

I sistemi informativi aziendali: l'importanza delle informazioni

di Roberta Molinari

Dati e Informazioni

- ▶ Il termine **dato** significa letteralmente "*fatto*".
- ▶ I dati sono una rappresentazione dei fatti.
- ▶ Un dato costituisce un'informazione se fornisce una nuova conoscenza attraverso una chiave di interpretazione
- ▶ **L'informazione** è l'incremento di conoscenza che può essere acquisita dai dati.

es. il dato 12, senza chiave di interpretazione, non costituisce informazione. Con la chiave di interpretazione "Quantità disponibile" assume l'informazione di "quantità disponibile per un determinato articolo"

Le Informazioni

- ▶ In ogni sistema di vita dell'uomo vengono trattate **informazioni**.
- ▶ Costituiscono un grande patrimonio, sono considerate risorse preziosissime grazie alle quali lo stesso sistema umano sopravvive.
- ▶ Individuate e raccolte devono essere memorizzate in modo che si possano facilmente eseguire le operazioni **CRUD**:
 1. *Create* - **AGGIUNGERE**
 2. *Read* - **RECUPERARE**
 3. *Update* - **MODIFICARE**
 4. *Delete* - **CANCELLARE**

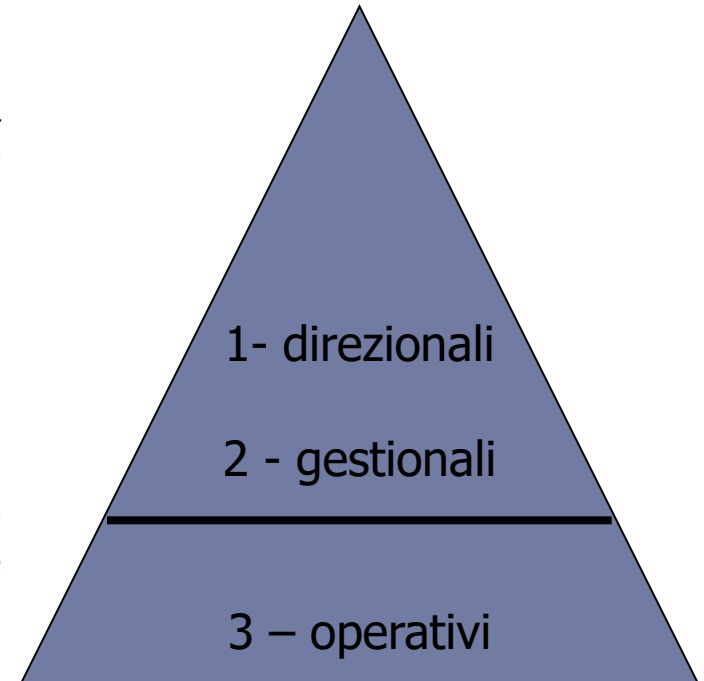
- ▶ Ogni impresa si organizza in base:
 - ▶ alla sua **missione** o **mission** (lo scopo per cui è nata: produrre scarpe)
 - ▶ ai suoi **obiettivi generali** o **target** a breve o a lungo termine (aumentare il fatturato)
- ▶ Al suo interno si possono individuare:
 - ▶ **Risorse**: tutto ciò con cui opera (materiale, immateriale, persone, interne o esterne)
 - ▶ **Processi**: insieme di attività (decisioni e azioni) svolte per il raggiungimento di mission e target tramite un risultato definito e misurabile (prodotto)

Classificazioni dei processi

Piramide di Anthony

Processi organizzativi

1. concorrono alla definizione degli obiettivi strategici Definiscono i piani a medio e lungo termine, progettano l'organizzazione dell'intera azienda (filiali) (decisioni strategiche)
2. concorrono alla traduzione degli obiettivi in criteri di gestione-programmazione ed effettuano il controllo del raggiungimento di tali obiettivi Controllano il corretto utilizzo delle risorse e definiscono piani a breve e medio termine (raggiungimento di un certo budget) (decisioni tattiche)



Processi operativi

3. concorrono all'attuazione concreta degli obiettivi (produzione delle scarpe)

Dati- Informazioni - Conoscenza

- ▶ Un caso molto particolare di risorsa su cui operano tutte le aziende è l'informazione. L'informazione è infatti una risorsa che riguarda tutte le altre risorse.
- ▶ I **dati** sono una materia prima in continua crescita
- ▶ Le **informazioni** sono il valore aggiunto ai dati
- ▶ Possedere la **conoscenza** è un'esigenza fondamentale per poter prendere decisioni



Sistema informativo aziendale (S.I.)

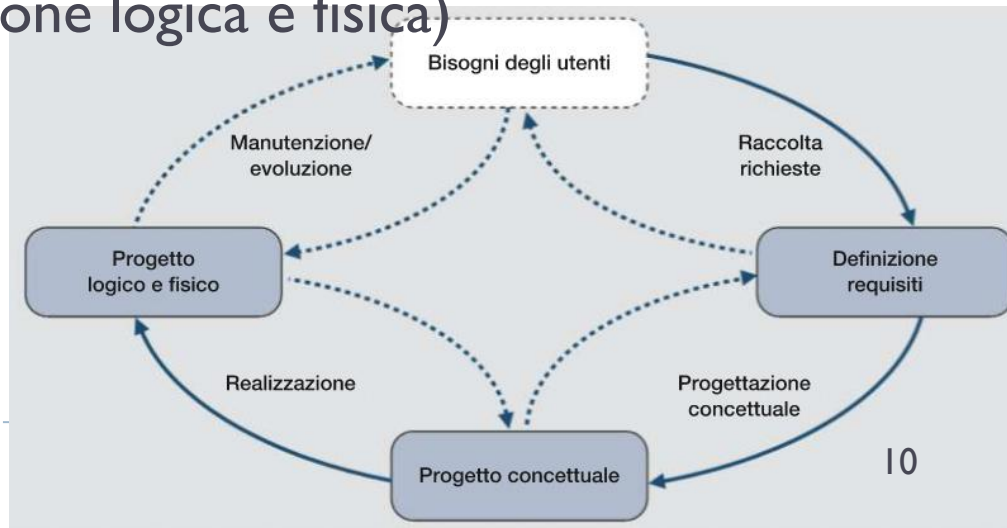
- ▶ Il **sistema informativo SI** è l'insieme delle persone, dei mezzi e delle procedure che riguardano la *raccolta*, la *produzione*, l'*archiviazione*, l'*elaborazione*, la *distribuzione* dei dati al fine di ottenere informazioni che servono da supporto alle funzioni operative, ai processi decisionali e al controllo di una organizzazione.

Sistema informatico aziendale

- ▶ Quella parte del sistema informativo in cui le informazioni sono raccolte, elaborate, archiviate, scambiate mediante l'uso dell'**ICT** (**I**nformation & **C**ommunication **T**echnology, sono le tecnologie della informazione e della comunicazione) costituisce il **sistema informatico**, anche chiamato **EDP** (**E**lectronic **D**ata **P**rocessing).
- ▶ È il sottoinsieme del sistema informativo formato da una componente:
 - ▶ **Software**: archivi e programmi di gestione (applicazioni)
 - ▶ **Hardware**: supporti fisici, computer, rete, infrastruttura,...

Ciclo di vita del sistema informatico

- L'informatizzazione di un SI deve essere occasione di razionalizzazione delle attività per renderle più efficaci e efficienti.
- Ci sono metodologie differenti, ma in generale è un processo ciclico e sono previste tre fasi
 - Raccolta delle richieste degli utenti
 - Progettazione concettuale
 - Realizzazione (progettazione logica e fisica)



Le basi di dati

Gli archivi

Gli **archivi** servono per conservare i dati in modo permanente, per poter essere reperiti e utilizzati in seguito.

Gli archivi sono un insieme di dati i cui elementi hanno le seguenti caratteristiche:

- sono legati tra loro da un *nesso logico* (si riferiscono ad uno stesso argomento Es. rubrica telefonica)
- sono rappresentati secondo un certo *formato* per permetterne l'interpretazione (Es. nominativo, indirizzo, tel)
- sono registrati in modo permanente su un certo *supporto* su cui è possibile leggere e scrivere (Es. la rubrica cartacea)
- sono *organizzati* per permetterne la consultazione (Es. ci sono le etichette delle lettere)

Se il supporto è di tipo informatico si parla di **Archivi elettronici** o **File di dati**

Gli archivi

Record logici

I dati, in generale, sono raggruppati in unità logiche ognuna riferita ad un singolo soggetto della “realtà” memorizzata.

Ogni unità è un **record logico**, che a sua volta è suddiviso in **campi**, i cui valori caratterizzano il soggetto. L'elenco dei campi e il relativo tipo costituiscono il **tracciato record**.

Es. di tracciato record per la rubrica telefonica

Cognome-Nome	Indirizzo	Numero Telefono
Alfabetico	Alfanumerico	Numerico

Es. di record

Rossi Mario	Via Roma 1	5556346
-------------	------------	---------

Gli archivi

Record logici

Campo: spazio riservato per memorizzare il dato relativo ad un'informazione di senso compiuto (es. indirizzo). Singolo dato

Record: insieme di campi correlati tra loro formanti un'unità informativa più ampia (es. persona).

Insieme di dati di uno stesso soggetto



Nome	Indirizzo	Telefono
Mario Rossi	Via Roma 1	0171-66666
Bianchi Luisa	P.zza Italia 14	011-999999
Verdi Ugo	Via Po 12	011-444444

Archivio: insieme di record omogenei (es. rubrica).

Insieme di dati di un archivio di più soggetti

Gli archivi

Operazioni

- **Creazione dell'archivio:** si definisce il supporto, il tracciato record, il nome e l'organizzazione
- **Manipolazione dei dati:**
 - *Inserimento*
 - *Modifica o aggiornamento*
 - *Cancellazione*
- **Interrogazione o consultazione dei dati:** reperimento di informazioni
- **Distruzione**

Archivi

Limiti archivi tradizionali file-based

- ▶ I programmi sono legati al linguaggio utilizzato: modifiche della struttura record comportano modifiche ai programmi che la utilizzano
- ▶ L'accesso ai dati è determinato dal tipo di organizzazione degli archivi (sequenziale, diretto) e si devono comunque leggere record per record
- ▶ Alcuni dati si presentano più volte nello stesso file o in file differenti
- ▶ L'accesso ai dati avviene solo tramite l'applicazione. Nuove interrogazioni richiedono la modifica delle applicazioni

Archivi

Limiti archivi tradizionali file-based

- ▶ In una gestione tradizionale ogni applicazione opera sui suoi dati la cui struttura è definita all'interno del programma stesso. È complicata la condivisione di dati da parte di più applicazioni.
- ▶ Se i dati sono utilizzati da più applicazioni, spesso vengono duplicati, con spreco di memoria e rischio di inconsistenza dei dati

Archivi

Limiti archivi tradizionali file-based

Ordini							
CodOrd	Codacc	Descr	Prezzo	Qt	Codcli	Nome	Indirizzo
01	M03	Batteria	100,00	3	010	Rossi Mario	Via Roma 1
01	M12	Antenna	25,00	1	010	Rossi Mario	Via Roma 1
02	M03	Batteria	100,00	2	020	Verdi Luca	C.so Italia 10

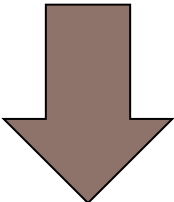
Archivi

Limiti archivi tradizionali file-based

Ridondanza

- 
- ▶ Gli stessi dati appaiono in più punti

Incongruenza

- 
- ▶ Conseguenza della ridondanza in caso di modifiche parziali delle occorrenze dei dati ripetuti

Inconsistenza

- ▶ Conseguenza della incongruenza: i dati non sono più affidabili

Database

Definizione

I **database** nascono per superare i limiti degli archivi tradizionali.

Sono una raccolta di dati (archivi) organizzati per essere usati in modo efficiente da differenti applicazioni e da utenti diversi. (è sicuramente su supporto informatico)

Nei database è presente sia la definizione della struttura dei dati (tracciato record: numero nome e tipo) che i dati stessi. La struttura non fa più parte dell'applicazione, è ora indipendente da essa.

Database

Teoria delle basi di dati

- ▶ Nell'informatica la **teoria delle basi di dati** studia come organizzare al meglio grandi quantità di informazioni, per poterle gestire in modo:
 - ▶ **Semplice**: per utenti e applicazioni
 - ▶ **Efficiente**: in tempo e spazio
 - ▶ **Efficace**: rappresentano realmente la realtà che si vuole gestire
 - ▶ **Sicuro**: da utenti non autorizzati
 - ▶ **Solido**: resistente a guasti o errori accidentali degli operatori
 - ▶ **Condiviso**: permettere l'accesso simultaneo

Database

Caratteristiche

I **database** garantiscono:

- ▶ **Indipendenza dalla struttura fisica dei dati:** si possono modificare i supporti senza modificare le applicazioni
- ▶ **Indipendenza dalla struttura logica dei dati:** si possono modificare le definizioni delle strutture senza modificare le applicazioni
- ▶ **Condivisione:** utilizzo da parte di più utenti o più applicazioni, è consentito accesso concorrente e viste parziali

Database

Caratteristiche

- ▶ **Eliminazione della ridondanza:** non si devono duplicare dati perché gli archivi sono integrati
- ▶ **Sicurezza dei dati:** protezione da accessi non autorizzati e guasti, si effettua tramite autenticazione, autorizzazione e controllo integrità
↳ GARANTIRE LA CONSISTENZA
- ▶ **Integrità e recupero dei dati:** vi sono controlli per recuperare anomalie causate da programmi o utenti autorizzati
- ▶ **Garantita la consistenza dei dati:** contro il pericolo dovuto ad accessi concorrenti di lettura/scrittura
- ▶ **Facilità di accesso:** accesso semplice e veloce
- ▶ **Interrogazioni:** richieste di dati che verifichino un certo criterio di ricerca
↳ QUERY
- ▶ **Garantita l'integrità dei dati** a 3 livelli:
 - ▶ di campo (tipo e vincoli espliciti) → DEFINIZIONE DEI TIPI + CHECK
 - ▶ di tabella (integrità sull'entità: non duplicati e pk non nulla) →
 - ▶ di associazione (integrità referenziale) →

Garantire l'integrità significa garantire la consistenza, validità dei dati. Un **vincolo di integrità** è una proprietà che deve essere soddisfatta dalle istanze di un database

DataBase Management System

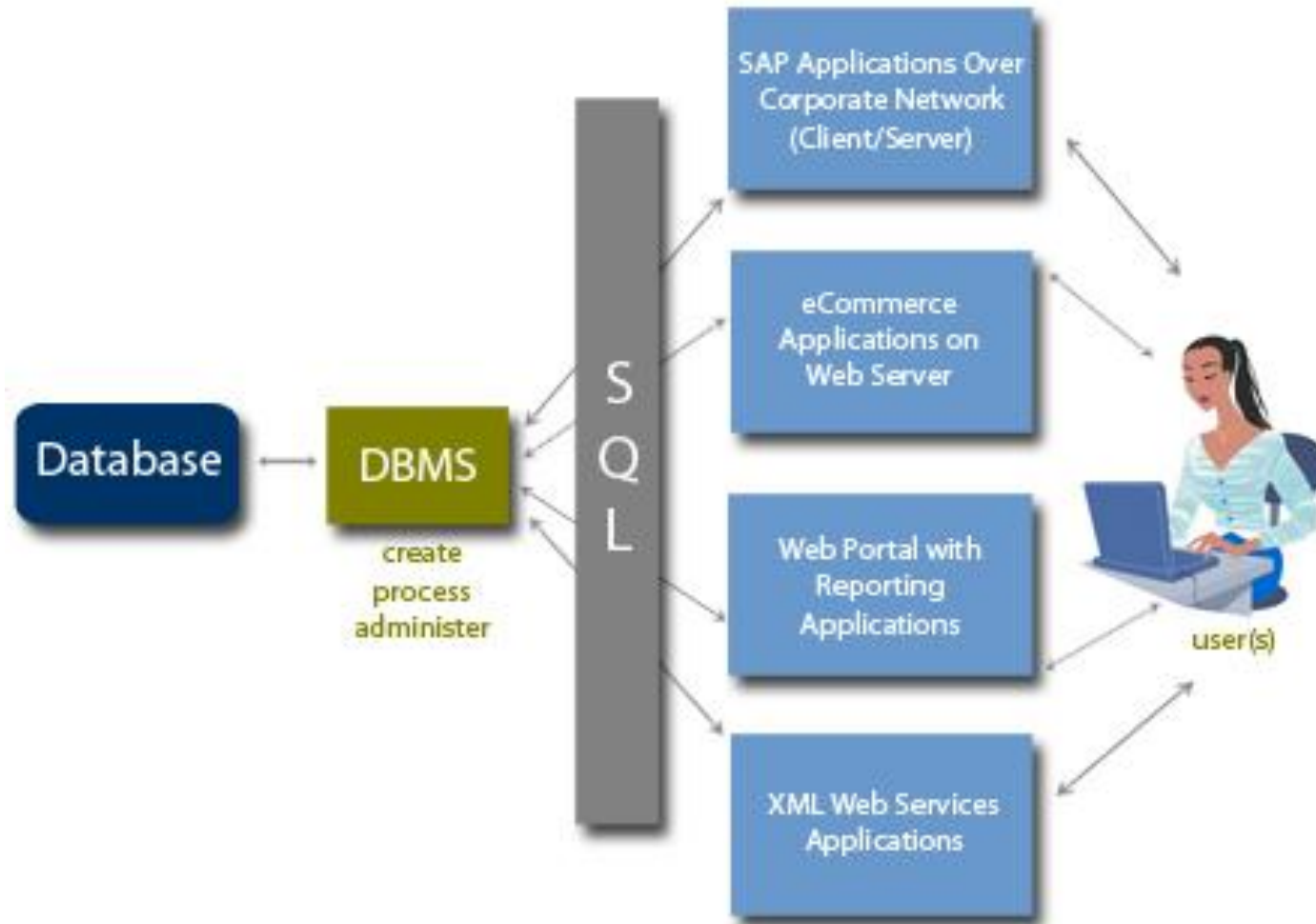
DBMS

I prodotti software per la gestione dei DB si chiamano **DBMS** - **DataBase Management System**.
Costituiscono un interfaccia utente/applicazione-DB.
Permettono di:

- ▶ Definire la struttura dei dati (modello logico)
- ▶ Manipolare ed interrogare il DB
- ▶ Controllare l'integrità dei dati
- ▶ Permettere la condivisione
- ▶ Garantire sicurezza e protezione e persistenza
- ▶ Gestire il modo in cui fisicamente sono archiviati i dati

DataBase Management System

DBMS



DBMS: modifica e interrogazione

- I linguaggi di interrogazione del database mediante *query* (**interrogazioni**) e i generatori di *report* permettono agli utenti di interrogare in maniera interattiva il database e di analizzarne i dati.
- Il DBMS fornisce un modo per **aggiornare e immettere** nuovi dati nel database, oltre che per interrogarlo, questa capacità permette di gestire database personali.

