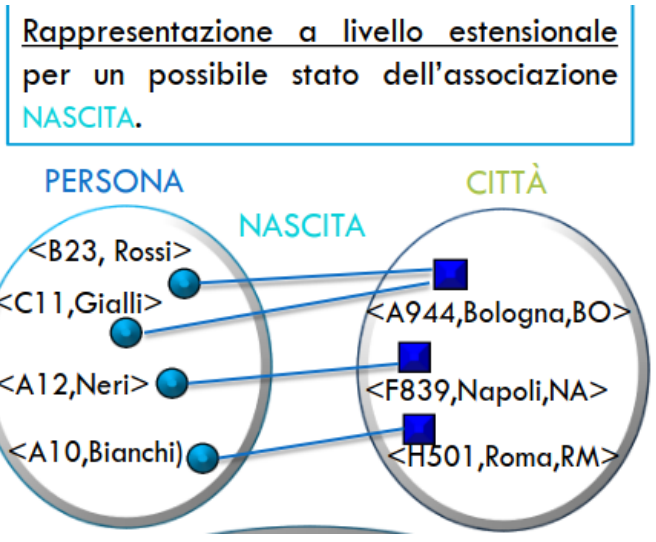
Istanza di associazione: combinazione di istanze delle entità che prendono parte all'associazione; dunque una ennupla costituita da occorrenze di entità, una per ogni entità coinvolta nell'associazione.

Tutte le ennuple devono essere distinte senza ripetizioni

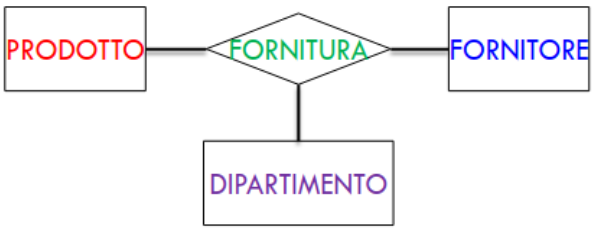
Associazione binaria, coppia (p, c)

Rappresentazione intensionale ed estensionale

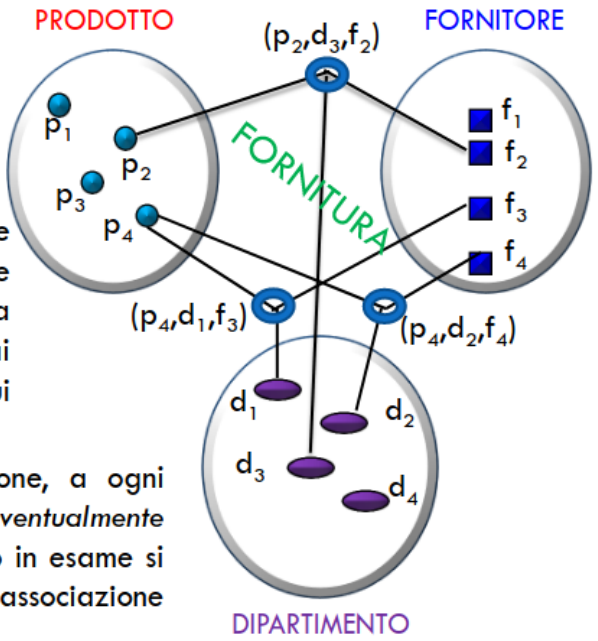
N.B. Per semplicità, per non appesantire la notazione si usa lo stesso nome per indicare un'entità (o un'associazione) a livello intensionale e a livello estensionale.



Rappresentazione a livello intensionale.



Rappresentazione a livello estensionale per un possibile stato dell'associazione FORNITURA.



Si ricorda che il simbolo usato per indicare un'istanza (es. p_1) rappresenta una notazione sintetica per indicare l'intero oggetto che a livello estensionale sarà caratterizzato dai valori delle proprietà definite per l'entità a cui appartiene.

N.B. Per ogni attributo definito sull'associazione, a ogni legame associativo si aggiunge un valore *(eventualmente assente se l'attributo è opzionale)*. Per l'esempio in esame si provi a visualizzare un'estensione dell'associazione definendo l'attributo Quantità per FORNITURA.

Associazione ternaria che collega tra diverse entità

Definizioni a confronto

Definizione adottata	Definizione di Elmasri-Navathe
entità	entity type (livello intensionale)
entità	entity set (livello estensionale)
istanza di entità	entity
associazione (a livello di entità)	relationship type
associazione (a livello di istanze)	relationship set
istanza di associazione	relationship instance

Denominazione di un'entità

- Il nome di un'entità deve essere univoco all'interno di uno schema
- Non esiste una convenzione universalmente accettata per il nome da assegnare a un'entità
- **Se si pone l'accento all'aspetto intensionale, è preferibile usare un sostantivo al singolare** (da usare)
- Se si pone l'accento all'aspetto estensionale, è preferibile usare un sostantivo al plurale
- Si preferisce scrivere il nome di un'entità in **maiuscolo** per rendere lo schema più leggibile

Denominazione associazioni

- Il nome di un'associazione deve **essere univoco all'interno di uno schema**
- È preferibile utilizzare un sostantivo al singolare invece di utilizzare un verbo
  residenza > risiede
- Sconsigliato l'uso di denominazioni che abbiano i nomi dell'entità partecipanti

Istanze di associazioni

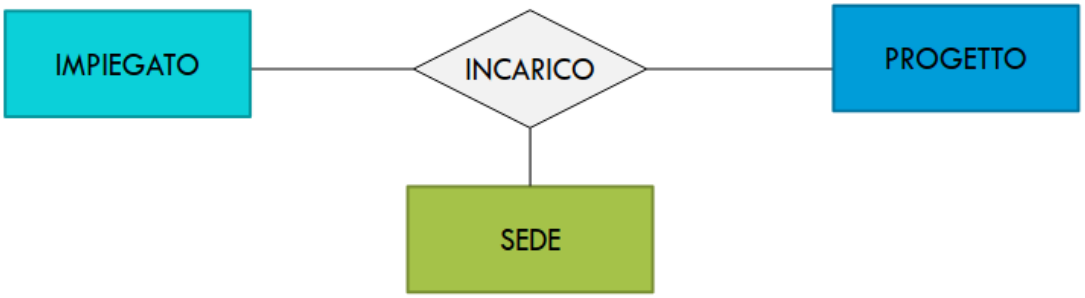
L'insieme delle istanze di un'associazione è un **sottoinsieme del prodotto cartesiano** degli insiemi delle istanze di entità che partecipano all'associazione. ne segue che **non possono esservi istanze ripetute in un'associazione**.

Grado delle associazioni

Un'associazione **n-aria** coinvolge **n** entità, non necessariamente distinte. Il **grado di un'associazione** è il numero di istanze di entità che sono coinvolte in un'istanza dell'associazione.

- **Associazione binaria:** grado = 2

- **Associazione ternaria:** grado = 3



È possibile stabilire più associazioni, con diverso significato, tra le stesse entità.

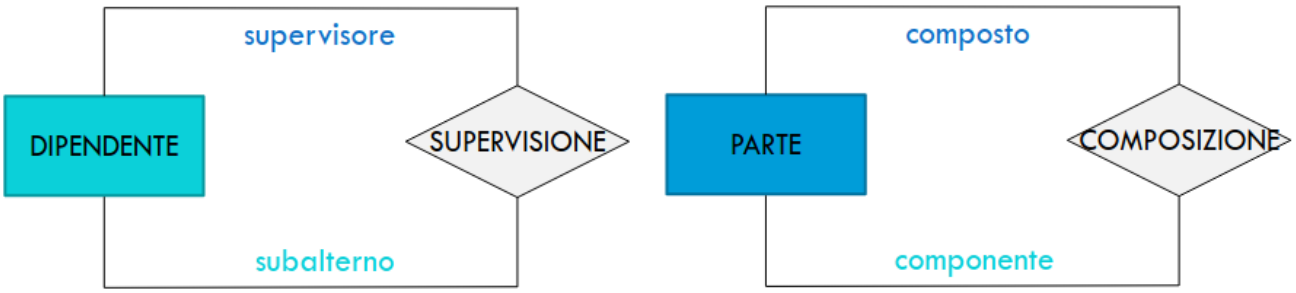
Associazione ad anello

Un'associazione ad anello lega un'entità con sé stessa, e quindi mette in relazione tra loro le istanze di una stessa entità

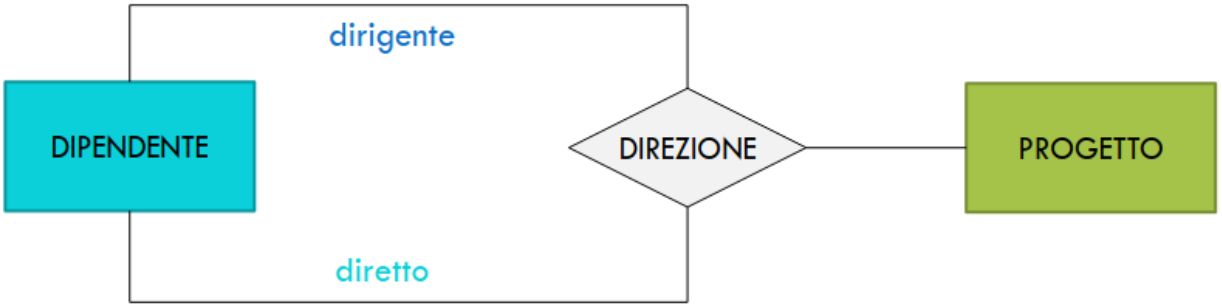
Un'associazione ad anello può essere o meno:

- Simmetrica: $(a,b) \in A \Rightarrow (b,a) \in A$
- Riflessiva: $(a,a) \in A$
- Transitiva: $(a,b) \in A, (b,c) \in A \Rightarrow (a,c) \in A$

Nelle associazioni ad anello non simmetriche è necessario specificare, per ogni ramo dell'associazione, il relativo ruolo. L'importanza di evitare i ruoli diventa evidente per esprimere correttamente i vincoli di cardinalità delle associazioni.
Le associazioni ad anello possono essere ricorsive.



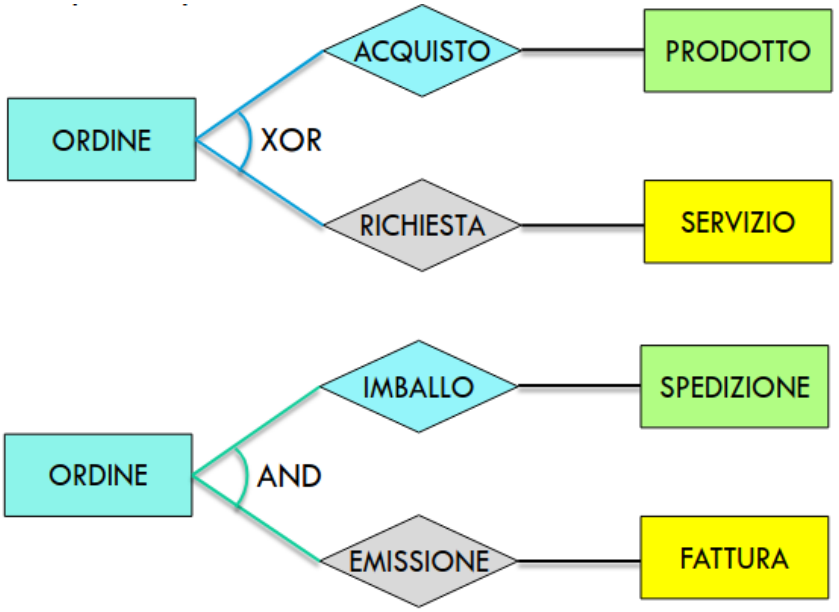
È possibile avere anelli anche in relazioni n-arie generiche (n > 2)



Associazioni XOR/AND

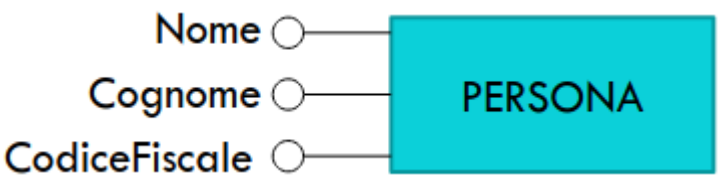
Non utilizzate nel corso

In alcune estensioni del modello E/R è possibile esprimere i vincoli sulla partecipazione delle entità alle diverse associazioni.



Attributi

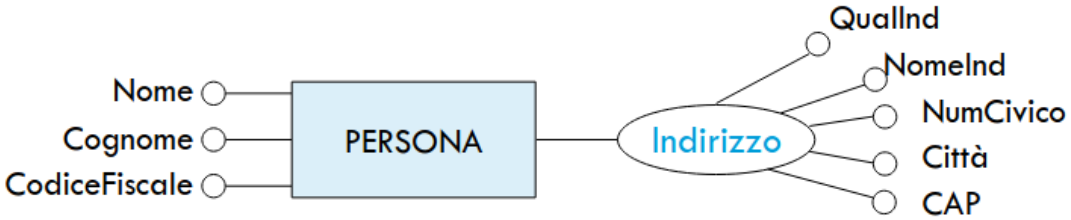
Un **attributo** è una **proprietà elementare** di un'istanza o di un'associazione. È denotato con un nome che deve essere univoco all'interno dell'entità o associazione a cui si riferisce.



Ogni attributo è definito su un **dominio di valori**. Associa a ogni istanza di entità un valore del corrispondente dominio.

Attributi composti

Sono attributi che si ottengono aggregando altri attributi, i quali prestano una forte affinità nel loro uso e significato. Un attributo non composto è detto **semplice**.



Esempio di attributo composto

Vincoli nel modello E/R

In ogni schema E/R sono presenti dei vincoli, alcuni sono **impliciti**, in quanto dipendono dalla semantica stessa dei costrutti del modello:

- **ogni istanza di un'associazione deve riferirsi a istanze di entità**
- **istanze diverse della stessa associazione devono riferirsi a differenti combinazioni di istanze delle entità partecipanti all'associazione**

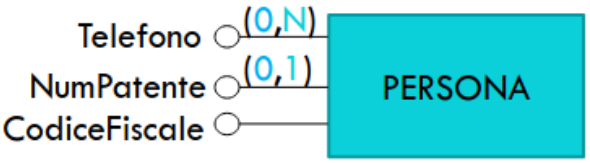
Altri vincoli sono **espliciti**, e sono definiti da chi progetta lo schema E/R sulla base della conoscenza della realtà che sta modellando:

- vincoli di cardinalità
- vincoli d'identificazione

Vincoli di cardinalità

Per un attributo è possibile specificare anche il **numero minimo** o **massimo** di valori che possono essere associati a un'istanza della corrispondente associazione o entità a cui l'attributo appartiene. Graficamente si può indicare la coppia (**min-card**, **max-card**) sulla linea che congiunge l'attributo all'entità o all'associazione, o di fianco al nome dell'attributo

- in assenza di default il valore è (1, 1)



Un attributo è detto:

- **opzionale:** se la cardinalità minima è 0
- **monovalore:** se la cardinalità massima è 1
- **multivalore:** se la cardinalità massima è N

Identificatori

Un vincolo d'identificazione per un'entità E definisce **identificatore** per E. Un identificatore ha lo scopo di permettere l'individuazione univoca delle istanze di un'entità. Deve valere anche la proprietà di **minimalità**: per ogni estensione ammissibile di un'entità nessun sottoinsieme proprio dell'identificatore deve essere a sua volta un identificatore.

- **identificatore interno:** si usano uno o più attributi di E
- **identificatore esterno:** si usano altre entità, collegate a E da associazioni, più eventuali attributi propri di E

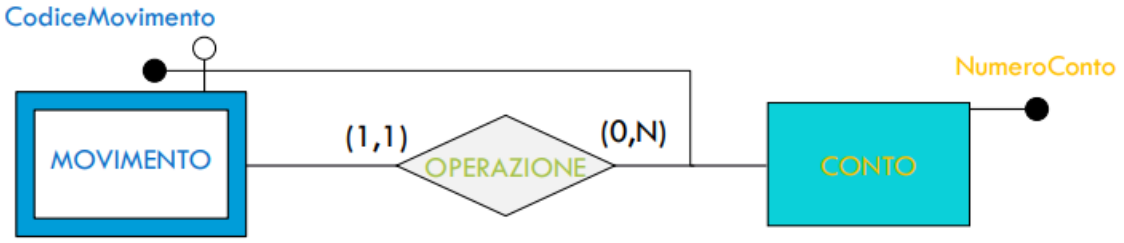
Se il numero di elementi (attributi o entità) che costituiscono un identificatore è pari a 1 si parla di **identificatore semplice**, altrimenti **composto**.
Se l'entità E è identificata esternamente attraverso l'associazione A, allora si ha min-card(E, A) = max-card(E, A) = 1;
Se basta E1, tramite A, a identificare E, allora max-card(E1, A) = 1, in caso contrario max-card(E1, A) = N;

Entità debole

Se un'entità ha solo identificatori esterni si dice che è un'entità **debole**, **forte** se ha solo identificatori interni. In alcune notazioni le entità deboli sono disegnate con una cornice.
Le entità deboli sono quelle entità che contengono istanze la cui presenza nella base dati è accettata solo se sono presenti determinate istanze di altre entità da cui queste **dipendono**.
Se l'esistenza dell'entità E2 dipende dall'esistenza dell'entità E1:

- E1 è detta **entità dominante** (o **proprietaria**)
- E2 è detta **entità debole** (o **subordinata**)

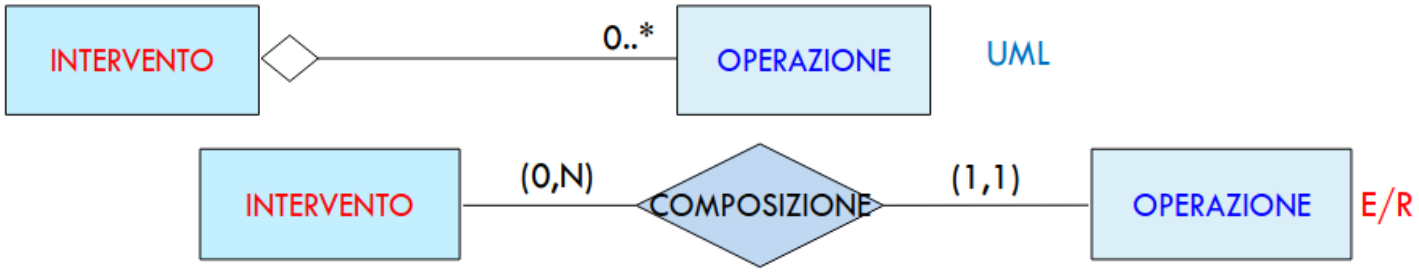
Nel caso di eliminazione dell'istanza di riferimento le istanze deboli devono essere eliminate; l'identificatore dell'entità debole deve contenere l'identificatore dell'entità da cui dipende.



Esempio di entità debole

Aggregazione

Nel modello E/R di base non vi è un costrutto particolare per l'aggregazione, astrazione che consente di rappresentare il concetto di **parte di (part of)**.

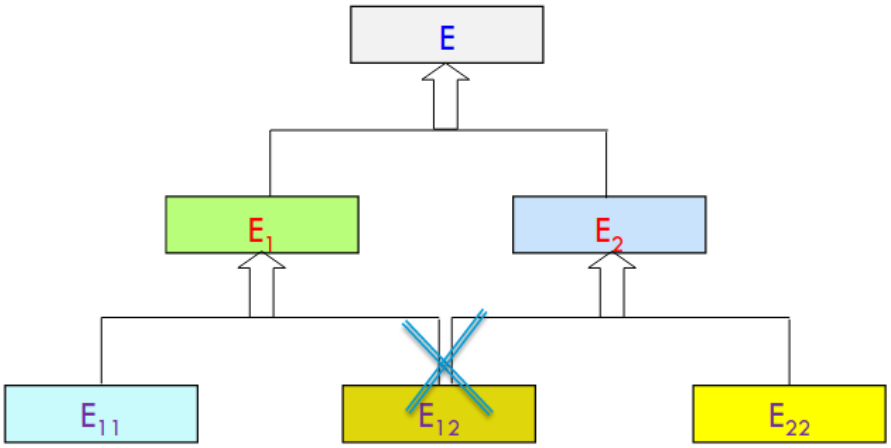


Generalizzazione

I concetti di generalizzazione e specializzazione sono stati già introdotti in precedenza, si rimanda pertanto alla lezione sui meccanismi d'astrazione. Pertanto è sufficiente sostituire al termine "classe" il termine "entità", a "sottoclasse" il termine "entità figlia" e a "superclasse" il termine "entità madre" o "genitore".

Ereditarietà singola

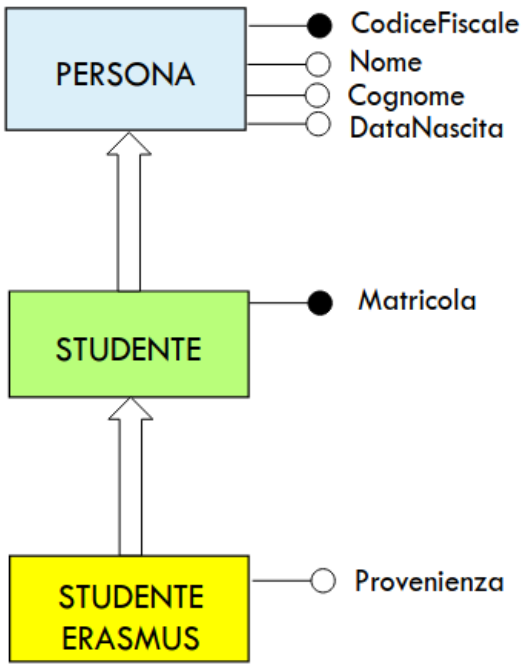
Una **gerarchia di generalizzazione** impone il vincolo che ciascuna sottoclasse abbia una sola superclasse (**ereditarietà singola**), alcune estensioni del modello prevedono ereditarietà multipla.



E12 non può essere figlia di entrambe le entità

Subset

Il **subset** è un caso particolare di gerarchia in cui si evidenzia una sola classe specializzata.



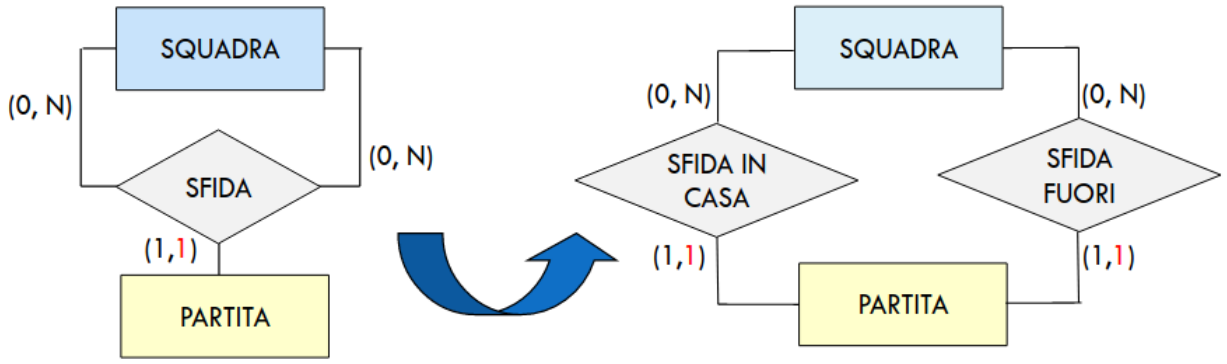
- ogni istanza Studente è anche un'istanza di PERSONA
- Studente eredita le proprietà di PERSONA e in più possiede una proprietà "Matricola"
- ogni istanza di Studente ERASMUS è un'istanza di STUDENTE
- Studente ERASMUS eredita le proprietà di STUDENTE e in più possiede una proprietà "Provenienza"
- non ha senso parlare di tipo di copertura

Associazioni ternarie: dipendenze funzionali

Quando in un'associazione ternaria esistono **dipendenze funzionali** tra le entità in gioco **è preferibile** sostituire la ternaria con associazioni binarie (**che modellano esplicitamente i vincoli del problema**).