DBMS: integrità

Il DBMS può mantenere **l'integrità** del database:

- non consentendo a più utenti di modificare lo stesso record contemporaneamente (blocco del record).
- Il database può impedire l'immissione di due record duplicati; per esempio può essere impedita l'immissione nel database di due clienti con lo stesso numero identificativo ("campi chiave").
- L'integrità è garantita dalla proprietà "ACID" delle transizioni.

DBMS: gestione autorizzazioni

Il sistema di sicurezza dei dati impedisce agli utenti non autorizzati di visualizzare o aggiornare il database.

Mediante l'uso di *password* (parole d'ordine) agli utenti è permesso l'accesso all'intero database o a un suo

sottoinsieme: in questo secondo caso si parla di ***subschema o vista***. Per esempio un database di impiegati può contenere tutti i dati riguardanti un singolo soggetto e un gruppo di utenti può essere autorizzato a vedere solamente i dati riguardanti lo stipendio, mentre altri utenti possono essere autorizzati a vedere solamente le informazioni che riguardano la sua storia lavorativa e la situazione sanitaria.

DBMS: affidabilità

Un DBMS affidabile deve essere:

- *Flessibile*: quando continua a fornire un servizio all'utente anche se si verificano problemi interni o esterni.
- *Ripristinabile*: quando a causa di un problema causato da un utente interno al sistema, il sistema può essere facilmente ripristinato a uno stato precedentemente conosciuto, senza perdita di dati.
- *Controllato*: quando fornisce un servizio preciso e tempestivo all'occorrenza.

DBMS: affidabilità

- *Ininfluenzabile*: quando le modifiche e gli aggiornamenti non influenzano la fornitura del servizio da parte del sistema.
- *Pronto per la produzione*: il sistema contiene difetti minimi che richiedono un numero limitato di aggiornamenti, comunque previsti.
- *Prevedibile*: funziona come previsto o promesso e ciò che funzionava in precedenza continua a funzionare.

DataBase Management System

I linguaggi di un DBMS

I DBMS permettono tramite linguaggi (comandi) specifici di:

- ▶ **Definire la struttura dati logica**, definire le maschere video e i prospetti (Data Definition Language-**DDL**)
- ▶ Definire la struttura fisica relativamente ad una specifica MM (Data Media Control Language-**DMCL** o Storage Definition Language-**SDL**)
- ▶ **Manipolazione dei dati** (Data Manipulation Language-**DML**)
- ▶ Definire i vincoli di sicurezza, le autorizzazioni agli accessi e tipi di operazioni consentite agli utenti (Data Control Language-**DCL**)
- ▶ **Interrogazione del DB** (**Data Query Language – DQL**)
- ▶ **TCL (Transaction Control Language)**: consente di avviare, concludere e gestire le transazioni.

DataBase Management System

I linguaggi di un DBMS in SQL

Istruzioni del linguaggio SQL

DDL

CREATE
ALTER
DROP
RENAME
TRUNCATE
COMMENT

DML

INSERT
UPDATE
DELETE
MERGE

DQL

SELECT
JOIN
SUBQUERY

DCL

GRANT
REVOKE

TCL

COMMIT
ROLLBACK
SAVEPOINT

DataBase Management System

Architettura a 3 livelli

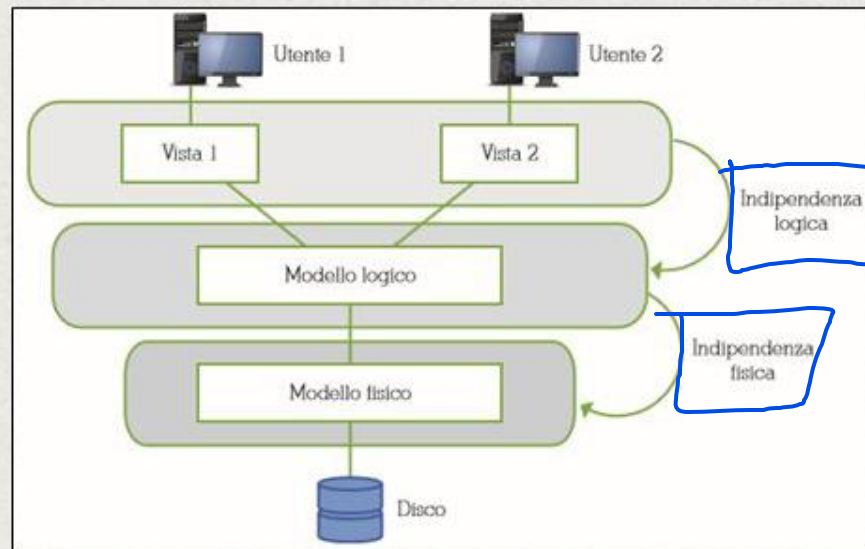
I DBMS permettono di interagire con il DB su 3 livelli:

- ▶ **Livello esterno:** usato dagli utenti del DB. Il **DBA** (*DB Administrator*) ha realizzato per essi delle **viste** differenti in base a permessi definiti con il **DCL** e compiti degli utenti. A questo livello è possibile modificare i dati con il **DML** o fare interrogazioni con il **DQL**.
- ▶ **Livello concettuale o logico:** viene definito lo schema dei dati indipendentemente dalla implementazione fisica. Si usa il **DDL**.
- ▶ **Livello interno o fisico:** a questo livello il **DBA** decide i supporti di memorizzazione, l'organizzazione, i metodi di accesso per il DB,... Si usa il **DMCL**.

DataBase Management System

Architettura a 3 livelli

- il **livello fisico** gestisce i *file* che dovranno essere memorizzati sul disco,
- il **livello logico** gestisce le tabelle relazionali,
- il **livello di interfaccia verso l'esterno** si occupa di quali dati far vedere ("vista") agli utenti e in che modalità.



DataBase Management System

Architettura a 3 livelli

L'architettura a tre livelli dei db, descrive i dati secondo 3 differenti livelli di astrazione, mediante opportuni schemi. Il DBMS realizza questi meccanismi di astrazione dei dati e assicura l'**indipendenza dei dati**: i livelli superiori non sono influenzati , entro certi limiti, dai cambiamenti che avvengono in quelli inferiori:

- ▶ **Indipendenza dalla struttura fisica dei dati**: si possono modificare i supporti, spostare tabelle, ecc. senza modificare il livello logico e quindi quello esterno realizzato dalle applicazioni
- ▶ **Indipendenza dalla struttura logica dei dati**: si possono modificare le definizioni delle strutture fatte a livello logico, senza modificare le viste esterne utilizzate dalle applicazioni

DataBase Management System

Tipi di architettura

- ▶ **Stand-alone**: insieme di dati di piccole dimensioni risiedente su un PC (Access, SQLite).
- ▶ **Terminal server**: architettura diffusa negli anni passati in cui in un **mainframe** erano presenti dati e procedure che venivano utilizzati da più utenti ai vari terminali (DB2 della IBM per AS400 e Oracle)
- ▶ **Client-server**: su un computer server risiedono i dati e il programma database server in grado di rispondere alle richieste del sw database client residente su altri computer collegati in rete. La comunicazione tra client e server prevede: instaurazione della connessione e dell'autenticazione, richiesta dal client che resta in attesa della risposta del server (DBMS MySQL, MariaDB).

DataBase Management System

Transazioni

- ▶ **Transazione:** insieme di operazioni che devono essere eseguite in modo atomico, come un unico blocco: **L'atomicità** delle transazioni implica che o vengono eseguite completamente o non vengono eseguite. Se una transazione è annullata, per ripristinare la situazione iniziale si devono annullare tutte le operazioni eseguite fino a quel momento.
- ▶ Si classificano in
 - ▶ **Implicite**, create in automatico dal DBMS quando esegue operazioni di aggiornamento, inserimento e cancellazione
 - ▶ **Esplicite**, dichiarate dal programmatore

Es.

- ▶ Trasferimento di denaro tra conti correnti.
- ▶ Prenotazione di un posto in aereo

DataBase Management System

Transazioni – Esempio

```
begin_transaction;  
read (saldoX);  
saldoX = saldoX - importo;  
write (saldoX);  
read (saldoY);  
saldoY = saldoY + importo;  
write (saldoY);  
    if(saldoX < fido) then  
        rollback;  
    else  
        commit;  
    end_if;  
end_transaction;
```

Annula tutte
le modifiche
dall'inizio della
transazione

Conferma tutte
le modifiche
apportate nella
transazione

DataBase Management System

Transazioni - Proprietà ACID

- ▶ Il DBMS deve garantire alle transazioni le seguenti proprietà
 - ▶ **Atomicity (atomicità):** tutte le istruzioni sono eseguite come un'unica unità (o tutto o niente)
 - ▶ **Consistency (consistenza):** il DB si deve trovare in uno stato consistente sia all'inizio che alla fine di una transazione, non deve violare eventuali vincoli di integrità
 - ▶ **Isolation (isolamento):** in caso di transazioni concorrenti solo una può modificare i dati, se una fallisce le altre non devono fallire
 - ▶ **Durability (persistenza):** una volta giunta a termine (commit), le modifiche effettuate dalla transazione sono permanenti nel DB



DataBase Management System

Transazioni – Transaction Log

- ▶ Per garantire l'atomicità il DBMS non esegue gli aggiornamenti direttamente sul DB, ma registra le operazioni da effettuare in un file di sistema chiamato **transaction log**. Ad ogni operazione sul DB si aggiunge un record nel file contenente:
 - ▶ Before image: dato prima della modifica
 - ▶ After image: dato dopo aggiornamento
 - ▶ User: chi ha richiesto la modifica
- ▶ Se la transazione è portata a termine (**committed**), allora viene effettivamente eseguita sul DB e si segna il punto in cui è arrivato nell'esecuzione delle operazione presenti nel log, con un check point
- ▶ Se la transazione è abortita (**aborted**), non si aggiornano i file e il record corrispondente nel file log viene cancellato

DataBase Management System

Transazioni – Lock

- ▶ Nel caso di transazioni concorrenti che vogliono modificare gli stessi dati, il DBMS per garantire la consistenza imposta dei blocchi (**lock**) alle informazioni in modo che solo una transazione possa modificarli.
- ▶ Questi blocchi possono essere impostati a livello di tabella, di riga e di campo.
- ▶ Si distinguono
 - ▶ **Blocchi ottimistici**: in cui le informazioni anche se bloccate da un'altra transazione, possono essere letti da altre transazioni
 - ▶ **Blocchi pessimistici**: in cui le informazioni bloccate non possono neanche essere letti da altre transazioni

DataBase Management System

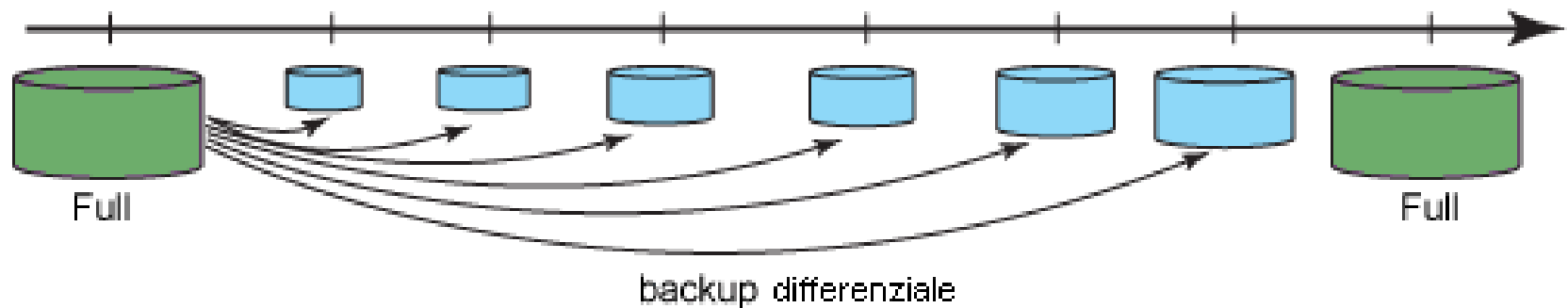
Backup

Uno dei compiti fondamentali di un DBMS è quello del salvataggio periodico dei dati (**backup**) e dell'eventuale loro ripristino (**restore**). Per poter ripristinare il DB, oltre ai file di backup, utilizza anche il **transaction log**. I backup si classificano in:

- ▶ **A caldo** (o Hot backup): effettuato mentre il database è in linea. I dati possono quindi essere modificati mentre il backup è in corso.
- ▶ **Completo** (o Full backup): backup di tutti i files sul sistema. A differenza della disk image, un full backup non include le tavole di allocazione, le partizioni ed i settori di boot.

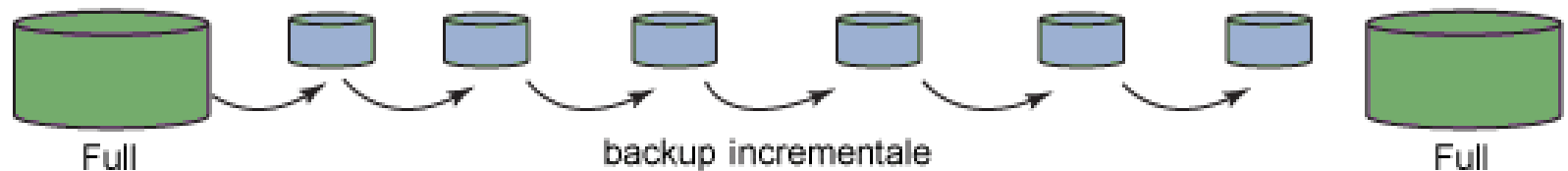
DataBase Management System Backup

- **Differenziale:** Backup cumulativo di tutti i cambiamenti effettuati a partire dall'ultimo backup completo effettuato. Il vantaggio è il minor tempo necessario rispetto ad un backup completo. Lo svantaggio è che i dati da salvare aumentano per ogni giorno trascorso dall'ultimo backup.



DataBase Management System Backup

- **Incrementale:** Backup che contiene tutti i files cambiati dal precedente backup (completo o incrementale). Il backup incrementale è più rapido di quello differenziale, ma richiede tempi di restore più lunghi poiché è necessario partire dall'ultimo backup completo e poi aggiungere in sequenza tutti i backup incrementali.