Associazioni ternarie false

Se una o più entità partecipano **con cardinalità massima 1** a un'associazione ternaria siamo in presenza di una "**falsa ternaria**" che può essere sempre modellata, attraverso associazioni binarie.



Errori comuni negli schemi E/R

In tutti i casi visti con schemi errati si può affermare che il problema è originato da un'**analisi dei requisiti** poco accurata, che porta a soluzioni intuitive ma non adeguate. I nomi di entità e associazioni alle volte traggono in inganno: è bene quindi provare a ragionare anche a livello **estensionale** in termini di istanze e domandarsi che cosa "contiene" effettivamente un'entità o un'associazione.
Si deve porre molta attenzione ai vincoli d'integrità e, in particolare alle **dipendenze funzionali**.

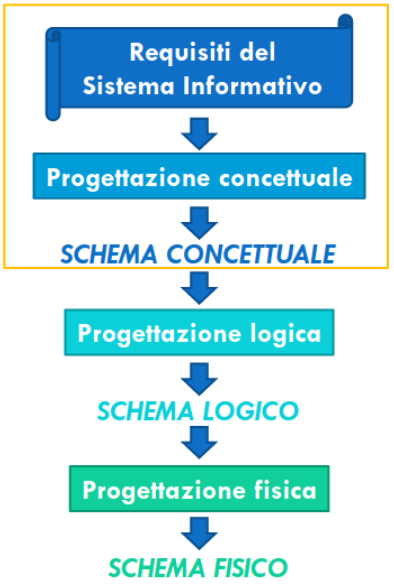
Utilità del modello E/R

Uno schema E/R è più espressivo di uno schema logico (relazionale), inoltre può essere impiegato con successo per altre attività, ad esempio:

- **documentazione**
- **reverse engineering:** a partire da un DB esistente, si può derivare una descrizione in termini E/R allo scopo di migliorare l'analisi del contesto applicativo ed eventualmente procedere a un'operazione di riprogettazione
- **integrazione di sistemi:** essendo indipendente dal modello logico dei dati, è possibile usare il modello E/R come "linguaggio comune" in cui rappresentare DB eterogenei

---

Progettazione concettuale



Fasi della progettazione di un DB

Raccolta dei requisiti

I requisiti devono essere **acquisiti**, le fonti possono essere molto diversificate tra loro:

- **utenti**, attraverso interviste o documenti di varia natura;
- **documentazione esistente**, normative, regolamenti interni, procedure aziendali e realizzazioni precedenti;
- **modulistica**

Attività non **standardizzabile**.

Interagire con gli utenti

È un'attività da considerare con molta attenzione, in quanto:

- **utenti diversi** possono fornire **informazioni diverse** con riferimento allo stesso tema
- utenti a livello più alto hanno spesso una **visione più ampia** ma **meno dettagliata**

In generale risulta spesso utile effettuare **verifiche** di comprensione e coerenza, richiedere **definizioni** e **classificazioni**, fa evidenziare gli **aspetti essenziali** rispetto a quelli marginali.

Requisiti: Documentazione Descrittiva

Regole generali

- scegliere il corretto **livello di astrazione**
- standardizzare la struttura delle frasi
- suddividere le frasi articolate
- separare le frasi sui **dati** da quelle sulle **funzioni**

Per evidenziare meglio i concetti che sono espressi nei requisiti è opportuno:

- costruire un **glossario dei termini**
- individuare **omonimi** e **sinonimi**
- rendere esplicito il **riferimento fra i termini**
- riorganizzare le frasi per i **concetti**

Glossario dei termini: esempio

Termine	Descrizione	Sinonimi	Collegamenti
Partecipante	Persona che partecipa ai corsi. Può essere un dipendente o un libero professionista.	Studente	Corso, Datore
Docente	Docente dei corsi. Può essere un collaboratore esterno.	Insegnante	Corso
Corso	Corso organizzato dalla società. Può avere più edizioni.	Insegnamento	Docente, Partecipante
Datore	Datore di lavoro attuale o passato di un partecipante ai corsi.	Posto di lavoro	Partecipante

Importante per eliminare al meglio le confusioni

Tabella delle operazioni

Le specifiche dei requisiti riguardano anche le operazioni da effettuare sui dati e la frequenza con cui devono essere eseguite. È opportuno utilizzare la stessa terminologia oer i concetti e far riferimento al glossario dei dati.

Operazione	Frequenza media (numero di volte al giorno)
Inserimento di un nuovo partecipante	40
Iscrizione di un partecipante a un'edizione di un corso	50
Inserimento di un nuovo docente con indicazione dei corsi per i quali può svolgere docenza	2
.....	...

Informazioni rilevanti per la fase di progettazione logica

Dai concetti allo schema E/R

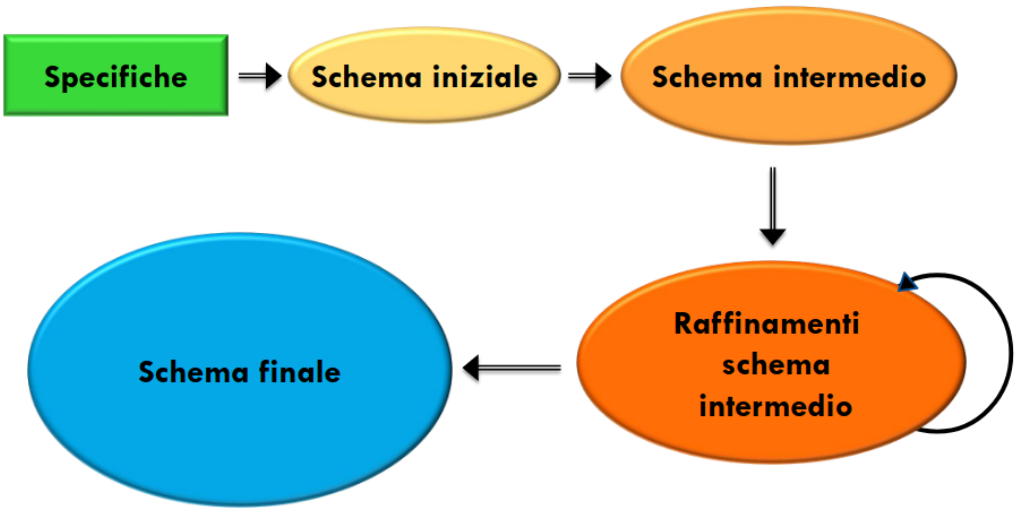
Un concetto sarà rappresentato come:

- **entità:** se ha proprietà significative e descrive oggetti con esistenza autonoma
- **attributo:** se è semplice (o composto) e non ha proprietà
- **associazione:** se correla due o più concetti
- **generalizzazione/specializzazione:** se generalizza altri concetti/se è una specializzazione di un altro concetto

Strategie di progettazione

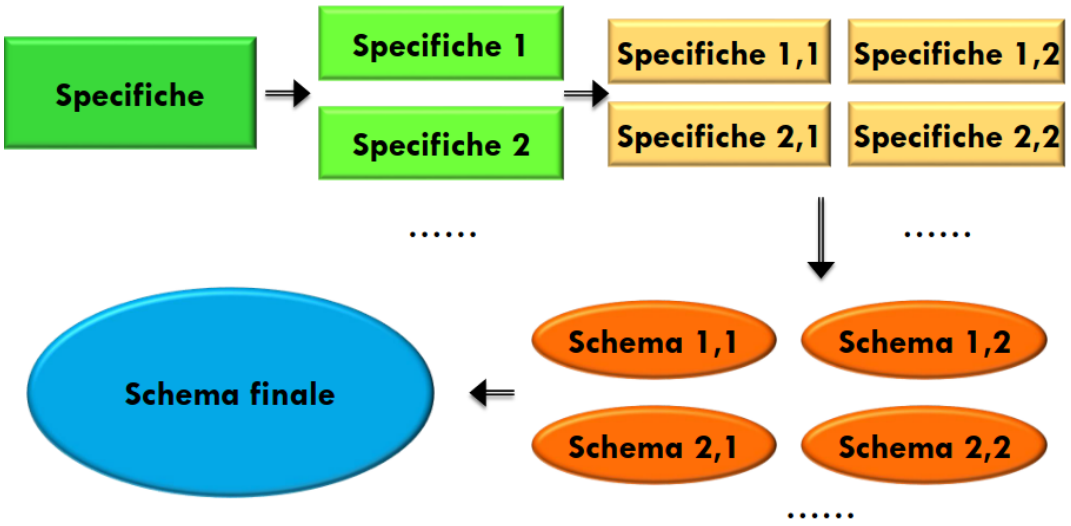
Strategia top-down

Si parte da uno schema iniziale molto astratto ma completo, che viene successivamente raffinato fino ad arrivare allo schema finale.
Inizialmente non è necessario specificare i dettagli, ma richiede una visione globale del problema, non sempre ottenibile.



Strategia bottom-up

Si suddividono le specifiche in modo da sviluppare semplici schemi parziali ma dettagliati, che poi vengono integrati tra loro.
Permette una ripartizione delle attività, ma richiede una fase d'integrazione.



Strategia inside-out

Lo schema si sviluppa a macchia d'olio, partendo dai concetti più importanti, aggiungendo quelli a essi correlati, è un caso particolare di strategia bottom-up, in cui il raffinamento inizia dai concetti principali per poi estendersi a quelli più lontani attraverso una navigazione delle specifiche.
Non richiede passi d'integrazione, ma richiede a ogni passi di esaminare tutte le specifiche per trovare i concetti non ancora rappresentati.

Approccio misto

Nella pratica si fa spesso uso di una **strategia ibrida:**

1. si individuano i **concetti principali** e si realizza uno **schema scheletro**
2. sulla base dello schema scheletro si può **decomporre**
3. successivamente si **raffina**, si **espande**, si **integra**

Qualità di uno schema concettuale

Lo schema E/R deve essere verificato accuratamente per accertarsi che risponda a requisiti di:

- **correttezza:** non devono essere presenti errori
- **completezza:** tutti i dati di interesse devono essere specificati
- **leggibilità:** riguarda anche aspetti prettamente estetici dello schema
- **minimalità:** è importante comprendere se esistono elementi ridondanti nello schema

Metodologia basata sulla strategia mista

- **Analisi dei requisiti**
  - analizzare i requisiti ed eliminare le ambiguità
  - costruire un glossario dei termini, raggruppare i requisiti
- **Passo base**
  - definire uno schema scheletro con i concetti più rilevanti
- **Passo di decomposizione**
  - decomporre i requisiti con riferimento ai concetti nello schema scheletro
- **Passo iterativo** (finchè non si è soddisfatti)
  - raffinare i concetti presenti sulla base delle loro specifiche
  - aggiungere concetti per descrivere specifiche non descritte
- **Passo di integrazione**
  - integrare i vari sottoschemi in uno schema complessivo, facendo riferimento allo schema scheletro
- **Analisi di qualità**
  - verificare le qualità dello schema e modificarlo

Il modello relazionale

Modello logico relazionale

Il modello relazionale è un **modello logico** nel senso che risponde al requisito di indipendenza dalla particolare rappresentazione dei dati adottata a livello fisico.
Nel contesto di un DB relazionale gli utenti che accedono ai dati e i programmatori che sviluppano applicazioni, fanno riferimento **solo al livello logico**, senza specificare i percorsi di accesso per eseguire le applicazioni.
Nel modello relazionale l'unica astrazione è il concetto di **relazione**; non vi sono costrutti concettuali di alto livello in grado di descrivere entità, associazioni, generalizzazioni, specializzazioni, aggregazioni.