DataBase Management System

Backup: modalità

- ▶ Se un DB è di dimensioni limitate si farà
 - ▶ Un backup completo ogni sera
 - ▶ Ogni mattina si crea un nuovo transaction log
- ▶ Se un DB è di grandi dimensioni si farà
 - ▶ Un backup completo ogni fine settimana
 - ▶ Un backup incrementale ogni giorno
 - ▶ Ogni mattina si crea un nuovo transaction log

In caso di malfunzionamento si ripristinerà il DB all'ultimo full backup, quindi all'eventuale backup differenziale della sera precedente e tramite le transaction log si ricostruiranno le operazioni eseguite nella giornata

Progetto software e Modellazione dei dati

Fasi per la produzione di un prodotto

- ▶ **studio**

- ▶ Si approfondisce la conoscenza dell'area di competenza

- ▶ **ideazione**

- ▶ Si definisce il modello astratto del prodotto e si specificano le caratteristiche

- ▶ **progettazione**

- ▶ Si formalizza il modello astratto tramite schemi

- ▶ **realizzazione**

- ▶ Si creano i primi prodotti

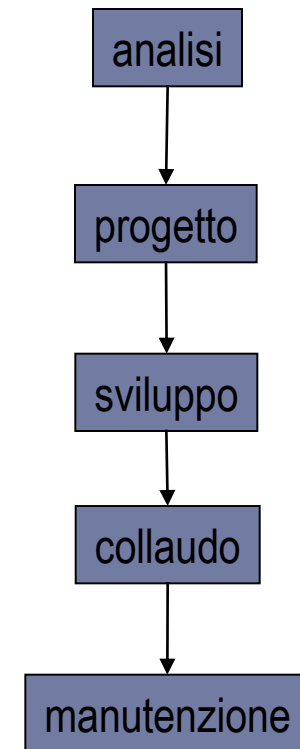
- ▶ **produzione**

- ▶ Termina il progetto inizia la produzione

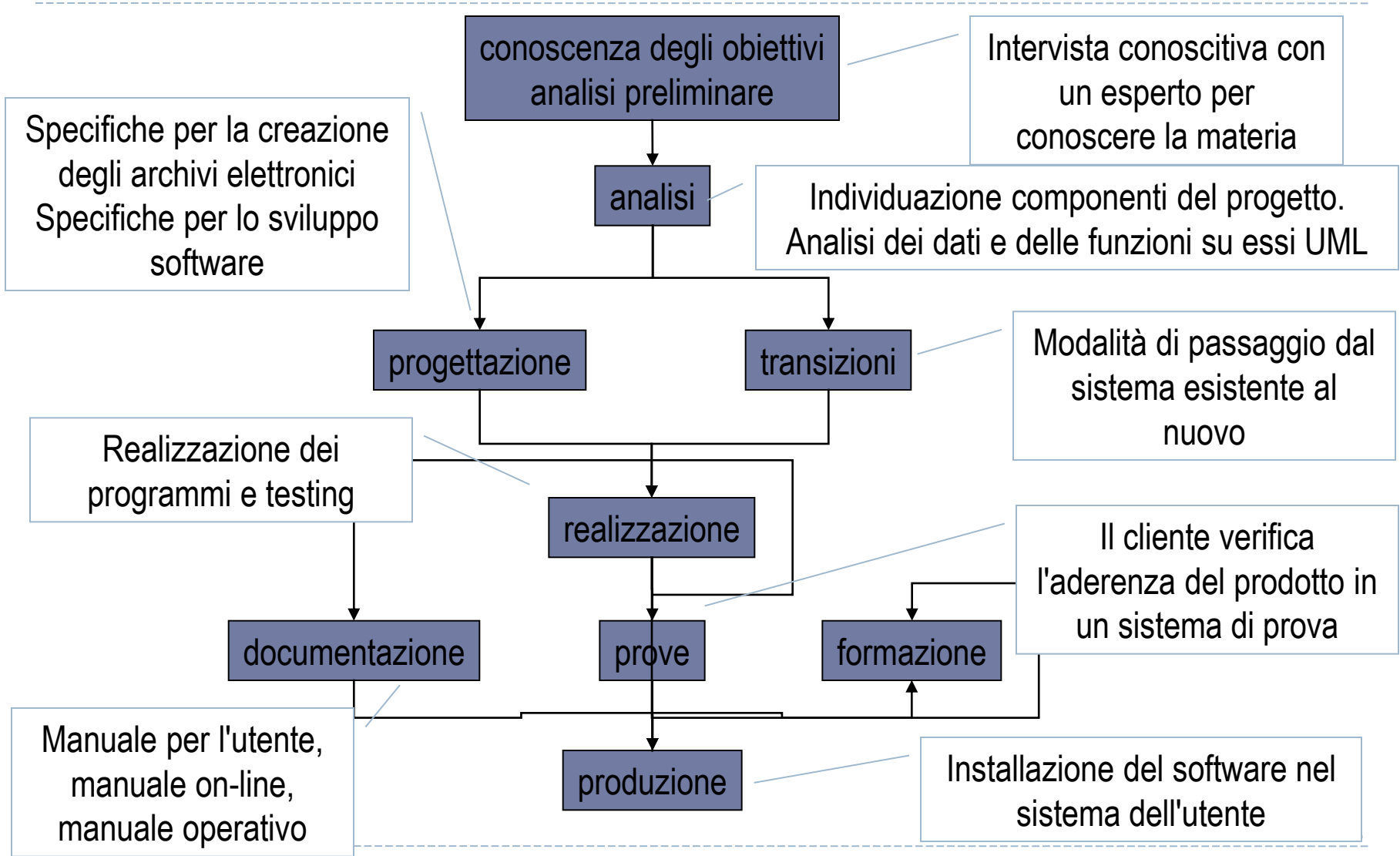
Ciclo di vita del prodotto software

Prodotto software: insieme dei programmi e dati (archivi) necessari per soddisfare le richieste del cliente

Es. Metodologia in cascata



Ciclo di vita del prodotto software



La progettazione di un DB

La progettazione di una base di dati fa parte della progettazione di un software e ha lo scopo di realizzare un database a partire da un insieme di specifiche che formalizzano le esigenze dell'utente.

Prevede:

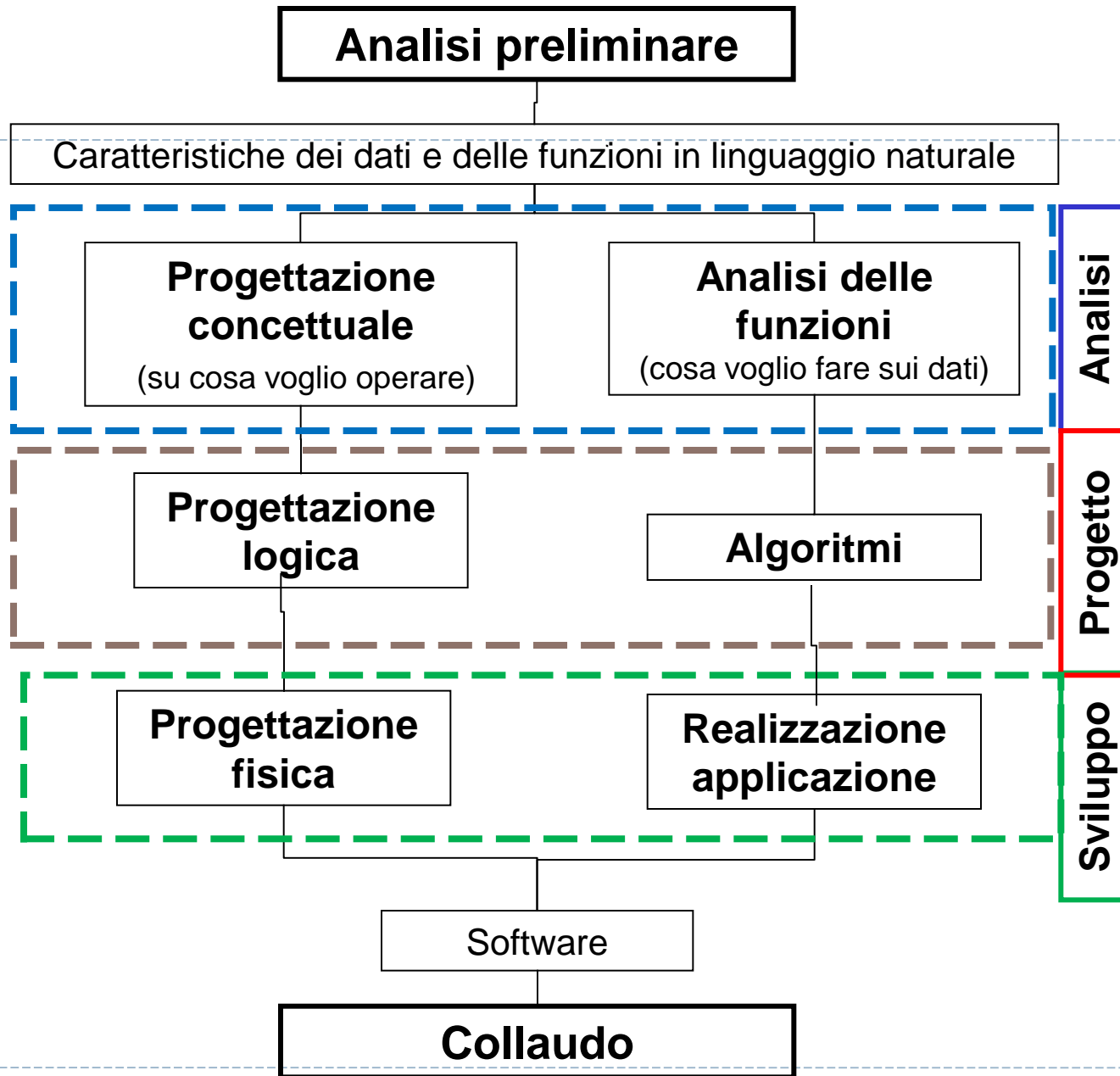
- ▶ Attività tra loro collegate
- ▶ Prodotti intermedi e finali di tali attività
- ▶ Criteri di verifica di qualità di tali fasi e prodotti

Modellazione dei dati

La **modellazione dei dati** si occupa di realizzare il **modello di dati** che è la rappresentazione astratta delle strutture di dati di un DB.

Il modello è indipendente dall'hardware e dal linguaggio che si vuole utilizzare.

Questa fase serve per tradurre i dati dal punto di vista dell'utente al punto di vista delle applicazione e del DB



Modellazione dei dati

Fasi

Progettazione Concettuale (analisi)

- a partire dai requisiti informativi (specificati in linguaggio naturale) viene creato uno schema concettuale (E/R o a oggetti), cioè una descrizione formalizzata e integrata delle esigenze aziendali, espressa in modo indipendente dal DBMS

Progettazione Logica (progettazione)

- si determinano le strutture logiche dei dati derivandole dal livello concettuale; operazione di *mapping*. Si sceglie lo schema logico in base al tipo di DBMS (nel caso di DB relazionale definisco le tabelle)

Progettazione Fisica (realizzazione)

- implementa lo schema logico definendo tutti gli aspetti fisici di memorizzazione e rappresentazione nel DBMS scelto (per esempio creo le tabelle in Access)

caratteristiche dei dati in linguaggio naturale

Progettazione Concettuale

Schema concettuale
(diagramma ER)

Progettazione Logica

Schema logico
(tabelle)

Progettazione Fisica

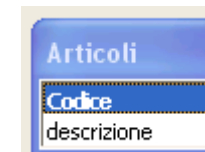
Schema fisico (definisce gli
aspetti fisici di memorizzazione
e rappresentazione. Es Access)



Mapping

Articoli (Codice, Descrizione, ...)

Implementazione



Modellazione dei dati

Modelli logici

Nello sviluppo della teoria dei DB si possono individuare i seguenti modelli logici:

- ▶ **Flat file:** un solo file per l'intero DB (fogli di calcolo)
- ▶ **Modello gerarchico:** il modello logico è rappresentato con uno schema ad albero in cui le entità sono i nodi e gli archi le relazioni. È possibile definire solo relazioni 1:N, rigidità.
- ▶ **Modello reticolare:** il modello logico è rappresentato con un grafo orientato in cui le entità sono i nodi e gli archi le relazioni. È difficile l'implementazione del grafo e la costruzione del software applicativo.

Modellazione dei dati

Modelli logici

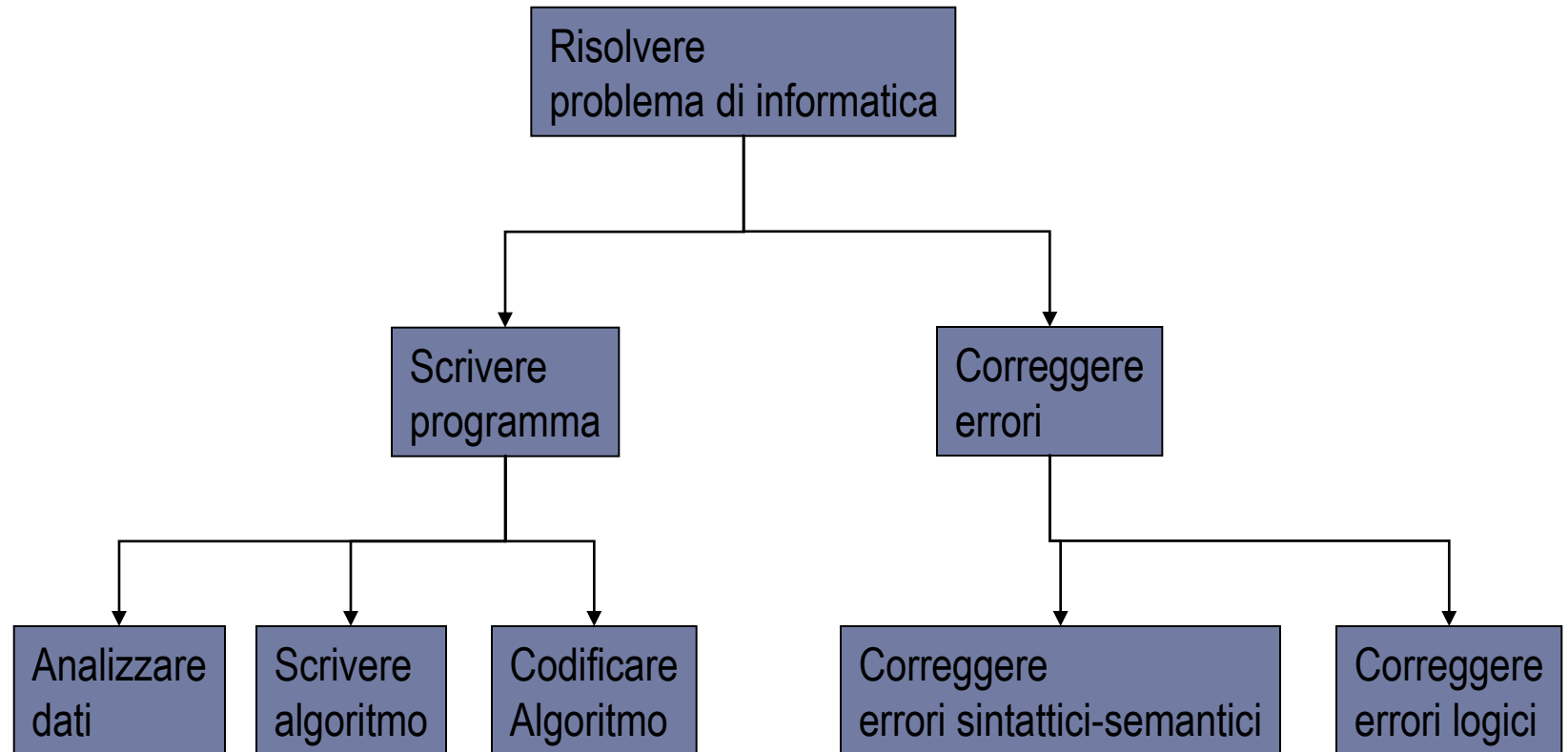
- ▶ **Relazionale:** il modello logico è costituito da tabelle. Derivato dalla matematica. È il più semplice ed efficace, il più utilizzato. (Edward Codd 1970)
- ▶ **OODB (Object Oriented DataBase):** il modello logico è costituito da oggetti e classi. È possibile definire sottoclassi. Adatto a gestire DB non solo testuali (multimediali)
- ▶ **XML:** non è un vero modello, ma costituisce lo standard per l'interscambio di informazioni tra DBMS diversi

Analisi delle funzioni

Si realizza una gerarchia tra le funzioni con un processo di raffinamenti successivi (come top-down). Si schematizza con un **funzionigramma**:

- ▶ Ogni nodo descrive sinteticamente una funzione con operazione da eseguire e oggetto su cui agisce (non si indica chi lo esegue)
- ▶ Le funzioni relative ad attività complesse (funz. **Madre**) si scompongono in funzioni (almeno 2) con maggiori dettagli (funz. **Figlie**)
- ▶ La funzione **radice** contiene il nome del progetto
- ▶ Tra le funz. Figlie dello stesso livello non esiste relazione (possono essere in alternativa o no)

Analisi delle funzioni



Modello Concettuale

Modello E-R

Modello E-R

- ▶ Il modello concettuale **E-R** o **Entità-Relazione** o **Entity-Relationship** è un modello grafico per la descrizione dei dati e delle loro relazioni in una realtà di interesse.
- ▶ Permette di modellare il mondo reale utilizzando solo entità e relazioni tra esse.
- ▶ È utile per i progettisti del DB sia per la realizzazione del progetto che per comunicare la struttura all'utente finale.
- ▶ È indipendente dalle applicazioni e dal DBMS scelto.
- ▶ È stato ideato da Peter Chen nel '76
- ▶ È molto semplice e intuitivo
- ▶ Facilita il passaggio al modello logico successivo

Modello E-R

Entità

I. Entità

- È un oggetto concreto o astratto distinguibile dagli altri. Sono insiemi di oggetti che sono di interesse per rappresentare la realtà (es. studenti). Tutti gli elementi dell'insieme si caratterizzano per un insieme di proprietà comuni
- Gli elementi di una entità sono le **istanze** (es. lo studente Mario Rossi), distinguibili tramite i valori assunti dalle varie caratteristiche. Se le entità sono gli insiemi, le istanze sono gli elementi.
- Corrispondono ai sostantivi
- Un entità si dice **forte** se non ha bisogno di altre entità per essere identificata, altrimenti è **debole** (per esempio un paziente è un entità forte, l'esame è debole)

Studente

Modello E-R

Attributi

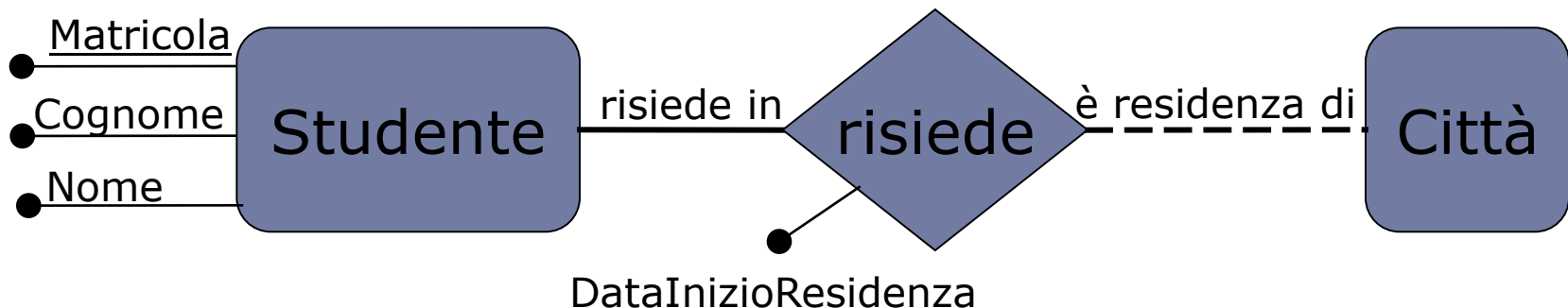
2. **Attributi**

- Sono proprietà, caratteristiche delle entità o delle associazioni (es. Nome, DataInizioResidenza)
- Per ciascuno di essi bisogna definire
 - ▶ **Nome**
 - ▶ **Formato** (car, num, data) e
 - ▶ relativo **Dominio** (insieme dei possibili valori)
 - ▶ **Dimensione** (numero di cifre o lettere, se intero o reale)
 - ▶ **Opzionalità** (se è obbligatorio non può essere nullo)
- Non si devono definire gli attributi derivati (deducibili da altri attributi) (età se ho data nascita)
- Dominio: insieme dei possibili valori di un attributo. Si possono definire dei vincoli espliciti (positivo, <13...). Può essere un dominio continuo o discreto.
- NULL: valore che indica "informazione mancante", "inapplicabile" o "valore sconosciuto". Non è 0 o ""

Modello E-R

Attributi

- Possono essere:
 - ▶ **Semplici** (es. Nome, Cognome) o **Composti/aggregati** (es Indirizzo) che possono essere scomposti in più attributi semplici
 - ▶ **Multipli**, per la stessa istanza, possono avere contemporaneamente più di un valore (es. attributo SportPraticati)



Modello E-R

Attributi

- Alcuni attributi possono avere un Dominio discreto (il suo valore si potrà scegliere da un elenco):
 - Stagione ▼: {'estate', 'autunno', 'inverno', 'primavera'}

ATTENZIONE!

- un attributo con dominio discreto non é per forza un attributo multiplo, cioè non è detto che nel campo ci possano essere contemporaneamente più valori. Es dominio discreto **NON** multiplo:
 - statoFamiglia: {'coniugato', 'vedovo', 'nubile',...}
- e viceversa, un attributo multiplo non ha per forza un dominio discreto. Es. multiplo con dominio **NON** discreto
 - recapitiTelefonici

Modello E-R

Chiavi

- ▶ Tra di essi si individuano una o più **chiavi candidate**: insieme minimo (non si considerano i sovrainsiemi) di attributi che identificano univocamente (non ammettono duplicati) una istanza (ce ne possono essere molte)
- ▶ Tra queste si "elegge" la **chiave primaria** secondo un principio di minimalità: si sceglie quella con il minor numero di campi o che occupi meno spazio in memoria. **NON** una chiave alfanumerica (ce n'è una sola)
- ▶ Se non si trovano candidate o sono troppo grandi se ne può creare una *artificiale*(ID), senza un significato proprio.

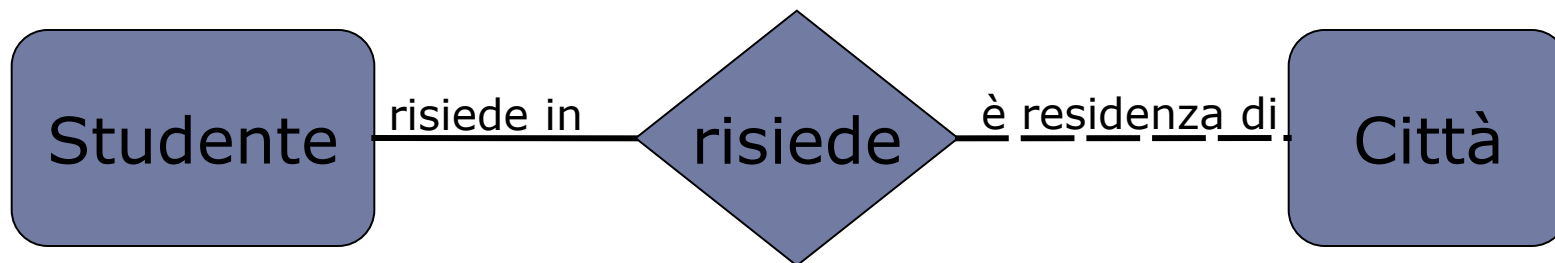
Chiave primaria: ogni istanza deve avere specificato un valore per essa, deve avere un valore univoco, non può diventare nullo durante la vita di un'istanza

Modello E-R

Relazioni o Associazioni

3. Relazioni o Associazioni

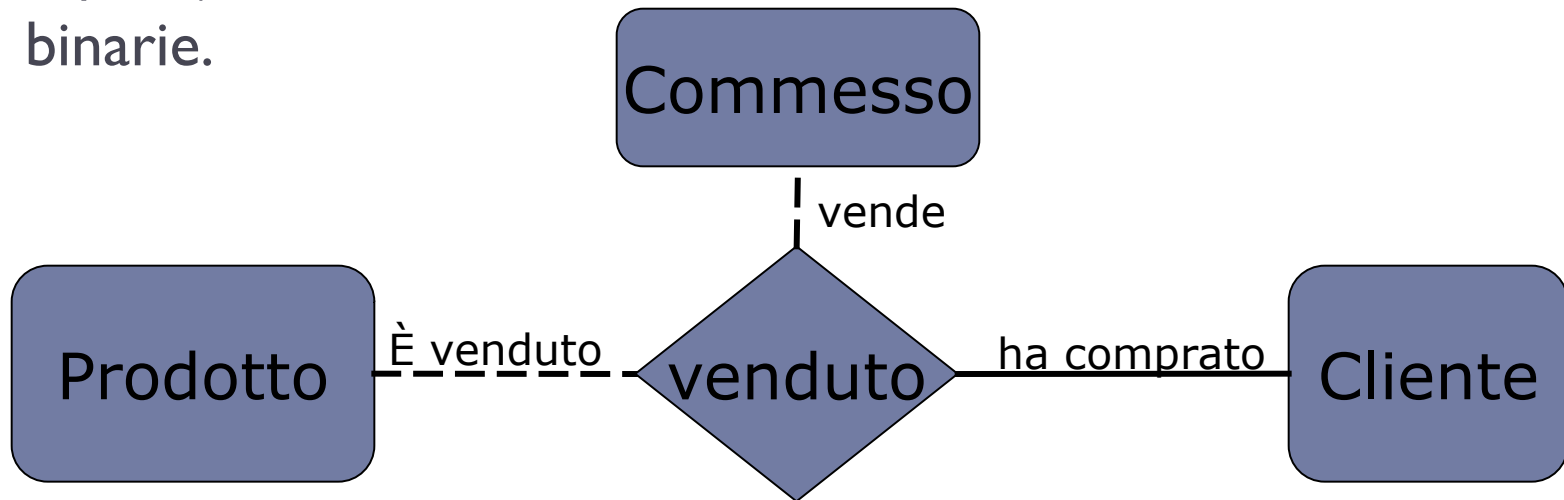
- È un legame che stabilisce un'interazione tra entità (es. risiede in) (sottoinsieme del prodotto cartesiano tra 2 insiemi)
- Ha due **versi** con significati diversi (es risiede in – è residenza di) che determinano il **ruolo** dell'entità nella associazione
- Corrispondono ai verbi
- Possono essere obbligatorie o opzionali. La relazione si chiamerà rispettivamente **totale** o **parziale** (rispetto ad un verso)



Modello E-R

Relazioni o Associazioni: grado

Il **grado** rappresenta il numero di entità coinvolte nella relazione: le associazioni tra 2 entità si dicono **binarie**. Le associazioni che collegano più di 2 entità si dicono **multiple o n-aria**. È sempre possibile trasformare relazioni multiple in relazioni binarie, mantenendo lo stesso contenuto informativo (si possono fare le stesse interrogazioni sui 2 schemi ottenendo le stesse risposte). Pertanto si utilizzeranno schemi con sole relazioni binarie.

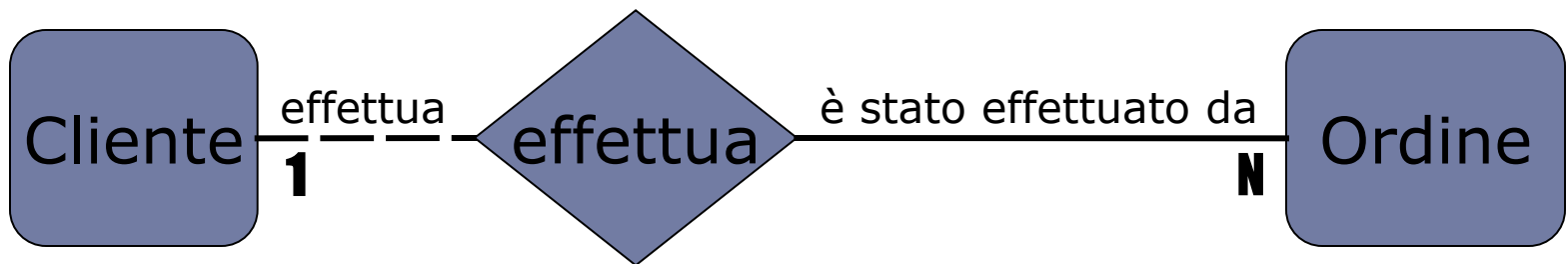
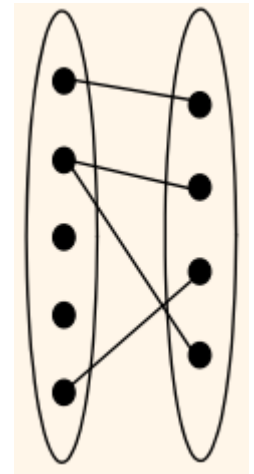


Modello E-R

Cardinalità: uno a molti

Associazione 1:N o semplice

- ▶ Ogni istanza della prima entità corrisponde a zero, una o più istanze della seconda, mentre ad ogni istanza della seconda corrisponde al più una sola istanza della prima

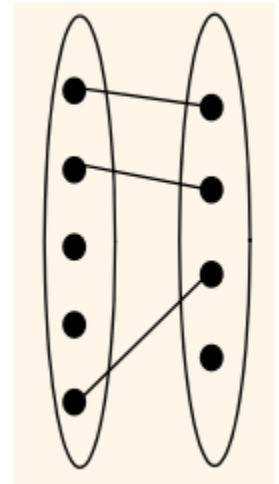


Modello E-R

Cardinalità: Uno a Uno

Associazione 1:1 o biunivoca

- ▶ Ad un'istanza della prima entità corrisponde al più una sola istanza della seconda e viceversa (es. bollo automobile)

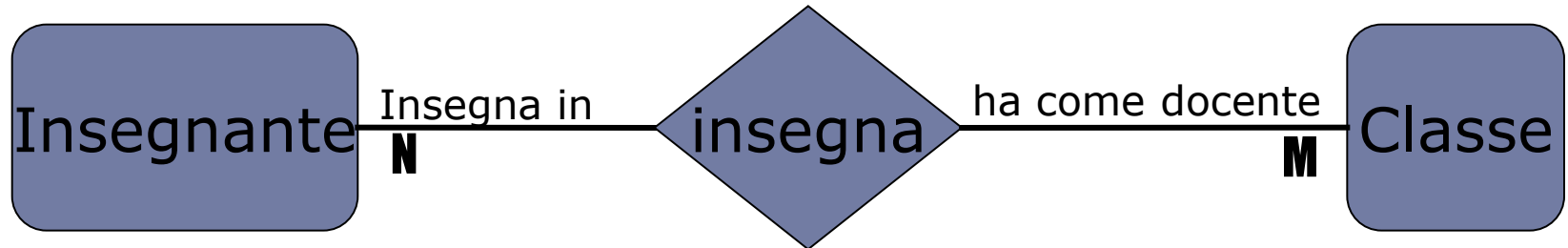
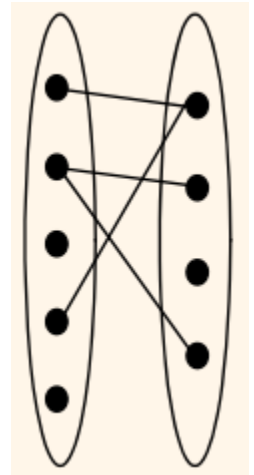


Modello E-R

Cardinalità: molti a molti

Associazione **N:M** o **N:N** o complessa

- ▶ Ad ogni istanza della prima entità corrisponde zero, una o più istanza della seconda e viceversa



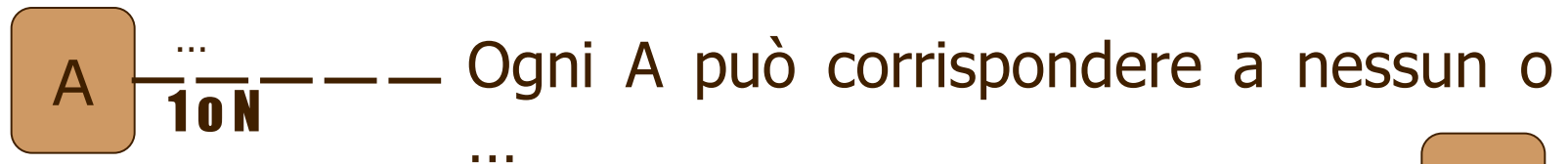
Modello E-R

Associazioni: direzione, esistenza

- ▶ **Direzione:** indica l'entità da cui trae origine la relazione binaria (entità **padre** verso entità **figlio**)
 - ▶ I:I la direzione è dall'entità forte a quella debole, se no è indifferente
 - ▶ I:N il padre è l'entità con cardinalità I
 - ▶ N:M indifferente
- ▶ **Esistenza:**
 - ▶ **Esistenza obbligatoria o associazione totale** : se un'istanza di una entità deve esserci perché sia inclusa in una relazione (es. un progetto deve essere gestito da un capoprogetto)
 - ▶ **Esistenza opzionale o associazione parziale:** se l'istanza non è richiesta (es. una persona può avere un'auto)

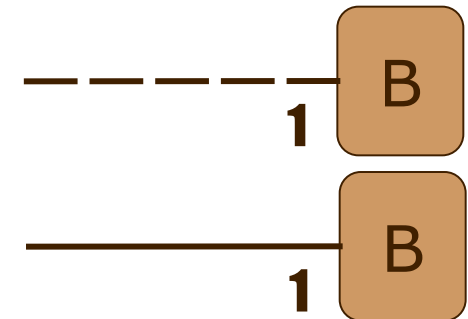
Modello E-R

Regole di lettura

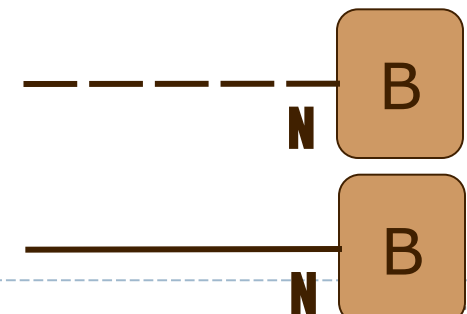


Si verifica la correttezza del modello ottenuto leggendolo: le frasi devono avere senso!

ad un solo B



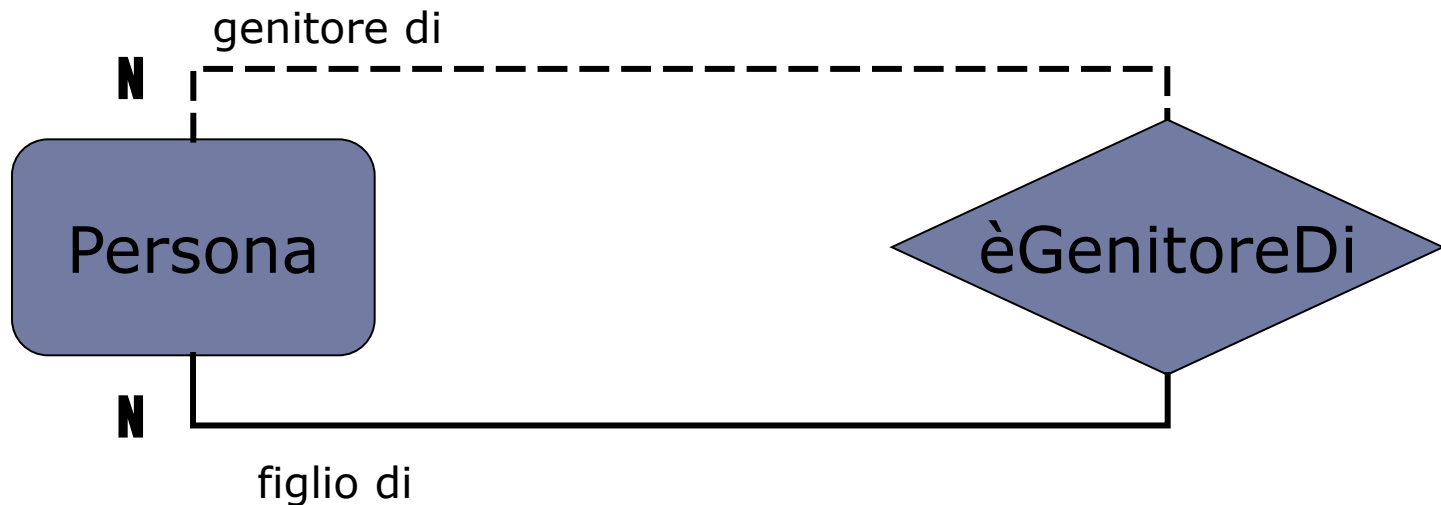
ad uno o più B



Modello E-R

Associazioni ricorsive

Può succedere che una relazione esista tra due entità identiche, si ha il caso particolare di un'associazione sulla stessa entità