Sul termine “relazione”

Il termine **relazione** può essere usato con diverse accezioni, che non devono essere confuse tra loro:

- nel linguaggio comune denota un legame di qualche tipo
- nella teoria degli insiemi denota una **relazione matematica**
- nel modello relazionale è una **generalizzazione della relazione matematica**
- nel modello E/R denota una **classe di legami fra entità**
- nei DBMS relazionali è usato spesso, come sinonimo di **tabella**

Relazione matematica

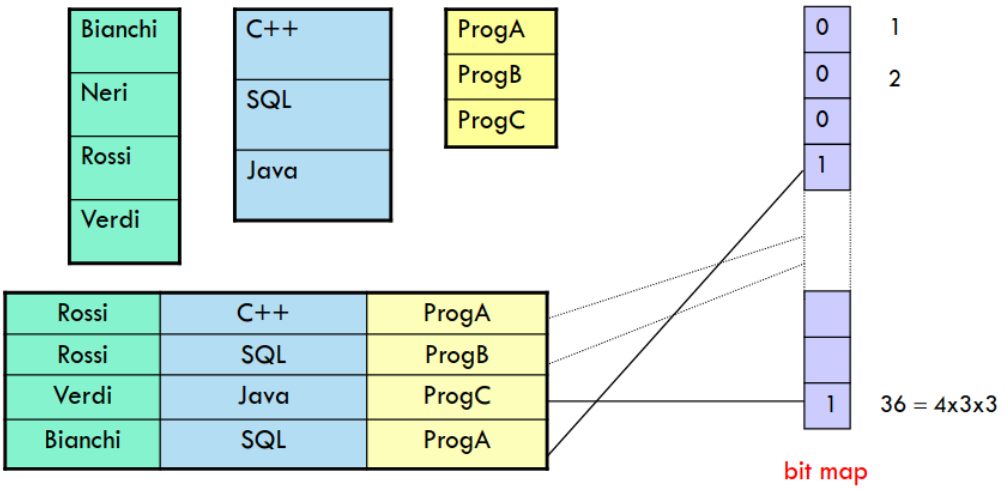
Una relazione è un insieme di n-ple: la definizione estesa di prodotto cartesiano contempla anche il caso di n = 1.

La definizione contempla anche:

- **relazioni con un numero infinito di n-ple** anche se ai fini pratici, a causa della dimensione finita della memoria, le relazioni sono necessariamente costituite da un numero finito di n-ple
- **domini finiti**, è utile per definire valori ammissibili anche se non presenti nella base di dati

Rappresentazione di relazioni

- **Rappresentazione insiemistica**, non adeguata per relazioni di grado n > 2 e/o con cardinalità superiore a qualche unità
- **Rappresentazione tabellare**, efficace e intuitiva
- **Rappresentazione multi-dimensionale**, adatta se n <= 3. Può essere utile per evidenziare alcune viste sui dati.
- **Bit map**



Relazione del modello relazionale

A ogni occorrenza di dominio si associa un nome univoco nella relazione, detto **attributo**, il cui compito è specificare il **ruolo** che quel dominio svolge nella relazione.
Nella **rappresentazione tabellare**, gli attributi sono le **intestazioni delle colonne**.

TeamCasa	TeamOspite	PuntiCasa	PuntiOspite
Enel Brindisi	Sidigas Avellino	92	88
MIA Cantù	Virtus Bologna	94	87
Fiat Torino	Vanoli Cremona	88	80
The Flex Pistoia	Consultinvest Pesaro	86	83

Definizione formale: relazione

Si indichi con dom(A) il dominio dell'attributo A e si consideri un insieme di attributi $X = \{A_1, \ldots, A_n\}$ una **tupla t** su X è una funzione che associa a ogni $A_i \in X$ un valore di $dom(A_i)$.
Uno **schema di relazione** su X è definito da un **nome** R e **dall'insieme di attributi** X, si indica con R(X).
Uno **stato o estensione** di relazione su X è un'insieme **r** di tuple su X, chiamato **relazione**.

attributi

PARTITE

TeamCasa	TeamOspite	PuntiCasa	PuntiOspite
Enel Brindisi	Sidigas Avellino	92	88
MIA Cantù	Virtus Bologna	94	87
Fiat Torino	Vanoli Cremona	88	80
The Flex Pistoia	Consultinvest Pesaro	86	83

Livelli intensionale ed estensionale

Uno schema R(X) definisce a livello intensionale una relazione; se è necessario a **livello estensionale**, per riferirsi a un generico stato di relazione con schema R(X), si usa semplicemente il nome dello schema in minuscolo, ovvero **r**.
A volte si usa la notazione r(X) per indicare una relazione su X.

Terminologia

Nella terminologia relazionale il termine **istanza** è sinonimo di **estensione** o **stato** di relazione e non di tupla.

Data Base relazionale

Lo schema di un **DB relazionale** è un'insieme di schemi di relazioni con nomi distinti:

$$R = \{R_1(X_1), R_2(X_2), ..., R_m(X_m)\}$$

Uno **stato** o **estensione** di un DB con schema $R = \{R_1(X_1), R_2(X_2), ..., R_m(X_m)\}$, è un insieme di stati di relazioni $r = \{r_1, r_2, \ldots, r_m\}$ con $\mathbf{r_i}$ stato di relazione su $R_i(X_i)$.

STUDENTI

Matricola	Cognome	Nome	DataNascita
29323	Bianchi	Giorgio	21/06/1978
35467	Rossi	Anna	13/04/1978
39654	Verdi	Marco	20/09/1979
42132	Neri	Lucia	15/02/1978

CORSI

CodCorso	Titolo	CodDocente	Anno
483	Analisi	0201	1
729	Analisi	0021	1
913	Sistemi Informativi	0123	2

ESAMI

Matricola	CodCorso	Voto	Lode
29323	483	28	no
39654	729	30	sì
29323	913	26	no
35467	913	30	sì

DOCENTI

CodDocente	Cognome	Nome	DataNascita
0021	Biondi	Carlo	21/06/1958
.....	.....	.....	.....

22

Esempio di modello relazionale

Vantaggi modello basato sui valori

Nella rappresentazione relazionale i **legami fra i dati non sono stabiliti con puntatori ma per mezzo dei valori dei domini che compaiono nelle tuple**.

- **Indipendenza dalle strutture fisiche** che possono cambiare anche dinamicamente
- Si rappresenta solo ci; che risulta **rilevante** dal punto di vista dell'applicazione utente
- **Maggiore portabilità** dei dati da un sistema all'altro
- I **puntatori sono direzionali** e pertanto stabiliscono un percorso di navigazione all'interno dei dati

Tabelle vs Relazioni

Tabella e **relazione** non sono sinonimi, una relazione del modello relazionale può essere vista come un particolare tipo di tabella.

Una **tabella** rappresenta una relazione se:

- i valori di ciascuna colonna sono tra loro omogenei (definiti sullo stesso dominio)
- le righe sono tra loro diverse
- le intestazioni delle colonne sono diverse tra loro

In una **tabella che rappresenta una relazione**:

- l'ordinamento delle righe è irrilevante
- l'ordinamento delle colonne è irrilevante

Create table

In SQL lo statement **CREATE TABLE** definisce al contempo gli attributi di ogni tupla e alloca un certo spazio per ospitare i record che saranno inseriti, si tratta di una **collezione di record**.

1NF, ovvero solo domini semplici

Il modello relazionale non permette di usare domini arbitrari per la definizione delle relazioni, in particolare **non è in generale possibile usare domini strutturati** (array, set, liste).
Una relazione in cui ogni dominio è **atomico** (non decomponibile), si dice che è in **Prima Forma Normale**, o **1NF** (First Normal Form). In molti casi è pertanto richiesta preliminarmente un'attività di **normalizzazione dei dati** che dia luogo a relazioni 1NF e che preservi l'informazione originale.

Considerazioni

In generale è bene ricordare che **ogni caso presenta una sua specificità e pertanto non deve essere trattato "automaticamente".**
Normalizzazione in 1NF è **un'attività di progettazione logica**, in quanto tale può essere solo oggetto di regole guida che però non hanno validità assoluta.

Valori nulli

La presenza di valori nulli non può essere sempre tollerata, ovvero è necessario imporre delle **restrizioni** al loro uso.
La presenza di un valore nullo non fornisce alcuna informazione sull'applicabilità o meno. È importante ricordare che **NULL** NON È UN VALORE DEL DOMINIO; tuttavia, **ai fini della verifica di assenza di tuple duplicate** è opportuno che i NULL siano considerati come gli altri valori e quindi uguali tra loro.

Vincoli di integrità

La **correttezza sintattica** di uno stato di una relazione non è condizione sufficiente affinché i dati rappresentino un'informazione possibile nel contesto reale considerato.
Un **vincolo di integrità** è una proprietà che deve essere soddisfatta da ogni possibile stato osservabile di una relazione; ogni vincolo può essere descritto da una funzione booleana che associa a ogni stato il valore VERO o FALSO.

Vincoli di dominio

Un vincolo che si riferisce ai valori ammissibili per un singolo attributo è detto **vincolo di dominio** (o sui **valori**).

Vincoli di tupla

I vincoli di dominio sono un caso particolare dei **vincoli di tupla**, ovvero vincoli che esprimono **condizioni su ciascuna tupla**, indipendentemente dalle altre.

ESAMI

Matricola	CodCorso	Voto	Lode
29323	483	28	no
39654	729	30	sì
29323	913	26	sì
35467	913	30	no

La Lode si può assegnare solo se il Voto è 30:

(Voto = 30) OR NOT(Lode = 'sì')

Vincoli di chiave

I vincoli di chiave vietano la presenza di tuple distinte che hanno lo stesso valore su uno o più attributi.
valori identificati **univocamente**

Chiavi e superchiavi

Dato uno schema R(X), un'insieme di attributi $K \subseteq X$ è:

- una **superchiave** se e solo se in ogni stato ammissibile r di R(X) non esistono due tuple distinte t1 e t2 tali che t1[K] = t2[K]
- una **chiave** se e solo se è una superchiave minimale, ovvero non esiste $K' \subset K$ con K' superchiave

Una **chiave** è pertanto un **identificatore minimale** per ogni r su R(X)

Importanza delle chiavi

L'esistenza delle chiavi garantisce l'accessibilità a ciascun dato del Db, in quanto ogni singolo valore è univocamente individuato da:

1. **nome della relazione:** individua una relazione del DB
2. **valore della chiave:** individua una tupla della relazione
3. **nome dell'attributo:** individua il valore desiderato

Chiave primaria

Per evitare i problemi è necessario scegliere una chiave, detta **chiave primaria** (**primary key**), su cui non si ammettono valori nulli. Si sottolineano gli attributi che costituiscono la chiave primaria nel modello relazionale.