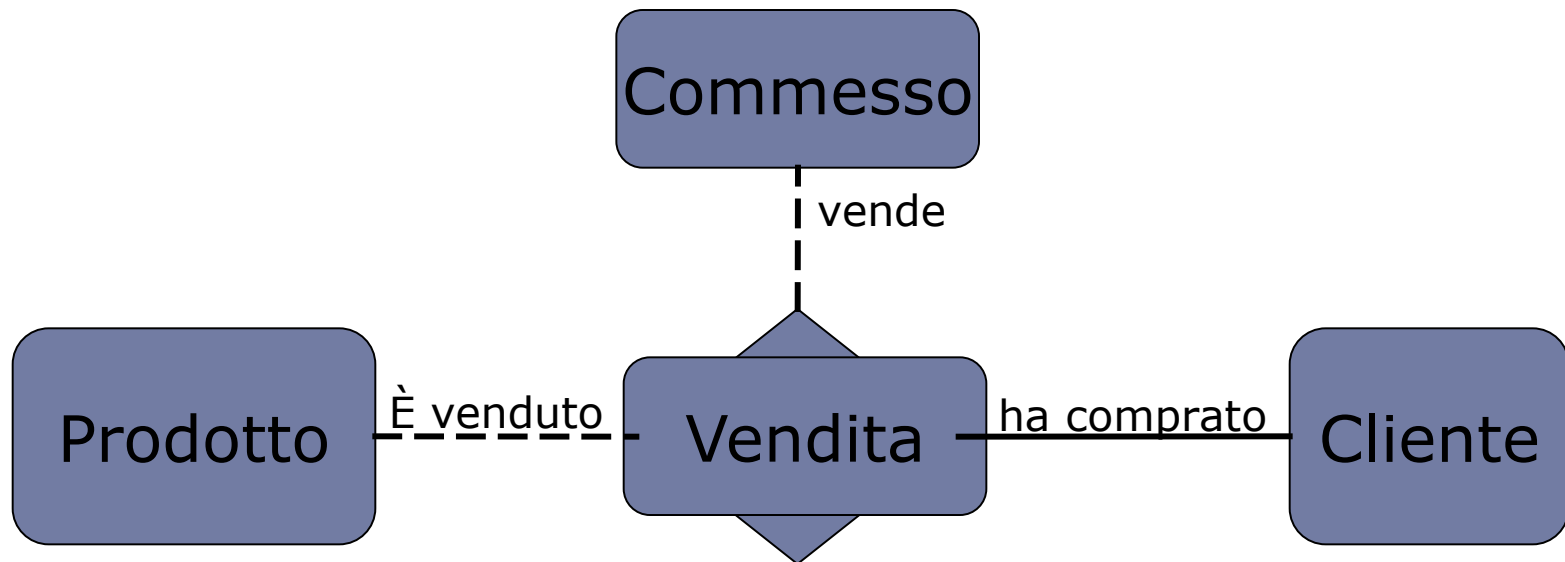


Modello E-R

Ottimizzazione

Bisogna evitare **relazioni complesse** (con grado >2).

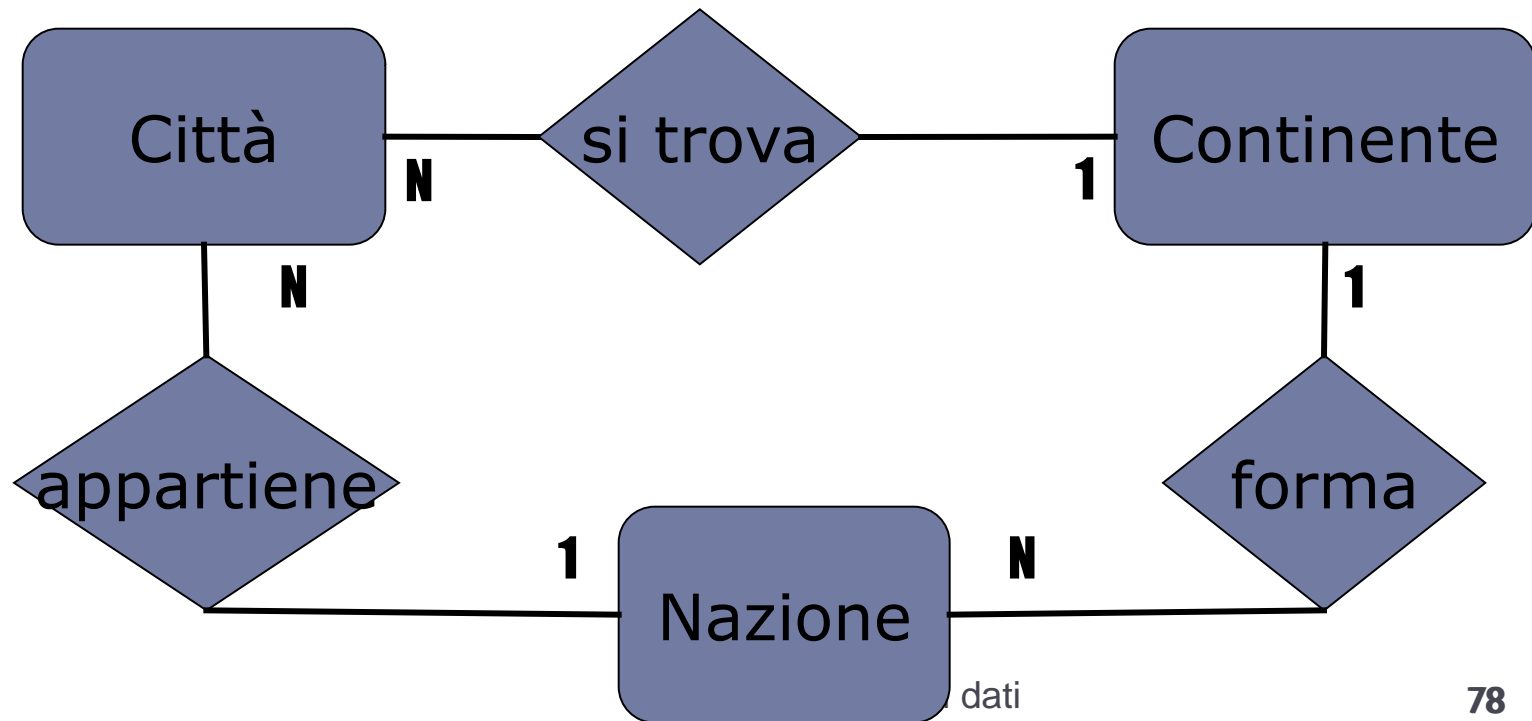
L'associazione si trasforma in una relazione associativa
(nell'esempio diventa VENDITA)



Modello E-R

Ottimizzazione

Bisogna evitare **relazioni ridondanti**: quando si viene a formare un ciclo si controlla se si può eliminare una associazione (nell'esempio si elimina "*si trova*")



Modello E-R

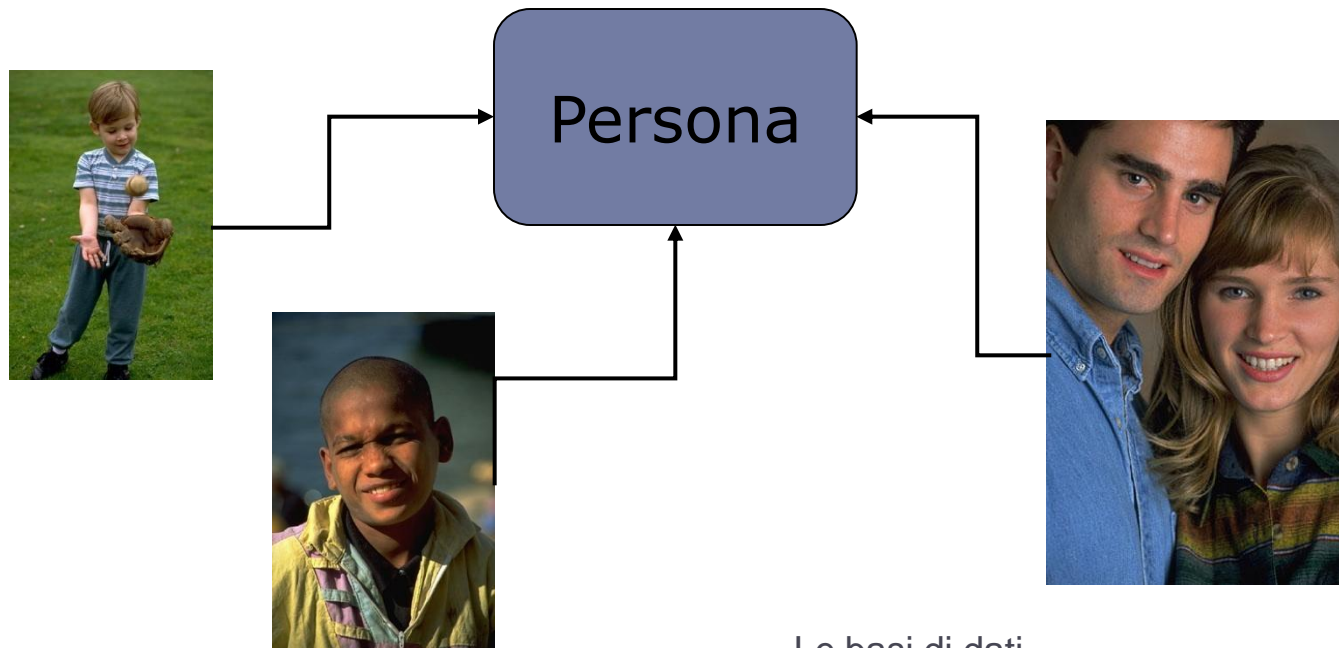
Astrazioni

- ▶ L'individuazione delle entità avviene tramite un processo di **astrazione** ovvero tramite l'individuazione di caratteristiche ritenute significative
- ▶ Esistono 3 procedimenti di astrazione per definire delle entità:
 - ▶ **Classificazione:** individuando caratteristiche comuni in oggetti reali
 - ▶ **Aggregazione:** a partire da entità componenti o proprietà generiamo una nuova entità
 - ▶ **Generalizzazione:** dall'unione di più entità si ottiene una entità più generale. Le entità di partenza rimangono sottoinsiemi della entità ottenuta

Modello E-R

Astrazioni: classificazione

Osservando bambini, adulti e ragazzi reali, vedo che hanno caratteristiche comuni (2 gambe, una testa,..) e li classifico nella classe *Persona*

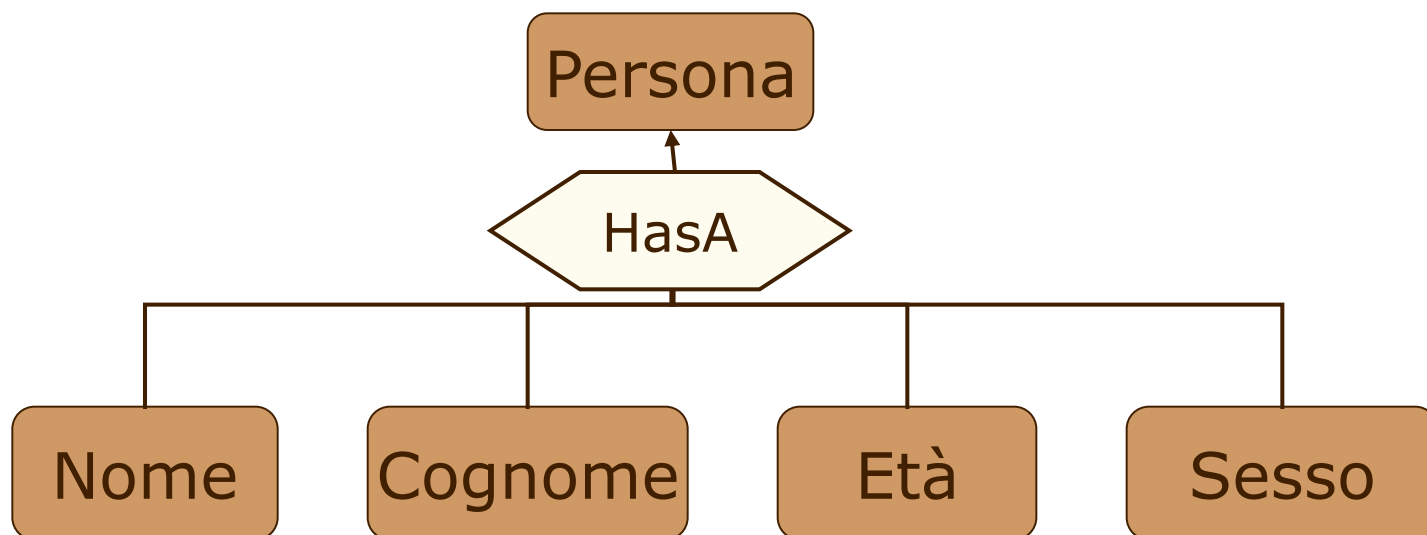


Le basi di dati

Modello E-R

Astrazioni: aggregazione implicita

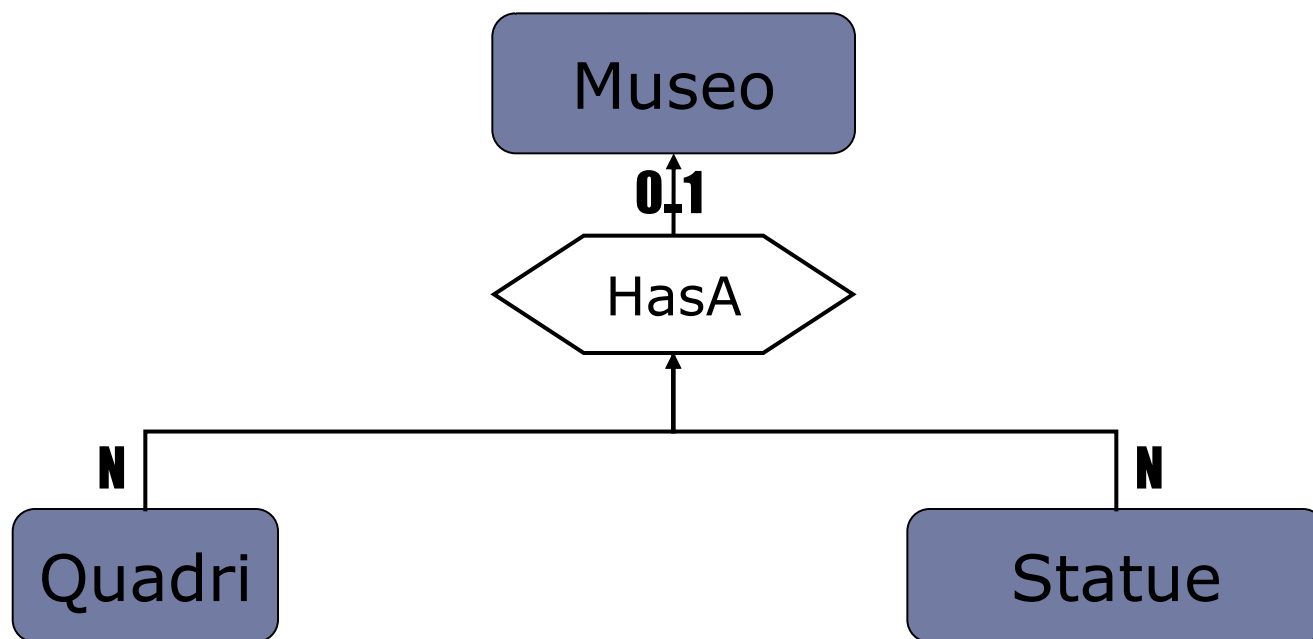
- Considerando le entità: *Nome*, *Cognome*, *Età*, *Sesso*, mi accorgo che la loro aggregazione caratterizza l'entità *Persona* (entità composizione o contenitore) Aggregazioni implicite (DA NON INDICARE): l'entità è aggregazione dei suoi attributi. Una associazione è aggregazione delle entità in relazione



Modello E-R

Astrazioni: aggregazione lasca

- Un'aggregazione è **lasca** se ad un'istanza della entità contenuta, può non corrispondere un'istanza nella entità contenitore, la parte esiste anche senza il tutto e il tutto senza la parte



Modello E-R

Astrazioni: aggregazione lasca

- ▶ È un'associazione più forte, di tipo “intero-parte”. Indica “contiene”, “è parte di”, “è un insieme di”.
- ▶ Le istanze aggreganti (parti) **possono appartenere a più di un'istanza aggregato ...**



- ▶ **... e le parti possono esistere indipendentemente dalle parti**

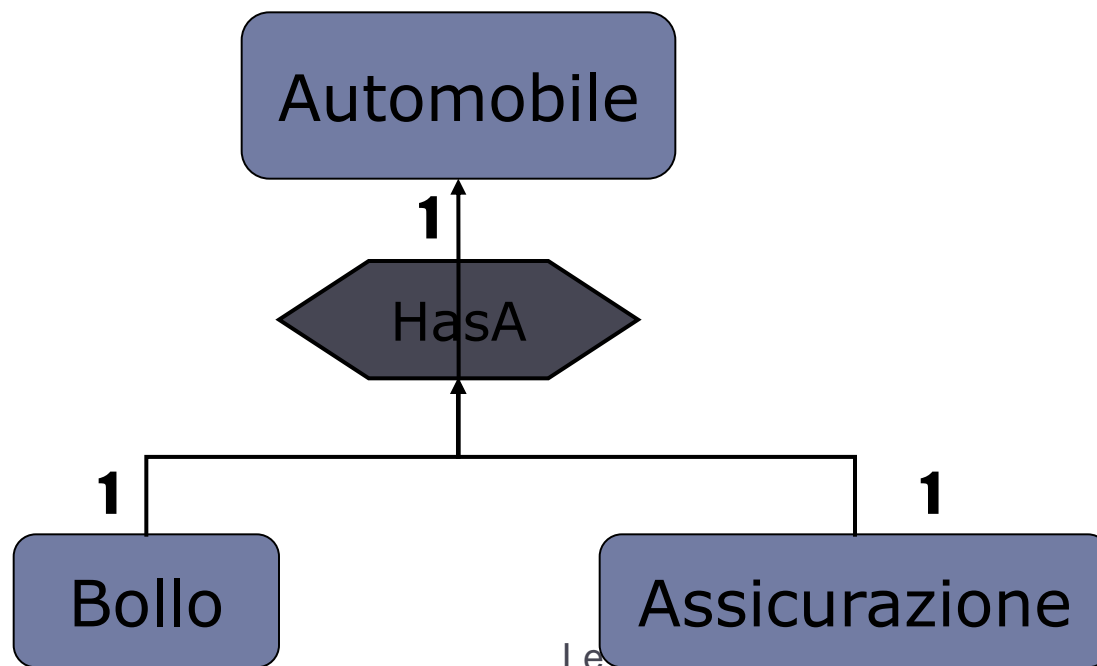


- ▶ L'entità che fa da intero ha molteplicità ≥ 0

Modello E-R

Astrazioni: aggregazione stretta

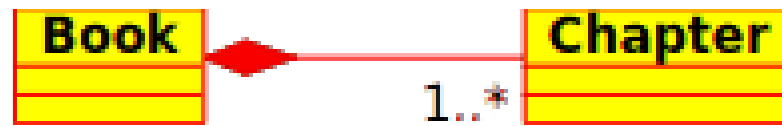
Un'aggregazione è **stretta (composizione)** se ad ogni istanza dell'entità contenuta, deve corrispondere un'istanza nell'entità contenitore. Il contenuto non ha senso senza il contenitore.



Modello E-R

Astrazioni: aggregazione stretta

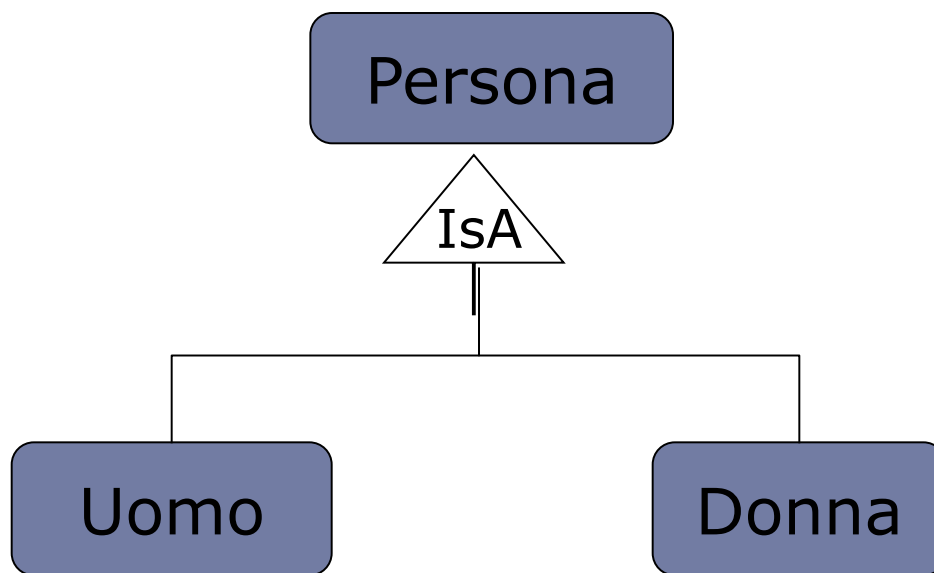
- ▶ È una forma di aggregazione ancora più forte (**HAS-A**) che indica che una “parte” può appartenere ad un solo “intero” in un certo istante di tempo, la parte non può esistere di per sé.
- ▶ La composizione associa composto e componente per tutta la vita dei due elementi
- ▶ La composizione è esclusiva: una specifica istanza componente non può appartenere a due composti contemporaneamente



Modello E-R

Astrazioni: generalizzazione

Considerando le entità di partenza: *Uomo*, *Donna*, generalizzando mi accorgo che la loro unione forma l'entità *Persona*, di cui esse sono sottoinsiemi. Le prime si chiamano **entità figlie** o **specializzazioni**, l'entità ottenuta, **entità genitore** o **generalizzazione**



Modello E-R

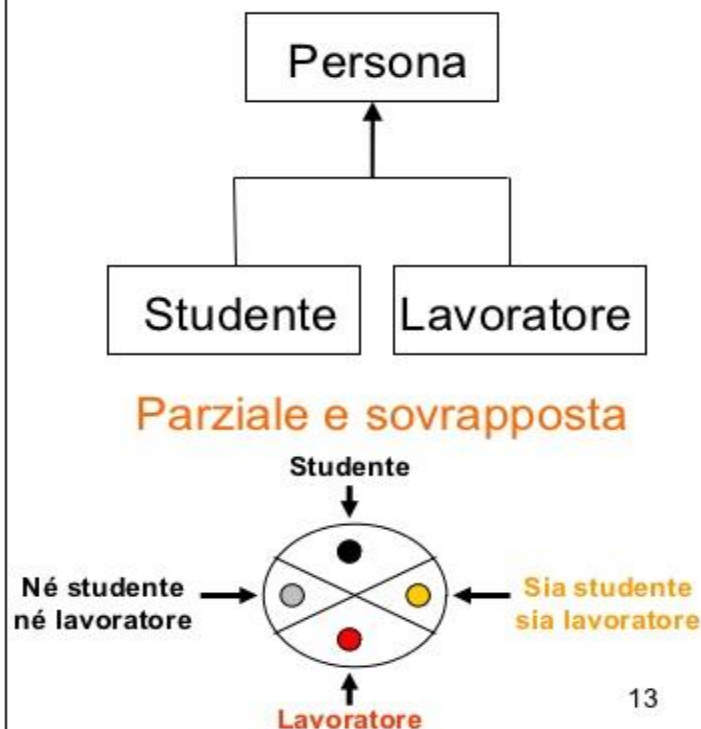
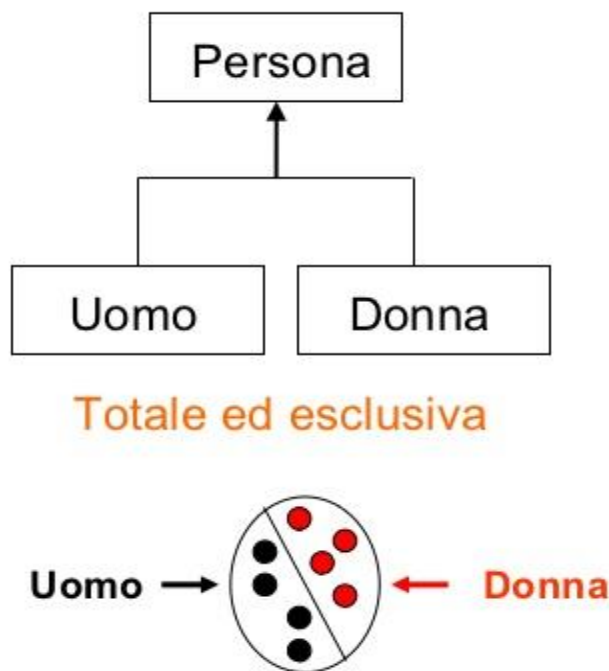
Astrazioni: generalizzazione

- ▶ **Ereditarietà:** le entità figlie ereditano dal genitore:
 - ▶ Gli attributi
 - ▶ Le associazioni
 - ▶ Le generalizzazioni
- ▶ Se le figlie hanno tutte lo stesso attributo, anche il genitore lo avrà
- ▶ Le entità figlie possono avere attributi che non ha né il genitore, né i "fratelli" (per es. servizio militare, nParti)
- ▶ Si ha generalizzazione **totale** se ogni istanza del padre è istanza di almeno una delle figlie, altrimenti è **parziale** (persona è totale)
- ▶ Si ha generalizzazione **esclusiva** se ogni istanza del padre è al massimo istanza di una delle figlie, altrimenti è **sovrapposta** (persona è esclusiva)

Modello E-R

Astrazioni: generalizzazione

Generalizzazione: esempi



13

Modellazione Logica

Modello Relazionale

Modello relazioni

Definizioni

Integrità sull'entità: non possono esserci record duplicati, quindi deve esistere una chiave primaria il cui valore è univoco e non NULL

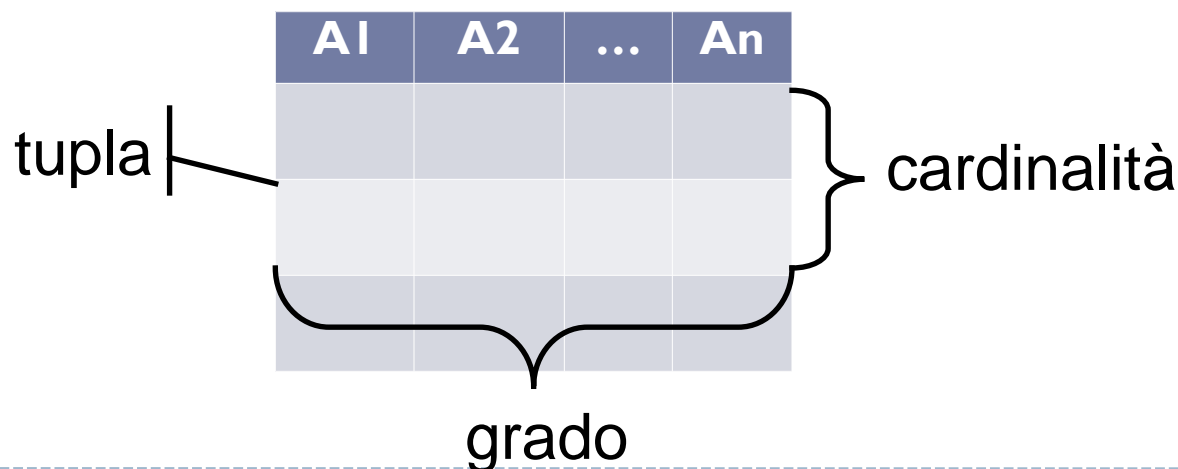
- ▶ Dati gli N insiemi A_1, A_2, \dots, A_N , si chiama **relazione** un sottoinsieme di tutte le N -ple (a_1, a_2, \dots, a_N) del prodotto cartesiano degli N insiemi. $R \subseteq A_1 \times A_2 \times \dots \times A_N$. Si rappresenta con una tabella.
- ▶ $N \geq 1$ è il **grado** della relazione. (n° di colonne)
- ▶ Gli insiemi A_i sono i **domini**, ad ogni dominio è associato un nome detto **attributo**.
- ▶ Gli elementi di R sono dette **tuple**, sono tutte diverse fra loro. Il loro insieme è variabile nel tempo: si possono aggiungere, modificare, cancellare.
- ▶ In ogni istante l'insieme delle tuple di R è detto **istanza della relazione**.
- ▶ Il numero di tuple presente in quel momento è la **cardinalità** della relazione. (n° di righe)
- ▶ L'insieme minimo di attributi che identificano univocamente le tuple si chiama **chiave primaria (pk)**, deve esistere e non può essere NULL (*integrità sull'entità*)

Modello relazionale

Rappresentazione

Una relazione viene a coincidere con una tabella, in cui le intestazioni di colonna sono gli attributi e le righe le tuple.

Relazione	Tabella o Tabella relazionale
Attributi	Colonne o Campi
Tupla	Riga o Record



Modello relazionale

Proprietà relazioni

Le tabelle relazionali hanno le seguenti proprietà:

1. I valori sono atomici, non ulteriormente scomponibili
2. Tutti valori di una colonna appartengono al medesimo dominio
3. Ogni riga è univoca (differisce almeno per la pk)
4. La sequenza delle colonne non è significativa
5. La sequenza delle righe non è significativa
6. Ogni colonna deve avere un nome univoco

Modello relazionale

Derivazione del modello logico

- ▶ Il modello relazionale, su cui si basano i DB relazionali, serve per definire la struttura dei dati per poter definire le tabelle nel DBMS scelto.
- ▶ La struttura si deriva dal modello ER seguendo delle semplici regole
- ▶ La derivazione dal modello ER al modello logico si chiama **mapping**

Modello relazionale

Mapping

- ▶ Ogni *entità* diventa una relazione (tabella)
- ▶ Ogni *attributo* dell'entità diventa un attributo della relazione e ne eredita le caratteristiche (tipo, dim, obbligatorietà). Gli attributi *composti* vengono suddivisi in elementari. Quelli *multipli* sono suddivisi in campi se i valori contemporanei sono limitati (es Telefoni) o in una associazione 1:N o N:N con una nuova entità se sono molti (es. Sports)
- ▶ La *chiave primaria* nell'entità diventa chiave primaria della relazione

Modello relazionale

Mapping

- ▶ L'associazione $1:1$ diventa una sola relazione fusione delle due entità con l'unione degli attributi e l'eventuale integrazione con l'attributo della associazione (oppure si tratta come una $1:N$)
- ▶ L'associazione $1:N$ fa aggiungere un attributo nella entità di arrivo (con la N) con valore la chiave dell'entità di partenza (diventa **chiave esterna**). Si aggiungono anche gli eventuali attributi della relazione

Modello relazionale

Mapping

- ▶ L'associazione $N:N$ diventa una nuova relazione composta dalle chiavi delle due entità che diventano esterne e dagli eventuali attributi della relazione.

Bisogna definire la chiave primaria:

- se non possono esistere due record con la stessa combinazione di valori per le chiavi esterne la chiave primaria sarà l'unione delle stesse chiavi,
- altrimenti si crea un campo apposta.

Modello relazionale

Chiave esterna

Una **chiave esterna** o **foreign key (fk)** crea una gerarchia tra le istanze delle tabelle associate:

- ▶ l'istanza che contiene la chiave esterna si chiama **figlio**,
- ▶ quella che contiene la chiave primaria corrispondente si chiama **padre**.

Modello relazionale

Integrità referenziale

Nel modello relazionale deve essere garantita l'**integrità referenziale** il cui scopo è di impedire la presenza di record orfani e di mantenere sincronizzati i riferimenti, in modo che non vi siano record che facciano riferimento a record non più esistenti: deve esistere coerenza tra le tabelle associate, cioè ad ogni chiave esterna non NULL deve corrispondere una chiave primaria nella tabella associata.

Per garantire l'integrità referenziale il DBMS deve seguire alcune regole a seconda dell'operazione effettuata:

- ▶ **Regole di inserimento o inserzione**
- ▶ **Regole di cancellazione**
- ▶ **Regole di modifica**

Modello relazionale

Integrità referenziale

Inserzione :

- ▶ **dipendente:** si può inserire un'istanza figlio solo se l'istanza padre esiste
- ▶ **automatica:** se si inserisce un figlio prima del padre, si crea anche il padre con gli altri campi *null*
- ▶ **nulla:** se si inserisce un figlio prima del padre, si assegna *null* alla fk
- ▶ **di default:** se si inserisce un figlio prima del padre, si imposta la fk a un valore predefinito

Modello relazionale

Integrità referenziale

Cancellazione:

- ▶ **con restrizione:** si può cancellare un'istanza del padre solo se non ha figli
- ▶ **a cascata:** quando si elimina un padre, si eliminano anche tutti i figli
- ▶ **nulla:** quando si elimina un padre, la fk nelle istanze dei figli viene impostata a *null*
- ▶ **di default:** quando si elimina un padre, la fk nelle istanze dei figli viene impostata a un valore predefinito

Modello relazionale

Integrità referenziale

Modifica della fk di un figlio senza avere una corrispondente pk in un padre:

- ▶ **dipendente:** non lo permette
- ▶ **automatica:** si crea un padre con quella pk e gli altri campi *null*
- ▶ **nulla:** si assegna *null* alla fk
- ▶ **di default:** si imposta la fk a un valore predefinito

Modifica della pk di un padre:

- ▶ **a cascata:** si aggiornano tutte le fk di tutti gli eventuali figli

Modello relazionale

Integrità referenziale in Access

- ▶ Se si applica l'integrità referenziale ad una relazione, verranno automaticamente rifiutati gli aggiornamenti che determinano la modifica della pk o l'eliminazione di un padre con dei figli.
- ▶ Con l'opzione **Aggiorna campi correlati a catena**, quando si aggiorna una chiave primaria, tutti i campi che fanno riferimento alla chiave primaria vengono aggiornati automaticamente.
- ▶ Con l'opzione **Elimina record correlati a catena**, quando si elimina il record che contiene la chiave primaria, tutti i record che fanno riferimento alla chiave primaria vengono eliminati automaticamente.

Modello relazionale

Mapping

- ▶ L'associazione *ricorsiva* 1:N o N:N si traduce normalmente, ma si tiene conto che le chiavi esterne sono tutte riferite alla stessa tabella

Persona (cod, Cognome, Nome, NatoIl)

èGenitoreDi (Genitore, Figlio)

- ▶ Le *associazioni multiple* (non dovrebbero esserci, ma solo binarie) generano una nuova tabella per l'associazione e si individuano le chiavi esterne nelle varie entità coinvolte.