Vincoli d'integrità referenziale

I vincoli sinora visti sono tutti di tipo **intra-relazionale**, in quanto interessano una relazione alla volta. Viceversa, **i vincoli di integrità referenziale** sono importanti tipi di vincoli **intra-relazionali** che enfatizzano come le correlazioni tra le tuple sono spesso ottenute usando i valori delle chiavi.
Un vincolo di integrità referenziale su Y impone che in ogni stato $r = \{r_1, r_2, \ldots\}$ del DB l'insieme dei valori di Y in r_2 sia un **sottoinsieme** dell'insieme dei valori della chiave primaria di $R_1(X_1)$ presenti nello stato r_1.
L'insieme Y viene detto una **foreign key** (o **chiave importata**).

STUDENTI

Matricola	Cognome	Nome	DataNascita
29323	Bianchi	Giorgio	21/06/1978
35467	Rossi	Anna	13/04/1978
39654	Verdi	Marco	20/09/1979
42132	Neri	Lucia	15/02/1978

CORSI

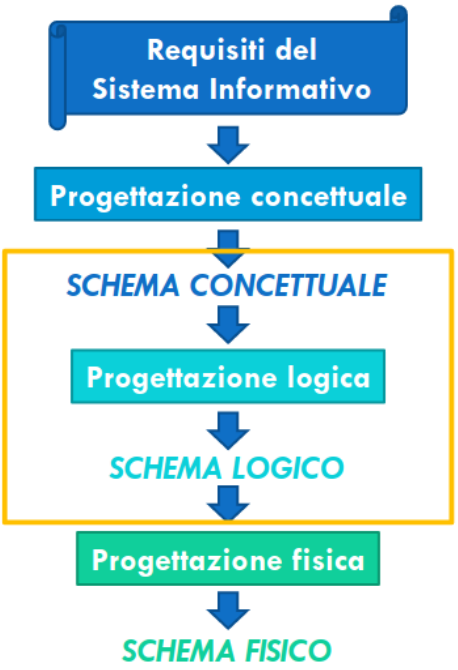
CodCorso	Titolo	CodDocente	Anno
483	Analisi	0201	1
729	Analisi	0021	1
913	Sistemi Informativi	0123	2

ESAMI (foreign key)

Matricola	CodCorso	Voto	Lode
29323	483	28	no
39654	729	30	sì
29323	913	26	no
35467	913	30	sì

In CORSI, {CodDocente} è una foreign key.
In ESAMI, {Matricola} è una foreign key, così come {CodCorso}.

Progettazione logica



Fasi della progettazione di un DB

Progettazione logica

L'obiettivo è quello di prevenire, a partire dallo schema concettuale, a uno schema logico che rappresenti **in modo fedele** i concetti e i requisiti analizzati e che sia allo stesso tempo **efficiente**.
L'**efficienza** è legata alle **prestazioni**, ma poiché queste non sono valutabili precisamente, né a livello concettuale né a livello logico, si ricorre all'impiego di **indicatori semplificati**.

Progettazione logica fedele

Fedeltà vuol dire che mediante $\mathbf{DB_{rel}}$ possiamo rappresentare esattamente le medesime informazioni documentate con lo schema $\mathbf{DB_{conc}}$. Più precisamente **fedeltà** significa che **i due schemi sono equivalenti dal punto di vista della loro capacità informativa**.
Il concetto di capacità informativa ha diverse definizioni, ma per i nostri scopi può essere considerato equivalente **all'insieme degli stati legali di uno schema**, indicato con **SL(DB)**, quindi:

$$\mathbf{DB_{rel}} \text{ e } \mathbf{DB_{conc}} \text{ sono equivalenti se } \mathbf{SL(DB_{conc}) = SL(DB_{rel})}$$

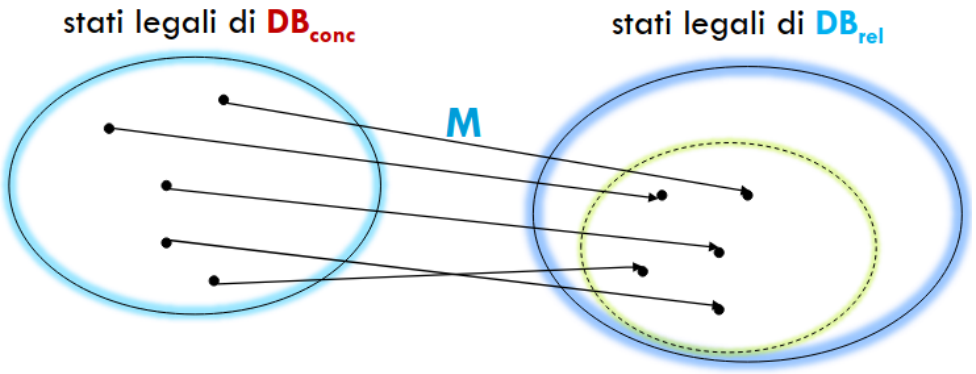
Progettazione che preserva l'informazione

La progettazione può essere vista come la definizione di un **mapping M** che spiega come trasformare ogni stato legale $\mathbf{db_{conc}}$ di $\mathbf{DB_{conc}}$ in un corrispondente stato $\mathbf{db_{rel}}$ di $\mathbf{DB_{rel}}$.
La progettazione **preserva l'informazione se M è totale e iniettiva**:

- **totale:** per ogni stato $\mathbf{db_{conc}}$ di $\mathbf{DB_{con}}$ esiste uno stato $\mathbf{db_{rel}}$ di $\mathbf{DB_{rel}}$ tale che $\mathbf{M(db_{conc}) = db_{rel}}$;
- **iniettiva:** non esistono due stati $\mathbf{db1_{conc}}$ e $\mathbf{db2_{conc}}$ tali che $\mathbf{M(db1_{conc}) = M(db2_{conc})}$.

Preservare l'informazione

La definizione intuitivamente asserisce che lo schema relazionale può contenere i dati dello schema E/R (**totalità**) e che si può ritornare indietro (**iniettività**).

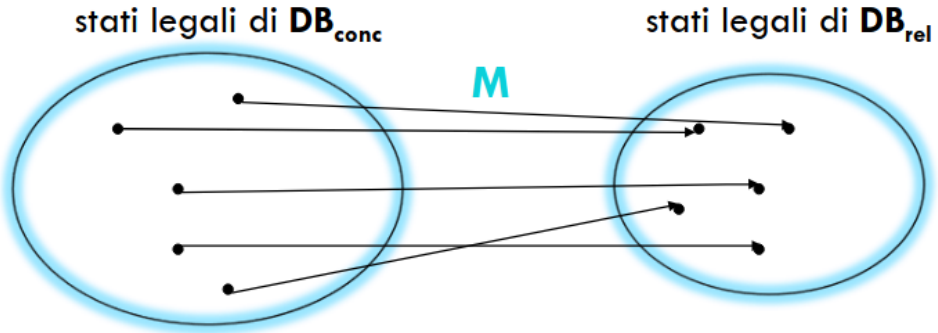


Progettazione che garantisce l'equivalenza

Diciamo che la progettazione **garantisce l'equivalenza** se:

- preserva l'informazione
- **per ogni stato legale db_{rel} di DB_{rel} esiste uno stato legale db_{conc} di DB_{conc} tale che $M(db_{conc}) = db_{rel}$**

La definizione intuitivamente asserisce che esiste una **bisezione** tra gli insiemi di stati legali.



Nella pratica…

La traduzione da schema E/R a schema relazionale avviene operando una **sequenza di trasformazioni/traduzioni semplici**, per ognuna delle quali è altrettanto semplice rispettare le regole che garantiscono l'equivalenza.

- **regole che preservano l'informazione (regole sulla struttura)**
- **regole aggiuntive che garantiscono l'equivalenza (regole sui vincoli)**

Fase della progettazione logica

La progettazione logica può essere articolata in due fasi principali:

- **Ristrutturazione:** eliminazione dallo schema E/R dei costrutti che non possono essere direttamente rappresentati nel modello logico target:
  - eliminazione degli **attributi multivalore**
  - eliminazione delle **gerarchie di generalizzazione**
  - **partizionamento/accorpamento** di entità e associazioni
  - scelta degli **indicatori principali**
- **Traduzione:** si mappano i costrutti residui in elementi del modello relazionale

Fase di ristrutturazione

Si pone l'obiettivo di **semplificare la traduzione** e **ottimizzare le prestazioni**; per confrontare tra loro diverse alternative bisogna conoscere il **carico di lavoro**, ovvero:

- le principali **operazioni** che la base dati dovrà supportare
- i **volumi dei dati** in gioco

Regola 80-20: il 20% delle operazioni produce l'80% del carico

Gli **indicatori** che deriviamo considerano due aspetti:

- **spazio:** numero di istanze (di entità e associazioni) previste
- **tempo:** numero di istanze visitate durante ogni operazione

Tavola dei volumi

Specifica il numero stimato di istanze per ogni entità E e associazione R dello schema, i valori sono **approssimati** MA **indicativi**.

Concetto	Costrutto	Volume
SEDE	E	10
DIPARTIMENTO	E	80
IMPIEGATO	E	2000
PROGETTO	E	500
COMPOSIZIONE	A	80
AFFERENZA	A	1900
DIREZIONE	A	80
PARTECIPAZIONE	A	6000

Esempio di tabella dei volumi

Descrizione delle operazioni

L'analisi delle operazioni principali richiede la codifica di:

- **tipo dell'operazione:** Interattiva (I) o Batch (B)
- **frequenza:** numero medio di esecuzioni in un certo periodo di tempo
- **schema di navigazione:** frammento dello schema E/R interessato dall'operazione sul quale viene evidenziato il **cammino logico** da percorrere per accedere alle informazioni di interesse

Per ogni operazione si costruisce una **tavola degli accessi** basata sullo schema di navigazione:

- il campo **costrutto** specifica il tipo di concetto (entità o associazione)
- nel campo **accessi** si conta il numero degli accessi
- il campo **tipo** è riferito al tipo di operazione: le operazioni di **scrittura** (**S**) sono più onerose rispetto a quelle di **letture** (**L**). **Il costo degli accessi in scrittura è in genere considerato doppio rispetto a quello delle letture**.

Analisi delle ridondanze

Una **ridondanza** in uno schema E/R è un'informazione significativa ma **derivabile da altre**. In questa fase si decide se eliminare o meno le ridondanze eventualmente presenti; è quindi comunque importante **averle individuate in fase di progettazione concettuale**.
Se si **mantiene** una ridondanza:

- si **semplificano** alcune interrogazioni
- si **appesantiscono gli aggiornamenti**
- si occupa **maggior spazio**

Le possibili ridondanze riguardano:

- **attributi derivabili** da altri attributi
- **associazioni derivabili** dalla composizione di altre associazioni

Eliminazione delle gerarchie

Il modello relazionale non può rappresentare direttamente le gerarchie di generalizzazione, entità e associazioni sono invece direttamente rappresentabili. Si eliminano perciò le gerarchie, sostituendo con entità e relazioni.

- accorpare le entità figlie nel genitore (**collasso verso l'alto**), **copertura:**
  - **totale esclusiva** -> tipo assume N valori quante sono le entità
  - **parziale esclusiva** -> Tipo assume N+1 valori (il valore in più è per le istanze che non appartengono a nessuno sotto-entità)
  - **sovrapposta** -> occorrono **tanti selettori quante sono le entità**.
- accorpare il genitore nelle entità figlie (**collasso verso il basso**)
  - **se la copertura non è completa il collasso verso il basso non si può applicare**
  - **se la copertura non è esclusiva introduce ridondanza**
- **sostituire la generalizzazione con associazioni**
  - tutte le entità vengono mantenute, le entità figlie sono in associazione binaria con l'entità padre e sono identificate esternamente
  - la sostituzioni con associazioni è **sempre possibile** indipendentemente dalla copertura

Quale scegliere?

Si consideri sia il numero degli accessi, sia l'occupazione di spazio:

- **Collasso verso l'alto:** conviene se gli accessi all'entità padre e alle entità figlie sono contestuali
- **Collasso verso il basso:** conviene se gli accessi alle entità figlie sono distinti, ma d'altra parte è possibile solo con generalizzazioni totali
- **Mantenimento di tutte le entità:** conviene se gli accessi alle entità figlie sono separati dagli accessi al padre

Partizionamenti e accorpamenti

È possibile ristrutturare lo schema accorpando o partizionando entità e associazioni, queste ristrutturazioni sono effettuate per rendere più efficienti le operazioni in base al principio già visto:

- **gli accessi si riducono:**
  - **separando gli attributi di un concetto** che vengono acceduti separatamente
  - **raggruppando attributi di concetti diversi** a cui si accede insieme

- i casi principali sono:
    - **partizionamento verticale** di entità, si separano gli attributi in gruppi omogenei anche usando associazioni
    - **partizionamento orizzontale** di associazioni (gestione **dati storici**)
    - eliminazione di **attributi multivalore,** si introduce una nuova entità le cui istanze sono identificate dai valori dell'attributo (uno a molti o molti a molti). Se è nota la **cardinalità massima di K** di un attributo multivalore è possibile prevedere **K attributi a singolo valore**
    - **accorpamenti di entità e associazioni**

Scelta degli identificatori principali

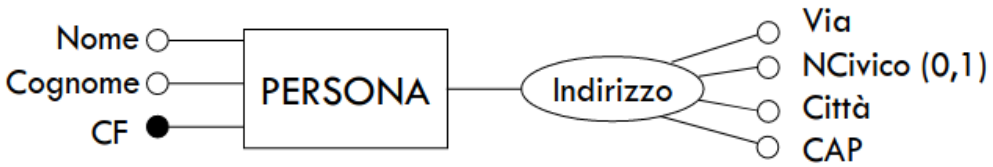
È un'operazione indispensabile per la traduzione nel modello relazionale, e corrisponde alla scelta della **chiave primaria**:

- **assenza di opzionalità**
- **semplicità**
- **utilizzo nelle operazioni più frequenti o importanti**

Se nessuno soddisfa i requisiti s'introducono nuovi attributi (**codici**) ad hoc.

Traduzione delle entità

- Ogni entità è tradotta con una relazione con gli stessi attributi
- La **chiave primaria** coincide con l'identificatore principale dell'entità
- Gli **attributi composti** vengono ricorsivamente suddivisi nelle loro componenti
- L'opzionalità e indicata dal simbolo asterisco (*)

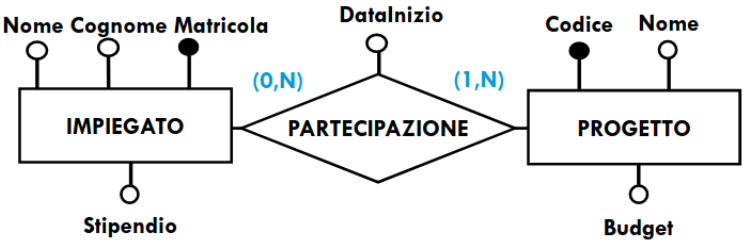


PERSONE(CF, Cognome, Nome, Via, NCivico*, Città, CAP)

Esempio di traduzione di entità

Traduzione delle associazioni

- Ogni associazione è tradotta con una relazione con gli stessi attributi, cui si aggiungono gli identificatori di tutte le entità che essa collega
- Gli **identificatori delle entità collegate** costituiscono una **superchiave**
- La **chiave** dipende dalle **cardinalità massime** delle entità nell'associazione
- Le **cardinalità minime** determinano la presenza o meno di **valori nulli**



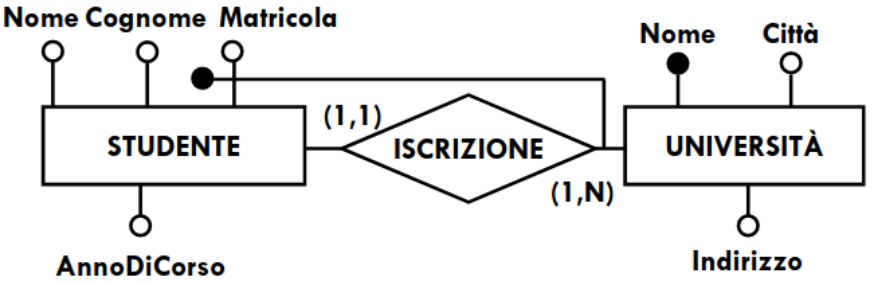
IMPIEGATI(Matricola, Nome, Cognome, Stipendio)

PROGETTI(Codice, Nome, Budget)

PARTECIPAZIONI(Matricola, Codice, DataInizio)
FK: Matricola REFERENCES Impiegati
FK: Codice REFERENCES Progetti

Entità con identificazione esterna

Nel caso di entità identificata esternamente, si importa l'identificatore della/e entità identificante/i, l'associazione relativa risulta automaticamente tradotta



STUDENTI(Matricola, Università, Cognome, Nome, AnnoDiCorso)
FK: Università REFERENCES Università
UNIVERSITÀ(Nome, Città, Indirizzo)

Nel caso generale, si possono avere **identificazioni esterne in cascata**, per operare correttamente occorre partire dalle entità **non identificate esternamente** e propagare gli identificatori che così si ottengono.

Osservazioni finali

La progettazione logica non deve essere condotta alla cieca; nel caso in cui vi siano varie alternative occorre valutare diversi fattori:

- la presenza o meno di valori nulli che dipende dal **volume dei dati**
- le porzioni di schema E/R interessate dalle varie **operazioni**
- la flessibilità degli schemi relazionali rispetto ad evoluzioni future

---

Normalizzazione

Forme normali

Una **forma normale** è una proprietà di uno schema relazionale che ne garantisce la qualità, cioè l'assenza di determinati difetti.
Una relazione normalizzata:

- presenta **ridondanze**
- si presta a provocare **anomalie di aggiornamento**

Le forme normali sono di solito definite sul modello relazionale, ma rivestono un ruolo importante anche in altri contesti.
L'attività che permette di trasformare schemi non normalizzati che soddisfano una forma normale è detta **normalizzazione**.

Ridondanza concettuale

Non vi sono replicazioni dello stesso dato, ma sono memorizzate informazioni che possono essere derivate da altre già contenute nel DB. Possono essere presenti anche negli schemi E/R

Ridondanza logica

Esistono duplicazioni sui dati che, oltre a comportare spreco di spazio di memoria, possono generare anomalie nelle operazioni sui dati.

- **Anomalia di aggiornamento**, se l'indirizzo di un ente cambia, è necessario modificare il valore in diverse tuple
- **Anomalia di inserimento**, se l'indirizzo di un ente cambia, è necessario modificare il valore in diverse tuple
- **Anomalia di cancellazione**, se si cancellano tutti i collaboratori afferenti a un ente, si perdono le informazioni dell'ente stesso

In un DB l'informazione può essere duplicata in modo:

- **NON RIDONDANTE**, se la duplicazione dei dati è necessaria, l'eliminazione delle duplicazioni comporta perdita di informazione
- **RIDONDANTE**, se la duplicazione dei dati non è necessaria, comporta spreco di memoria, è causa di possibili **anomalie e inconsistenze**

Le ridondanze logiche si possono eliminare mediante scomposizione degli schemi.

Summary

- **Ridondanza**, presenza di dati ripetuti in diverse tuple senza aggiungere informazioni significative
- **Anomalie di aggiornamento**, necessità di estendere l'aggiornamento di un dato a tutte le tuple in cui esso compare
- **Anomalia di cancellazione**, l'eliminazione di una tupla, può comportare l'eliminazione di dati che conservano la loro validità
- **Anomalia di inserimento**, l'inserimento di informazioni relative a uno solo dei concetti di pertinenza di una relazione è impossibile se non esiste un intero insieme di concetti in grado di costituire una tupla completa

Dipendenza funzionale

Per formalizzare i problemi visti si introduce un nuovo tipo di vincolo, la dipendenza funzionale (**FD**) tra attributi di una relazione.

- Si considerino:
  - uno schema di relazione R(T) e un'estensione r;
  - due sottoinsiemi (non vuoti) di T denominati X e Y rispettivamente.

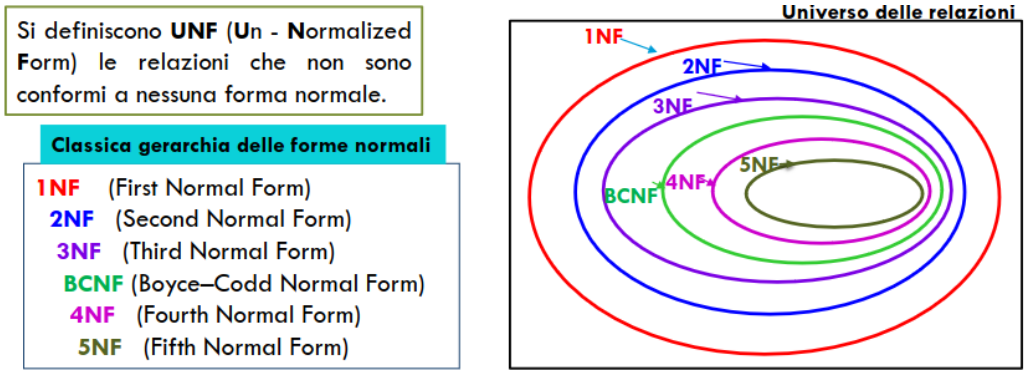
- Si dice che in r vale la dipendenza funzionale $X \rightarrow Y$ (X determina funzionalmente Y) se

$$\forall\, t1,t2 \in r : t1[X] = t2[X] \Rightarrow t1[Y] = t2[Y]$$

cioè per ogni coppia di tuple t1 e t2 di r con gli stessi valori su X, t1 e t2 hanno gli stessi valori anche su Y.

Forme normali

Si definiscono **UNF (Un-Normalized Form)** le relazioni che non sono conformi a nessuna forma normale.



First Normal Form (1NF)

Uno schema R(T) è in 1NF se e solo se **il dominio di ciascun attributo comprende solo valori atomici** e il valore di ciascun attributo in una tupla è un valore singolo del dominio di quell'attributo
La 1NF non permette relazioni dentro relazioni e relazioni come attributi di tuple. I soli valori di attributi ammissibili sono i singoli valori atomici rispetto al RDBMS.

- **non-scomponibilità:** non intende che il valore dell'attributo non possa essere suddiviso in sotto-parti. Quello che importa è che ogni valore dell'attributo sia dal punto di vista semantico **un'informazione unica**.

Second Normal Form (2NF)

Uno schema R(T) con vincoli F è in 2NF se e solo se **ogni attributo non-primo dipende completamente da ogni chiave candidata dallo schema**, ovvero se non c'è dipendenza parziale di un attributo non-primo da una chiave.
Uno schema in 1NF le cui chiavi sono tutte semplici, ovvero formate da un singolo attributo, è anche in 2NF.

Third Normal Form (3NF)

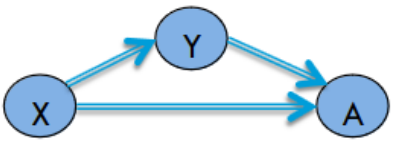
Uno schema R(T) con vincoli F è in 3NF se e solo se **ogni attributo non-primo non dipende transitivamente da nessuna chiave** ovvero se non cè dipendenza transitiva di un attributo non-primo da una chiave.