Commessi (...)

Prodotti (...)

Clienti (...)

Vendite (codCommesso, codProd, codCli, Data, Qt)

Modello relazionale

Mapping

Le *associazioni IsA* possono venir tradotte in diversi modi a seconda dello schema ER di partenza:

1. **Accorpamento delle figlie nel padre**

Persona (ID, Cognome, Nome, NatoIl, Sex▼, SerMilitare-, nParti-)

Per cui sex specifica la sottoclasse e i campi delle figlie sono opzionali

2. **Accorpamento del padre nelle figlie se totale**

Uomini (cod, Cognome, Nome, NatoIl, ServizioMilitare)

Donne (cod, Cognome, Nome, NatoIl, nParti)

3. **Sostituendo la generalizzazione con associazioni 1:1 se esclusiva o 1:N se sovrapposta**

Persona (cod, Cognome, Nome, NatoIl)

Uomini (codPersona, ServizioMilitare)

Donne (codPersona, nParti)

Modello relazionale

Mapping

- ▶ Ad ogni entità coinvolta in una *associazione HasA* corrisponde una nuova tabella. Se il numero di elementi è fisso, nella tabella dell'entità contenitore ci saranno delle chiavi esterne associate ad ogni entità contenuto, se no si seguono le regole come per le associazioni 1:N o N:N (solo per lasche)
- ▶ Le associazioni di aggregazione *lasca* avranno vincoli di integrità referenziale con chiavi esterna opzionali

Museo(codM, Nome, Indirizzo, Città, Paese)

Quadri(codQ, titolo, autore, Museo-)

Statue(codS, titolo, autore, Museo-)

- ▶ Le associazioni di aggregazione *stretta* avranno vincoli di integrità referenziale che richiedono l'obbligatorietà delle chiavi esterne (le modifiche o cancellazioni si ripercuotono sulle tabelle associate)

Auto(targa, Marca, Prezzo)

Bolli(cod, Importo, Validità, targa)

Assicurazioni(cod, Compagnia, Premio, Massimale,
Scadenza, targa)

Modello relazionale

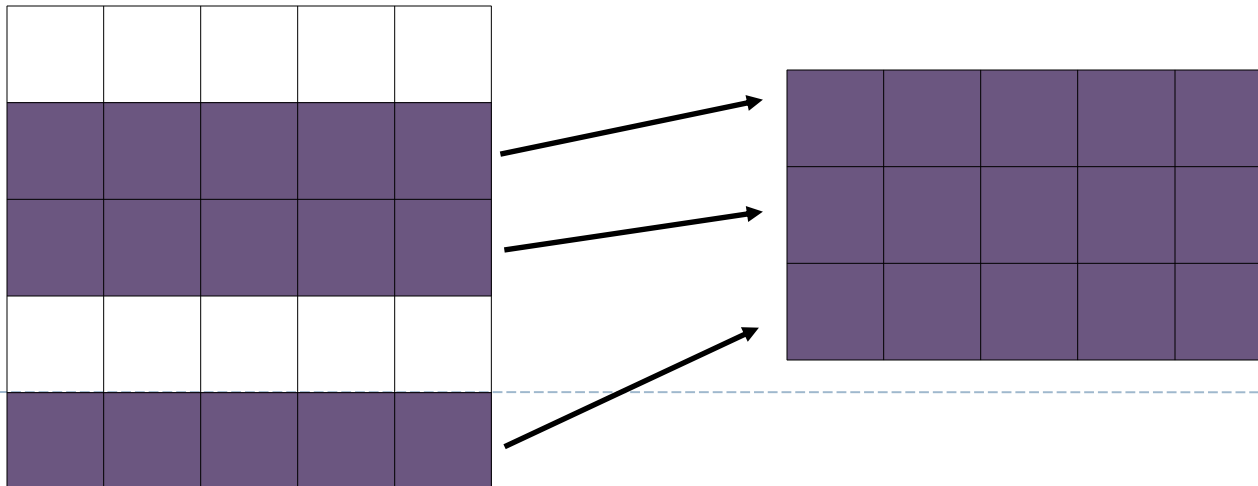
Operazioni relazionali

Selezione (select): genera una nuova relazione composta dalle sole tuple che soddisfano certe condizioni. Avrà lo stesso grado, ma cardinalità \leq

SELECT R WHERE cond

SELEZIONE di R PER cond

$\sigma_{\text{cond}} R$



Modello relazionale

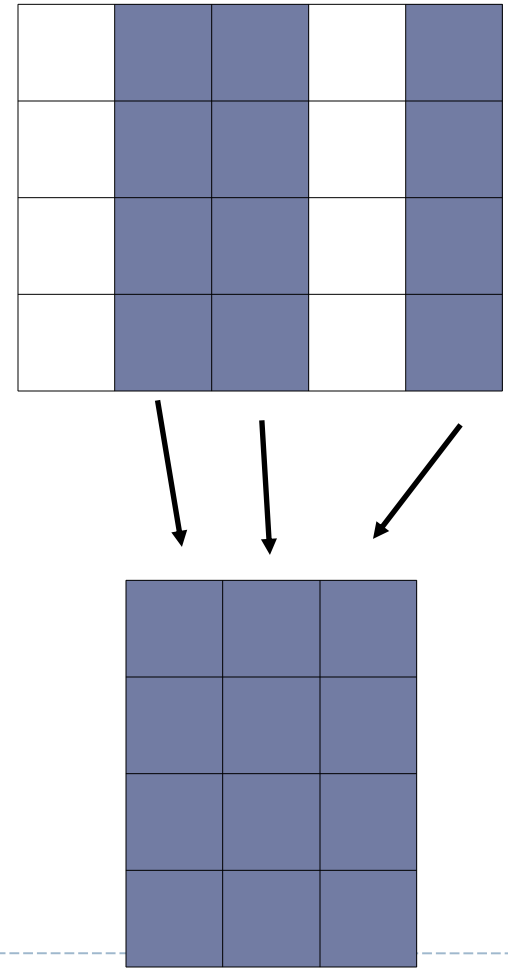
Operazioni relazionali

Proiezione (project): genera una nuova relazione composta dalle sole colonne relative ai campi di interesse. Se si generano 2 tuple uguali, ne rimane una sola. Avrà grado \leq , e cardinalità \leq (minore se tuple ripetute)

PROJECT R ON A_x, A_y, \dots, A_z

PROIEZIONE di R SU A_x, A_y, \dots, A_z

$\pi_{A_x, A_y, \dots, A_z} R$



Modello relazionale

Operazioni insiemistiche

Prodotto cartesiano (cross join) $R1 \times R2$: produce una relazione formata da tutte le tuple che è possibile ottenere combinando tutte le tuple di $R1$ con tutte le tuple di $R2$. La cardinalità è uguale al prodotto delle cardinalità, il grado dalla somma dei gradi

a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

a_2	d_2
a_5	d_3



a_1	b_1	c_1	a_2	d_2
a_1	b_1	c_1	a_5	d_3
a_2	b_2	c_2	a_2	d_2
a_2	b_2	c_2	a_5	d_3
a_3	b_3	c_3	a_2	d_2
a_3	b_3	c_3	a_5	d_3

Le basi di dati

Modello relazionale

Operazioni relazionali

Congiunzione (inner o theta join): genera una nuova relazione a partire da 2 relazioni aventi un attributo comune, composta dalla fusione delle tuple con valori che soddisfano un criterio tra attributi.

Se il criterio è l'uguaglianza si parla di **equi-join**: in questo caso due colonne coincidono. Se si eliminano le colonne ridondanti si parla di **equi-join naturale**.

Se l'operatore di confronto tra gli attributi non è per forza l'=
si parla di **inner join**.

Prima è eseguito un prodotto cartesiano tra le tabelle e poi una selezione.

Modello relazionale

Operazioni relazionali

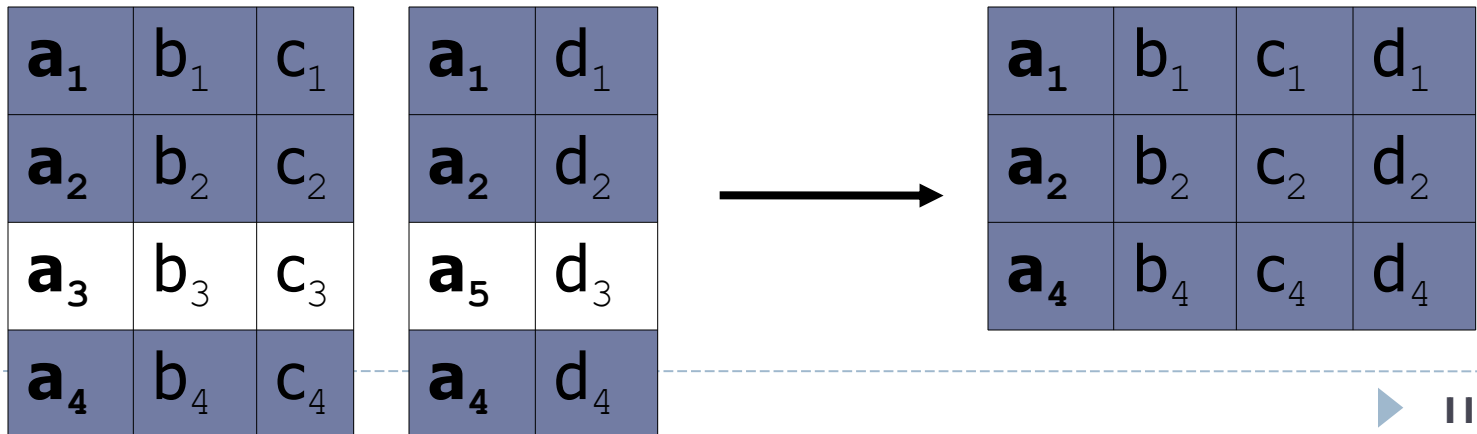
Congiunzione (join naturale):

$R.A_x \text{ JOIN } S.A_y$

CONGIUNZIONE di R SU A_x E di S SU A_y

$R_{A_x} \bowtie S_{A_y}$

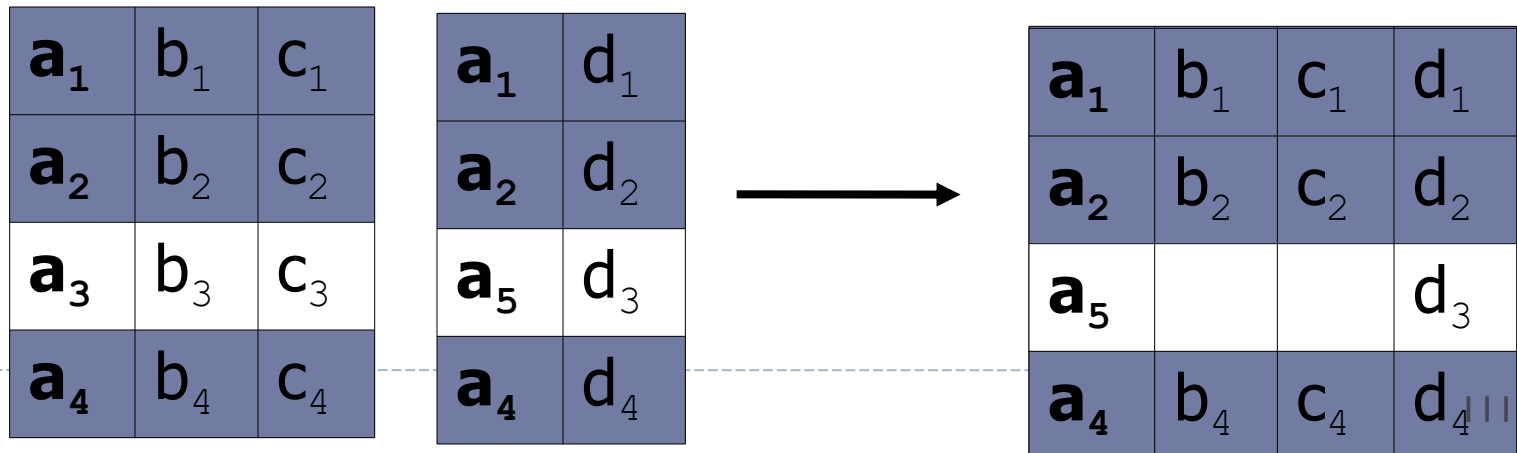
Avrà $\text{grado} = N1 + N2 - 1$, mentre la cardinalità non è prevedibile



Modello relazionale

Operazioni relazionali

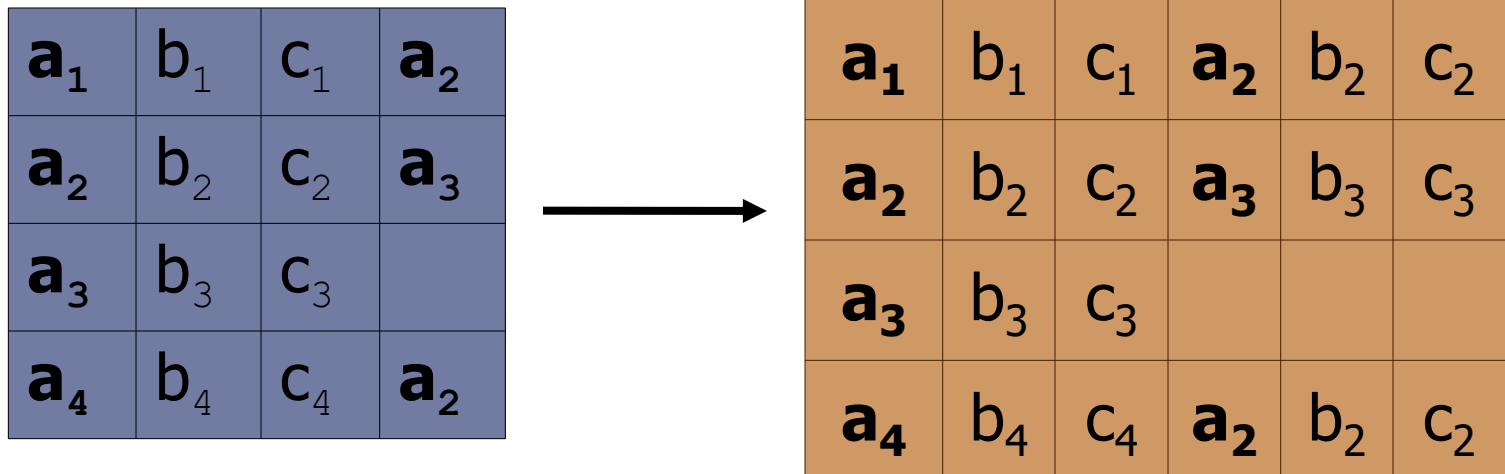
Join esterno (outer join): genera una nuova relazione a partire da 2 relazioni aventi un attributo comune, composta da tutte le tuple della prima relazione più quelle ottenute dalla fusione delle tuple con valori uguali per quell'attributo (**left join**). Oppure è formata da tutte le tuple della 2^a più quelle con i valori comuni (**right join**) o dall'unione di entrambi (**full join**)



Modello relazionale

Operazioni relazionali

Self Join (autocongiunzione): se esiste una associazione ricorsiva 1:N in una tabella, ogni tupla può essere in relazione con una tupla della tabella stessa. In questo caso ci sono due colonne con gli stessi domini (es. tabella persone, campo FiglioDi)

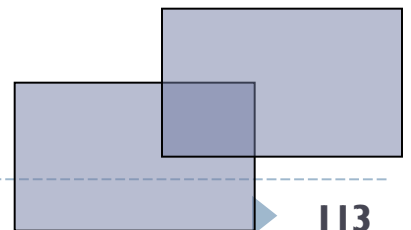


Modello relazionale

Operazioni insiemistiche

Se le relazioni hanno lo stesso grado e gli stessi domini si possono usare i seguenti operatori insiemistici per ottenere nuove relazioni, che avranno sempre lo stesso grado e gli stessi domini delle relazioni di partenza.