Dipendenza transitiva

Dato uno schema R(T):

A dipende transitivamente da X se esiste $Y \subset T$ tale che:

1. $X \rightarrow Y$ {X determina Y}
2. $\neg (Y \rightarrow X)$ {Y non determina X}
3. $Y \rightarrow A$ {Y determina A....}
4. $A \notin Y$ {...non banalmente}



Decomposizione senza perdita

La decomposizione non deve alterare il contenuto informativo del DB.
Uno schema R(X) si decompone senza perdita negli schemi $R_1(X_1)$ e $R_2(X_2)$ se, **per ogni stato legale r su R(X), il join naturale delle proiezioni di r su X_1 e X_2 è uguale a r stessa**

$$\pi_{X_1}(r) \bowtie \pi_{X_2}(r) = r$$

Per decomporre senza perdita è necessario e sufficiente che il join naturale sia eseguito su una superchiave di uno dei due sottoschemi, ovvero che valga:

$$X1 \cap X2 \rightarrow X1 \text{ oppure } X1 \cap X2 \rightarrow X2$$

Boyce-Codd Normal Form (BCNF)

BCNF è una forma normale più restrittiva di 3NF, essa estende le considerazioni sinora svolte anche agli attributi primi.
Uno schema R(T) con vincoli F è in BCNF se, **per ogni dipendenza funzionale (non banale) X -> Y definita su di esso, X è una superchiave di R(T)**.

Preservazione delle dipendenze

Si dice che una decomposizione **preserva le dipendenze** se ciascuna delle dipendenze funzionali dello schema originario coinvolge attributi che compaiono tutti insieme in uno degli schemi decomposti.
Se una FD non si preserva diventa più complicato capire quali sono le modifiche nel DB che violano la FD stessa.

Qualità di decomposizione

- **deve essere senza perdita**, per garantire la ricostruzione delle informazioni originarie
- **dovrebbe preservare le dipendenze**, per semplificare il mantenimento dei vincoli di integrità originari

È sempre opportuno normalizzare?

La normalizzazione non deve essere intesa come un obbligo; infatti in alcune situazioni le anomalie che si riscontrano in schemi non normalizzati sono un male minore rispetto alla situazione che si verrebbe a creare normalizzando

- **normalizzare elimina le anomalia ma può appesantire l'esecuzione di certe operazioni** (join tra gli schemi normalizzati)
- **la frequenza con cui i dati sono soggetti a modifica incide su qual è la scelta più opportuna** (relazioni quasi statiche danno un minor numero di problemi se non sono normalizzate)
- **la ridondanza presente in relazioni non normalizzate va qualificata** al fine di comprendere quanto possa incidere sull'occupazione di memoria e sui costi derivanti dall'aggiornamento di repliche di una stessa informazione

---

Algebra relazionale

L'algebra relazionale (AR) è costituita da un insieme di operatori di base che si applicano a una o più relazioni e che producono una relazione:

- **operatori di base unari:** selezione σ, proiezione π, ridenominazione ρ
- **operatori di base binari:** unione ∪, differenza —, join (naturale) ⋈
- altri operatori derivati possono essere definiti a partire da quelli di base

La **semantica** di ogni operatore si definisce specificando:

- come **lo schema** del risultato dipende dallo schema degli operandi
- come **lo stato della relazione** il risultato dipende dagli stati delle relazioni in ingresso

Selezione

L'operatore di selezione, σ, permette di selezionare un sottoinsieme delle tuple di una relazione, applicando a ciascuna di esse una formula booleana F.

Espressione:	$\sigma_F(R)$	
Schema	R(X)	X
Stato	r	$\sigma_F(r) = \{ t \mid t \in r \text{ AND } F(t) = \text{vero} \}$
	Input	Output

F si compone di predicati connessi da AND, OR e NOT.

Proiezione

L'operatore di proiezione, π, è ortogonale alla selezione, in quanto permette di selezionare un sottoinsieme di Y degli attributi di una relazione.

Espressione:	$\pi_Y(R)$	
Schema	R(X)	Y
Stato	r	$\pi_Y(r) = \{ t[Y] \mid t \in r \}$
	Input	Output

Cardinalità del risultato

In generale, la cardinalità di $\pi_Y(r)$ è minore o uguale della cardinalità di r (la proiezione elimina i duplicati), l'uguaglianza è garantita se e solo se Y è una superchiave di R(X).

Join naturale

L'operatore di join naturale, ⋈, combina le tuple di due relazioni sulla base dell'uguaglianza dei valori degli attributi comuni alle due relazioni, cioè quelli presenti in X1 ∩ X2.
Ogni tupla che compare nel risultato del join naturale di r1 e r2, estensioni rispettivamente di R1(X1) e R2(X2), è ottenuta come combinazione di una tupla di r1 con una tupla di r2 sulla base dell'uguaglianza dei valori degli attributi comuni.

	Espressione: $R_1 \bowtie R_2$	
Schema	$R_1(X_1), R_2(X_2)$	X_1X_2
Stati	r_1, r_2	$r_1 \bowtie r_2 = \{ t \mid t[X_1] \in r_1 \text{ AND } t[X_2] \in r_2 \}$
	Input	Output

Osservazioni

- Quando le due relazioni **hanno lo stesso schema** (X1 = X2) allora due tuple fanno match se e solo se hanno lo stesso valore per gli attributi, **ovvero sono identiche**, per cui: **se X1 = X2 il join naturale equivale all'intersezione delle due relazioni**.
- Se non vi sono attributi in comune (X1 ∩ X2 = ∅) allora, non essendovi condizioni di join, due tuple fanno sempre match, per cui: **se X1 ∩ X2 = ∅ il join naturale equivale al prodotto cartesiano**
- Poiché le relazioni sono insiemi, sono ben definite le operazioni di unione ∪, e di differenza —. Entrambi gli operatori si applicano a relazioni con lo stesso insieme di attributi.

ıttributi.	Espressione: $R_1 \cup R_2$	
Schema	$R_1(X), R_2(X)$	X
Stati	r_1, r_2	$r_1 \cup r_2 = \{ t \mid t \in r_1 \text{ OR } t \in r_2 \}$
	Input	Output

	Espressione: $R_1 - R_2$	
Schema	$R_1(X), R_2(X)$	X
Stati	r_1, r_2	$r_1 - r_2 = \{ t \mid t \in r_1 \text{ AND } t \notin r_2 \}$
	Input	Output

Intersezione

L'intersezione r1 ∩ r2 si può esprimere tramite l'operatore differenza: r1 ∩ r2 = r1 - (r1 - r2). È pertanto **un'operatore derivato**.

Prodotto cartesiano

La definizione di prodotto cartesiano assume che gli insiemi degli attributi di R1 e R2 siano **disgiunti**, cioè $X1 \cap X2 = \varnothing$, ed è dunque coincidente con la definizione data per il join naturale.

Espressione: $R_1 \times R_2$

Schema	$R_1(X_1), R_2(X_2)$ con $X_1 \cap X_2 = \varnothing$	X_1X_2
Stati	r_1, r_2	$r_1 \rhd\lhd r_2 = \{ t \mid t[X_1] \in r_1 \text{ AND } t[X_2] \in r_2 \}$
	Input	Output

Ridenominazione

L'operatore di ridenominazione, ρ, modifica lo schema di una relazione, cambiando i nomi di uno o più attributi. La definizione formale si omette per semplicità d'esposizione, è sufficiente ricordare:

- dato lo schema R(XZ), $\rho_{Y \leftarrow X}(R)$ cambia lo schema in YZ, lasciando invariati i valori delle tuple
- nel caso in cui si cambi il nome di più attributi, allora l'ordine in cui si elencano è significativo

Espressione: $\rho_{Y \leftarrow X}(R)$

Schema	R(XZ)	YZ
	Input	Output

Ridenominazione

X	Z

→

Y	Z

Self-join

La ridenominazione permette di eseguire in modo significativo il join di una relazione con sé stessa (self-join)

GENITORI

Genitore	Figlio
Luca	Anna
Maria	Anna
Giorgio	Luca
Silvia	Maria
Enzo	Maria

$\rho_{Nonno,Genitore \leftarrow Genitore,Figlio}(GENITORI)$

Nonno	Genitore
Luca	Anna
Maria	Anna
Giorgio	Luca
Silvia	Maria
Enzo	Maria

Divisione

La divisione, ÷, di r1 per r2, con r1 su R1(X1) e r2 su R2(X2), è il più grande insieme di tuple con uno schema X1, tale che, facendo il prodotto cartesiano con r2, ciò che si ottiene è una relazione contenuta in r1 o uguale a r1

Espressione: $R_1 \div R_2$

Schema	$R_1(X_1\, X_2), R_2(X_2)$	X_1
Stati	r_1, r_2	$r_1 \div r_2 = \{ t \mid \{t\} \rhd\lhd r_2 \subseteq r_1 \}$
	Input	Output

In modo equivalente si definisce $r_1 \div r_2 = \{ t \mid t \in \pi_{X_1}(r_1) \wedge \forall u \in r_2 (tu \in r_1) \}$

$R_1 \div R_2$ si può esprimere come: $\pi_{X_1}(R_1) - \pi_{X_1}((\pi_{X_1}(R_1) \rhd\lhd R_2) - R_1)$.

Theta-join

L'operatore theta-join, $\rhd\lhd_F$, è la combinazione di prodotto cartesiano e selezione con R_1 e R_2 senza attributi in comune e F formula composta di predicati di join, ossia del tipo x1 Θ x2.

- Se F è una congiunzione di uguaglianze, si parla di equi-join
- Il natural join può essere simulato per mezzo della ridenominazione, dell'equi-join e della proiezione
- Il theta-join e il join naturale sono detti **inner join**

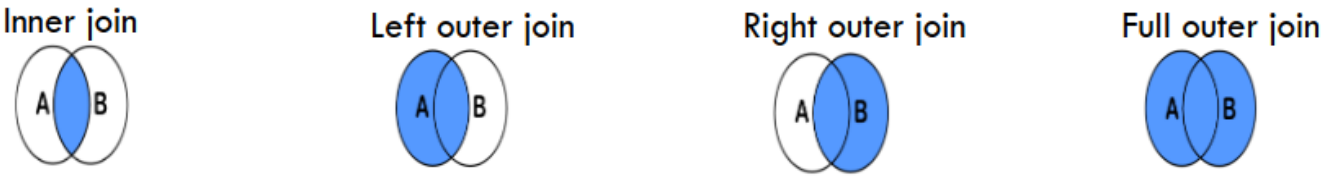
Semi-join

Il semi-join da S a R, indicato con $R \ltimes S$, è la proiezione del natural join $R \rhd\lhd S$ sugli attributi dello schema R, detto anche **left semi-join**.
Si definisce anche il **right semi-join** $S \rtimes R$ che equivale a $R \ltimes S$.
IL SEMI-JOIN NON È SIMMETRICO

Espressione:	$R \ltimes S$
Schema: R (X), S (Y)	X
Stati: r, s	$r \ltimes s = \pi_X(r \rhd\lhd s) = r \rhd\lhd \pi_{X \cap Y}(s)$
Input	Output

Outer-join

In alcuni casi è utile anche le **tuple dangling** di un join compaiono nel risultato, a tale scopo si introduce l'operatore **outer-join** (**external join**) che completa con valori nulli le tuple dangling.



Viste

Al fine di semplificare espressioni complesse è anche possibile fare uso di viste, ovvero espressioni a cui viene assegnato un nome e che è possibile riutilizzare all'interno di altre espressioni.
La sintassi è V := E dove V è il nome della vista ed E è l'espressione.

Equivalenza di espressioni

Push-down delle proiezioni

Usualmente un RDBMS cerca di eliminare quanto prima gli attributi che non servono per produrre il risultato di una query. un attributo A è utile se è richiesto in output o è necessario per un operatore che non è stato ancora eseguito.