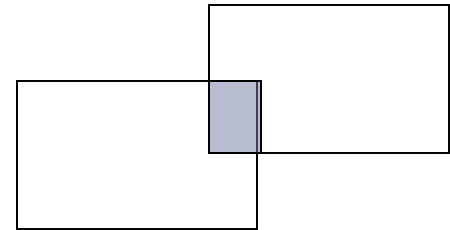
Unione $R \cup R2$: produce una relazione che contiene tutti gli elementi che appartengono all'una e/o all'altra (ci sono tutti gli elementi di entrambe, con quelli in comune ripetuti una sola volta)



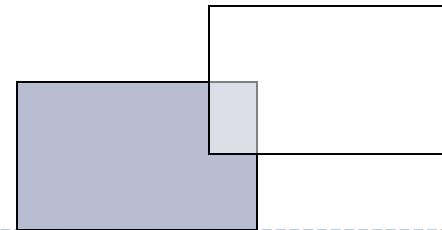
Modello relazionale

Operazioni insiemistiche

Intersezione $R1 \cap R2$: produce una relazione che contiene le sole tuple che appartengono ad entrambe le relazioni



Differenza $R1 - R2$: produce una relazione che contiene le tuple che appartengono alla prima relazione, ma non alla seconda. Non è simmetrica $R1 - R2 \neq R2 - R1$



La normalizzazione

La normalizzazione

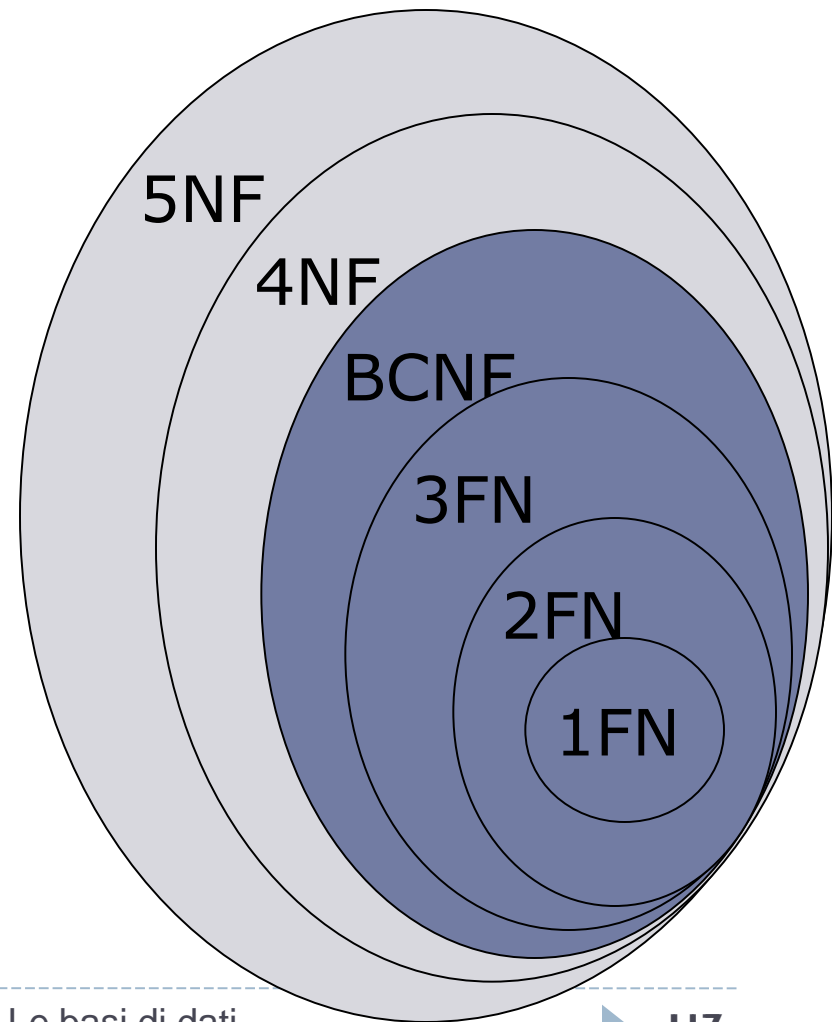
Introduzione

- ▶ È un procedimento di analisi e scomposizione di tabelle in più tabelle al fine di eliminare la ripetizione o ridondanza di dati, senza perdita di informazioni.
- ▶ A partire da delle relazioni definite a livello logico, ne crea delle altre corrispondenti a un livello di **forma normale** via via crescente (dalla prima si passa alla seconda e così via).
- ▶ L'aumento del numero delle tabelle a livello fisico rallenta l'aggiornamento e il reperimento dei dati, ma garantisce l'integrità degli stessi

La normalizzazione

Introduzione

- ▶ Le FN di ordine superiore contengono la stessa quantità di informazioni di quelle inferiori
- ▶ Solo la 1NF è richiesta dal modello relazionale ed è sufficiente la 3NF
- ▶ La 4FN e la 5FN servono a risolvere problemi legati alla presenza di attributi multivalore e a rendere minimo il numero degli attributi delle chiavi composte



La normalizzazione

Concetti base

Principio di minimalità: si sceglie quella con il minor numero di campi o che occupi meno spazio in memoria

- ▶ **Chiave candidata:** insieme minimo (non si considerano i sovrainsiemi, vedi superchiave) di attributi che identificano univocamente una tupla (ce ne possono essere molte)
- ▶ **Chiave primaria:** chiave candidata eletta a primaria secondo un principio di minimalità (ce n'è una sola)
- ▶ **Attributo non-chiave o non-primo:** campo che non fa parte di nessuna chiave primaria o candidata
- ▶ **Chiave :** chiave primaria o candidata
- ▶ **Superchiave o Sovrachiave:** chiave o soprainsieme di chiave

La normalizzazione

Dipendenza

- ▶ Esiste la **dipendenza funzionale FD** tra A e B se il valore di B dipende dal valore di A (che è il suo **determinante**). Ovvero se ad ogni valore della colonna A corrisponde un solo valore nella colonna B.
- ▶ Si indica così $A \rightarrow B$ (Es. $CAP \rightarrow Città$)
- ▶ Ovviamente tutti gli attributi sono funzionalmente dipendenti dalla chiave primaria o dalle chiavi candidate o dalle superchiavi
- ▶ Se il Y è composto e se $Z \subseteq Y$ allora è sempre vero $Y \rightarrow Z$. Se $A \rightarrow B$ e $A \subseteq C$ allora $C \rightarrow B$ (dipendenze banali)
- ▶ Se $A \rightarrow B$ e $B \rightarrow C$ allora $A \rightarrow C$, cioè C **dipende transitivamente** da A, con B non chiave

La normalizzazione

Quali problemi risolve

La ripetizione dei dati e la dipendenza creano:

- ▶ spreco di spazio
- ▶ **anomalie di aggiornamento**
 - ▶ **di modifica:** se modifico un valore ripetuto o determinante, lo devo modificare in tutte le occorrenze
 - ▶ **di cancellazione:** se cancello un valore ripetuto o determinante, lo devo cancellare in tutte le occorrenze. Inoltre potrei perdere informazioni
 - ▶ **di inserimento:** potrei non poter aggiungere delle nuove informazioni di campi dipendenti

La normalizzazione

Esempio

Esempio di relazione non normalizzata

Iscritti	CodiceCorso	Corso
Rossi Mario	1111	matematica
Verdi Lucia	1111	matematica
Bianchi Ugo	1212	logica

- ▶ Se cambiasse il codice dei corsi devo cambiarlo in tutte le tuple in cui compare
- ▶ Se elimino Bianchi Ugo perdo le informazioni sul corso di logica
- ▶ Per aggiungere un nuovo corso devo aver almeno un iscritto

La normalizzazione

- ▶ Si vuole fare in modo che all'interno delle tabelle non ci siano dipendenze, se non quelle con la PK.
- ▶ Per far ciò (a partire dalla 2NF) si decompongono le tabelle iniziali in tabelle più piccole attraverso proiezioni.
- ▶ La decomposizione deve essere senza perdita
 - ▶ di **informazioni** (si deve poter riottenere le tabelle iniziali attraverso dei join naturali)
 - ▶ di **dipendenze** (gli attributi coinvolti nella dipendenza iniziale devono comparire tutti insieme in uno degli schemi decomposti)ovvero deve permettere di ricostruire esattamente la relazione originaria

La normalizzazione

Problemi di perdita

<u>Impiegato</u>	<u>Progetto</u>	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Un impiegato deve operare su una sola sede e anche i progetti devono insistere su una sola sede

Impiegato → Sede

Progetto → Sede



La normalizzazione

Problemi di perdita

<u>Impiegato</u>	Sede		<u>Progetto</u>	Sede
Rossi	Roma		Marte	Roma
Verdi	Milano		Giove	Milano
Neri	Milano		Saturno	Milano
			Venere	Milano

<u>Impiegato</u>	<u>Progetto</u>	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
	Verdi	Saturno
	Verdi	Venere
	Neri	Giove
	Neri	Saturno
	Neri	Venere

**NON riottengo
la relazione di
partenza!!**

La normalizzazione

Problemi di perdita

<u>Impiegato</u>	<u>Progetto</u>	<u>Sede</u>
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

**Decomposizione
senza perdita di
informazioni**

<u>Impiegato</u>	<u>Sede</u>
Rossi	Roma
Verdi	Milano
Neri	Milano

<u>Impiegato</u>	<u>Progetto</u>
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere

La normalizzazione

Problemi di perdita

- ▶ Supponiamo di voler inserire una nuova ennupla che specifica la partecipazione dell'impiegato Neri, che opera a Milano, al progetto Marte
- ▶ Una istanza legale nello schema decomposto genera sullo schema ricostruito una soluzione non ammissibile
- ▶ Ogni singola istanza è (“localmente”) legale, ma il DB (“globalmente”) non lo è
 - ▶ Infatti il progetto “Marte” risulta essere assegnato a due sedi, in violazione del vincolo *Progetto* → *Sede*
- ▶ Problemi di consistenza dei dati si hanno quando la decomposizione “separa” gli attributi di una FD. Per verificare che la FD sia rispettata si rende necessario far riferimento a entrambe le relazioni.

La normalizzazione

Problemi di perdita

- ▶ Una decomposizione preserva le dipendenze se ciascuna delle dipendenze funzionali dello schema originario coinvolge attributi che compaiono tutti insieme in uno degli schemi decomposti
 - ▶ Nell'esempio la dipendenza *Progetto* → *Sede* non è conservata
- ▶ Se una FD non si preserva diventa più complicato capire quali sono le modifiche del DB che non violano la FD stessa

La normalizzazione

Prima Forma Normale 1NF

Una relazione è in **1NF** se rispetta i requisiti del modello relazionale:

1. Tutte le righe hanno lo stesso n° di colonne
2. Gli attributi sono atomici, né multivalore, né composti (un dato può considerarsi indivisibile se le eventuali sottoparti non hanno significato particolare nel contesto di interesse)
3. I valori di una colonna appartengono allo stesso dominio
4. Ogni tupla differenzia dalle altre per almeno un campo (esiste una PK)
5. L'ordine delle colonne è irrilevante

La normalizzazione

Seconda Forma Normale 2NF

Una relazione è in **2NF** se è in 1NF e:

- ▶ nessun campo non-chiave dipende funzionalmente da un sottoinsieme degli attributi di una chiave primaria composta

Es. Per una classe esiste la relazione

VotiMaterie(data, materia, voto, insegnante)

Insegnante dipende solo dalla materia, per cui si creano
2 relazioni

VotiMaterie(data, materia, voto)

Insegnanti_Materie(materia, insegnante)

La normalizzazione

Seconda Forma Normale 2NF

Procedimento per trasformare la tabella in 2NF
(si fa solo per tabelle con chiavi primaria composte):

1. si individuano gli attributi dipendenti da sottoinsiemi della chiave primaria composta
2. si crea una nuova tabella per ogni dipendenza individuata e si copiano le colonne determinante e dipendente
3. le colonne determinanti saranno le nuove PK
4. si cancellano dalla tabella di partenza le colonne dipendenti
5. le colonne determinanti nella tabella di partenza diventano chiavi esterne sulle nuove tabelle

La normalizzazione

Terza Forma Normale 3NF

Una relazione è in **3NF** se è in 2NF e:

- ▶ nessun campo non-chiave dipende funzionalmente da altri campi non-chiave, non ci deve essere dipendenza transitiva di un attributo non primo dalla chiave.
- ▶ TEOREMA ogni relazione può essere portata in 3NF
- ▶ Se c'è un solo attributo non primo automaticamente è in 3NF

Studente (matricola, cognome, nome, Via, CAP, Città)

La città dipende dal CAP, per cui si creano le relazioni

Studente (matricola, cognome, nome, Via, CAP)

CAPCittà (CAP, Città)

Una **relazione** è in **3NF** se per ogni FD non banale $X \rightarrow Y$ è vera una delle seguenti condizioni:
X è una superchiave della relazione
Y è un attributo primo

La normalizzazione

Terza Forma Normale 3NF

Procedimento per trasformare la tabella in 3NF:

1. si individuano gli attributi dipendenti da un attributo o combinazione di attributi non chiave
2. si crea una nuova tabella per ogni dipendenza individuata e si copiano le colonne determinante e dipendente (se uno o più determinanti si determinano reciprocamente $A \rightarrow B$ $B \rightarrow A$ non si divide lo schema)
3. le colonne determinanti saranno le nuove PK
4. si cancellano dalla tabella di partenza le colonne dipendenti
5. le colonne determinanti nella tabella di partenza diventano chiavi esterne sulle nuove tabelle

La normalizzazione

Forma Normale di Boyce-Codd

Una relazione è in **Forma normale di Boyce-Codd BCNF** se è in 1NF e:

- ▶ ogni determinante è una chiave candidata o superchiave. Non è possibile garantire sempre il raggiungimento della BCNF senza perdite

Esempio

data la relazione ABCD(A,B,C,D) se esiste la dipendenza non banale

$AD \rightarrow B$ è in 2NF e in 3NF (B non è un campo non chiave), ma non è in BCNF perché AD non è né chiave, né superchiave.

Analogamente per i casi

$AB \rightarrow C$ e $D \rightarrow A$

La normalizzazione

Forma Normale di Boyce-Codd

Lo schema

TEL(Prefisso,Numero,Località,Abbonato,Indirizzo)

ha i seguenti vincoli

- ▶ *Località,Numero* → *Prefisso,Abbonato, Indirizzo*
- ▶ *Prefisso,Numero* → *Località,Abbonato, Indirizzo* la scelgo come PK per principio di minimalità
- ▶ *Località* → *Prefisso*
- ▶ È in 2NF e in 3NF, in quanto *Prefisso* è primo, ma non è in BCNF

<u>Prefisso</u>	<u>Numero</u>	Località	Abbonato	Indirizzo
051	457856	Bologna	Rossi	Via Roma 8
059	452332	Modena	Verdi	Via Bari 16
051	987856	Bologna	Bianchi	Via Napoli 77
051	552346	Castenaso	Neri	Piazza Borsa 12
059	387654	Vignola	Mori	Via Piave 65

La normalizzazione

Forma Normale di Boyce-Codd

- ▶ Una soluzione consiste nel decomporre lo schema in
 - ▶ NUM_TEL(Numero, Località, Abbonato, Indirizzo)
 - ▶ PREF_TEL(Località, Prefisso)

<u>Numero</u>	<u>Località</u>	Abbonato	Indirizzo		<u>Località</u>	Prefisso
457856	Bologna	Rossi	Via Roma 8		Bologna	051
452332	Modena	Verdi	Via Bari 16		Modena	059
987856	Bologna	Bianchi	Via Napoli 77		Castenaso	051
552346	Castenaso	Neri	Piazza Borsa 12		Vignola	059
387654	Vignola	Mori	Via Piave 65			

La normalizzazione

Forma Normale di Boyce-Codd

- ▶ Se una relazione è in BCNF è anche in 2 e 3NF in quanto esclude che un determinante possa essere composto solo da una parte della chiave, come avviene per le violazioni alla 2NF, o che possa essere esterno alla chiave, come avviene per le violazioni alla 3NF. Non è vero il contrario.
- ▶ La 3NF garantisce di non perdere informazioni e dipendenze, non la BCNF, ovvero possono esserci relazioni che non possono essere normalizzate nella forma Boyce-Codd senza perdita di dipendenze funzionali (la BCNF non è senza perdita di dipendenze)

[Esempi](#)

La normalizzazione

Problema non superabile

- Ogni dirigente si trova in una sola sede. Un progetto può svilupparsi su più sedi, ma in ogni sede ha un solo dirigente. Pertanto le FD sono:

1. *Progetto, Sede* → *Dirigente*
2. *Dirigente* → *Sede*

Dirigente	<u>Progetto</u>	<u>Sede</u>
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

La normalizzazione

Problema non superabile

- ▶ La 2^a FD rispetta la 3NF perchè *Sede* è un attributo chiave, ma non è in BCNF perché *Dirigente* non è chiave candidata
- ▶ Però la 1^a FD coinvolge tutti gli attributi e quindi nessuna decomposizione può preservare tale dipendenza
- ▶ Quindi potrebbe non essere possibile decomporre in BCNF e preservare le FD
- ▶ Potrei pensare di cambiare $T1(\underline{\text{Progetto}}, \underline{\text{Dirigente}})$ e $T2(\underline{\text{Dirigente}}, \underline{\text{Sede}})$. Ma così potrei aggiungere *Verdi-Saturno*, ma violerebbe il vincolo che ogni progetto in una particolare sede ha un solo dirigente (in Milano risulterebbero Verdi e Neri sullo stesso progetto)

La normalizzazione

In pratica

- ▶ Se la relazione non è normalizzata si decompone in terza forma normale
- ▶ Si verifica se lo schema ottenuto è anche in BCNF
- ▶ Se uno schema non è in BCNF si hanno 3 alternative:
 - ▶ Si lascia così com'è, gestendo le anomalie residue (se l'applicazione lo consente)
 - ▶ Si decompone in BCNF, predisponendo opportune query di verifica (per verificare le dipendenze originarie vengano violate)
 - ▶ Si cerca di rimodellare la situazione iniziale, al fine di permettere di ottenere schemi BCNF

La normalizzazione

Procedimento

- ▶ Aggiungere ipotesi
- ▶ 1FN scrivere le considerazioni che portano a scegliere o meno al suddivisione o la non suddivisione. Scegliere la PK nei passi successivi
- ▶ Elencare le DF partendo dai campi singoli, aumentando la complessità
- ▶ Individuare chiavi candidate (saranno in ordine crescente di dimensione e tenendo conto dell'eventuale ordine logico per fare gli indici) e poi eleggere la PK
- ▶ 2FN giustificare se lo è, altrimenti indicare la definizione e indicare le DF che non la soddisfano e creare le nuove tabelle
- ▶ 3FN come sopra
- ▶ BCNF come sopra, valgono solo più le DF che hanno tutti i campi determinanti in una tabella con almeno un campo dipendente nella stessa tabella
- ▶ Sottolineare le tabelle risultato