Structured Query Language

SQL è il linguaggio standard de facto per DBMS relazionali, che riunisce in se funzionalità di:

- **DDL = Data Definition Language**
- **DML = Data Manipulation Language**
- **DCL = Data Control Language**

Nato come linguaggio **dichiarativo**, ovvero non specifica la sequenza di operazioni da compiere per ottenere il risultato. È un linguaggio **relazionalmente completo**, nel senso che per ogni espressione dell'algebra relazionale può essere tradotta in SQL.
Il modello dei dati di SQL è basato su **tabelle anziché relazioni**:

- possono essere presenti righe duplicate
- in alcuni casi l'ordine delle colonne ha rilevanza

Data Definition Language (DDL)

Il DDL di SQL permette di definire schemi di relazioni, modificarli o eliminarli; permette inoltre di specificare i **vincoli**, sia a livello di tupla sia a livello di tabella.
Permette di definire nuovi **domini**, oltre a quelli predefiniti; inoltre si possono definire le **viste**, ovvero tabelle virtuali, e **indici**, per accedere efficientemente ai dati.

I domini

In SQL sono utilizzabili 2 tipi di domini:

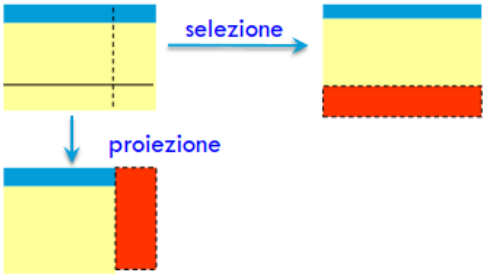
- **Domini elementari**
  - carattere
  - bit
  - numerici
  - data, ora, intervalli di tempo
- **Domini definiti dall'utente ->** utilizzabili in definizioni di relazioni, anche con vincoli e valori di default.

```
CREATE DOMAIN Voto AS SMALLINT
DEFAULT NULL
CHECK ( value >=18 AND value <= 30 )
```

Selezione e proiezione in SQL

Sono due operazioni ortogonali:

- **selezione:** realizza una decomposizione orizzontale includendo nel risultato solo le ennuple che soddisfano i requisiti; produce un risultato che contiene tutti gli attributi dell'operando e solo alcune ennuple dell'operando
- **proiezione**: realizza una decomposizione verticale includendo nel risultato solo gli attributi richiesti; produce un risultato che contiene solo una parte degli attributi dell'operando e contiene un numero di ennuple pari quelle dell'operando



Data Manipulation Language (DML)

Le istruzioni principali del DML sono:

- **SELECT**
- **INSERT** (può usare il risultato di una query per eseguire inserimenti multipli)
- **DELETE**
- **UPDATE**

DELETE e UPDATE possono fare uso di condizioni per specificare le tuple da cancellare o modificare.

Definizione di Viste

Mediante l'istruzione CREATE VIEW si definisce una **vista**, ovvero una **tabella virtuale**; le **tuple della vista** sono il risultato di una query che viene valutata dinamicamente ogni volta che si fa riferimento ad una vista.

Le viste possono essere create per vari scopi:

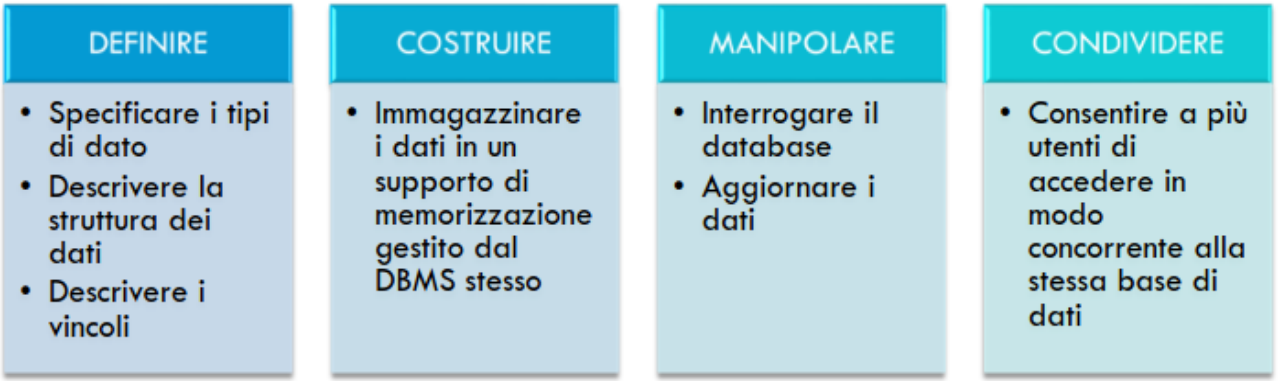
- permettere agli utenti di avere una **visione personalizzata del DB** che in parte astragga dalla struttura logica del DB stesso
- fa fronte a **modifiche dello schema logico** che comporterebbero una ricompilazione dei programmi applicativi
- **semplificare la scrittura di query complesse**

Possono essere anche usate come **meccanismo per il controllo degli accessi**, fornendo a ogni classe di utenti gli opportuni privilegi.

Funzionalità DBMS

Il DBMS

Un DataBase Management System (DBMS) è un insieme di programmi che permettono agli utenti di definire, costruire, manipolare e condividere una base di dati.



Le principali funzionalità di un DBMS sono:

- supporto per almeno un **modello dei dati**
- uso di **cataloghi** per memorizzare la descrizione di DB
- supporto di viste multiple sui dati condivisi tra più utenti
- **indipendenza** tra programmi e dati, e tra programmi e operazioni
- **gestione efficiente ed efficace** di grandi quantità di dati **persistenti e condivisi**
- **linguaggi di alto livello** per la definizione dei fati, l'interazione con il DB, e l'amministrazione e il controllo dei dati
- **funzionalità di supporto** per semplificare la descrizione delle informazioni, lo sviluppo delle applicazioni, l'amministrazione di un DB, le interfacce ecc.

Modello logico dei dati

L'**astrazione logica** con cui i dati sono resi disponibili all'utente definisce un **modello dei dati**. Un **modello dei dati** è una collezione di concetti utilizzati per descrivere i dati, le loro associazioni, e i vincoli che questi devono rispettare.
Un ruolo di primaria importanza nella definizione di un modello dei dati è svolto dai **meccanismi che possono essere usati per strutturare i dati**.

Natura autodescrittiva di una base di dati

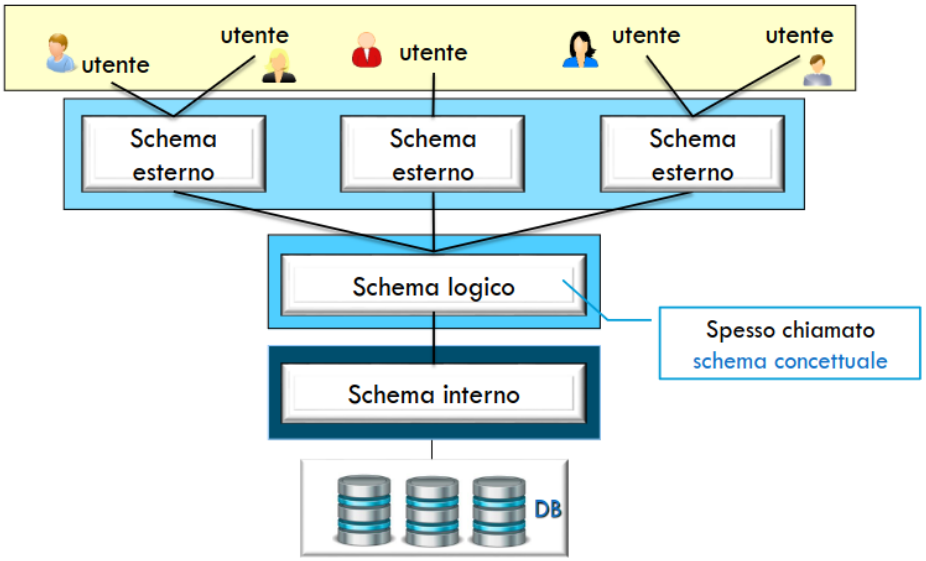
Il sistema di basi di dati contiene non solo la base di dati, ma anche una definizione o **descrizione completa della sua struttura** e dei suoi vincoli. Tale definizione è memorizzata nel **catalogo** del sistema, che contiene informazioni come la **struttura** di ciascun file, il **tipo** e il **formato** di memorizzazione di ogni dato e vari **vincoli** sui dati.
Le informazioni memorizzate nel catalogo sono dette **metadati**.

Indipendenza fisica e logica

Un' importante obiettivo di un DBMS consiste nel fornire caratteristiche di **indipendenza logica** e **indipendenza fisica**.

- **indipendenza fisica:** l'organizzazione fisica dei dati dipende da considerazioni legate all'efficienza delle strutture dati adottate; la riorganizzazione fisica dei dati, o la creazione di struttura d'accesso aggiuntive, non deve comportare modifiche allo schema logico del DB, lasciando inalterate anche le particolari viste d'utente, né deve causare effetti collaterali sui programmi applicativi.
- **indipendenza logica:** pur in presenza di uno schema logico integrato non è spesso utile o conveniente che ogni utente ne abbia una visione uniforme; è desiderabile che modifiche allo schema logico non comportino aggiornamenti degli schemi esterni e delle applicazioni.

Architetture a tre livelli



Il livello fisico (o interno)

Il DB fisico consiste di una serie di file, residenti su dispositivi di memoria permanenti che contengono dati, indici e altre tipologie di strutture. Lo **schema fisico** descrive come il DB, definito a livello logico è rappresentato a livello fisico.
La gestione del DB fisico è a carico del **DBA** (**DataBase Administrator**) e non degli utenti, i quali possono concentrarsi su aspetti di più alto livello.

Livello delle viste (o esterno)

Il **livello esterno** è costruito a partire dallo schema logico integrato mediante la definizione di viste ad hoc che descrivono parte dello schema logico secondo le esigenze di operatività dei diversi utenti.
La distinzione tra livello esterno e logico può, in molti casi, risultare trasparente agli utenti, che, ad esempio, in un RDBMS “vedono” semplicemente un insieme di tabelle.

Utilità delle viste

Oltre a fornire una personalizzata visione del DB, le viste possono svolgere un ruolo importante anche per altri motivi:

- una **ristrutturazione dello schema integrato** può, in alcuni casi, essere mascherata facendo uso di viste
- mediante le viste è possibile regolare meglio il **controllo degli accessi** al DB, ad es. mascherano dati riservati
- le viste possono essere usate per **calcolare dinamicamente** nuovi dati a partire da quelli memorizzati nel DB, senza per questo introdurre ridondanza

Condivisione: regolamentare gli accessi

Gli utenti di un DB sono naturalmente classificabili in diverse tipologie, a cui vanno pertanto associate **autorizzazioni** distinte, ad esempio:

- uno **studente** può leggere i propri dati, ma non quelli di altri studenti;
- un **docente** può leggere i dati dei soli studenti del proprio corso, non può modificare l'elenco degli esami già sostenuti da uno studente, ma può registrare esami del proprio corso;
- la **segreteria studenti** può leggere i dati di tutti e può registrare nuovi studenti

La gestione delle autorizzazioni può essere complessa, per questo motivo sono previste specifiche figure di **DataBase Administrator** che conferiscono agli utenti i "giusti" privilegi.

Concorrenza e protezione da guasti

Un DBMS deve garantire che gli accessi ai dati, da parte di diverse applicazioni, non interferiscono tra loro violando vincoli d'integrità e alterando la consistenza del DB.
A tale scopo è pertanto necessario far ricorso a opportuni meccanismi di **controllo della concorrenza**.
Quando un'applicazione deve effettuare più operazioni di modifica, è possibile che per qualche motivo solo una parte di queste sia effettivamente eseguita.
In questo caso, per garantire l'integrità e la consistenza dei dati, il DBMS deve provvedere ad **annullare** tali modifiche.

RDBMS, proprietà ACID