13. IL MONDO DELLE BASI DI DATI

DEF: Una **base di dati** (in inglese **database**) è una raccolta di dati progettati per essere utilizzati in maniera ottimizzata da differenti applicazioni e da utenti diversi.

Una base di dati, per poter essere definita tale, deve essere:
- **sicura**: ossia deve essere progettata in modo da impedire che essa possa essere danneggiata da eventi accidentali (come crash di sistema) o da accessi non autorizzati;
- **integra**: ossia deve essere garantito che le operazioni effettuate da utenti autorizzati non possano provocare una perdita di consistenza dei dati;
- **consistente**: ossia i dati in essa contenuti devono essere significativi ed effettivamente utilizzabili nelle applicazioni dell'azienda per cui è stata progettata;
- **condivisibile**: ossia applicazioni ed utenti diversi devono poter accedere, secondo opportune modalità, ai dati comuni;
- **persistente**: ossia deve avere un tempo di vita che non è limitato a quello delle singole esecuzioni dei programmi che la utilizzano (il contrario dei dati gestiti in memoria centrale);
- **efficiente**: l'utilizzo delle risorse deve essere ottimizzato riguardo ai ben noti parametri *tempo* (efficiente utilizzo della CPU) e *spazio* (efficiente uso della memoria).

Da questa definizione segue che una base di dati contiene i dati di molti utenti e non di uno solo.

Ogni utente in generale sarà interessato solo ad una piccola parte della raccolta completa di dati.

I dati comuni a più utenti sono presenti una volta sola all'interno della raccolta dati (sono messi in comune) evitando inutili duplicazioni ed ottenendo una serie di vantaggi che verranno dettagliatamente chiariti in seguito.

SISTEMA INFORMATIVO E SISTEMA INFORMATICO

Distinguiamo ora tra **sistema informativo** e **sistema informatico**.

DEF: Un **sistema informativo** è un insieme organizzato di strumenti automatici, procedure manuali, risorse umane e materiali, norme organizzative, orientato alla gestione delle informazioni rilevanti per un'organizzazione.

La parte del sistema informativo di un'organizzazione che può essere automatizzata è chiamata *sistema informatico*.

DEF: Un **sistema informatico** (spesso chiamato anche ***EDP***: ***E****lectronic* ***D****ata* ***P****rocessing*) è quel sottoinsieme del *sistema informativo* dedicato alla gestione automatica di informazioni rappresentate mediante *dati digitali*.

La parte di sistema informativo che non fa parte del sistema informatico è costituita dagli operatori, dagli archivi cartacei, dalla rete di informazioni verbali, dalle tabelle di consultazione, dalle procedure organizzative aziendali.

N.B. Nel prosieguo del nostro discorso ci riferiremo sempre **a sistemi informativi basati sull'uso di sistemi informatici**.

***Esempio 1**: Consideriamo come organizzazione il team di vendita di una piccola ditta artigianale.*
*Le informazioni rilevanti ossia il suo **sistema informativo** per questa organizzazione è l'insieme dei numeri telefonici dei clienti effettivi e di quelli potenziali. Fanno parte del sistema informativo dell'organizzazione:*
- un insieme di numeri telefonici molto piccolo tenuto a mente dal titolare;
- un altro insieme di numeri telefonici più sostanzioso trascritto si di un agenda clienti;
- un elenco telefonico della città dove reperire il numero telefonico di potenziali clienti;
- alcuni documenti cartacei informali (quali biglietti da visita, ricevute di ristoranti ed alberghi, fatture, ordini, etc.)
*Il **sistema informatico** dell'organizzazione in questione potrebbe essere rappresentato da un'agenda elettronica nella quale memorizzare i numeri di telefono dei clienti.*

***Esempio 2**: Consideriamo come organizzazione una grande compagnia di trasporti aerea.*
*Il suo **sistema informativo** è costituito da un insieme di procedure che permettono un efficiente prenotazione dei posti sui voli, di gestire il personale di terra e di volo, gestire la manutenzione dei velivoli, gestire la contabilità, gestire il magazzino dei pezzi di ricambio, gestire il rifornimento degli aerei, etc.*
*Il suo **sistema informatico** è costituito dagli archivi elettronici in cui sono memorizzati i dati di interesse relativi ai clienti, agli aerei, al personale, dalle componenti fiche che costituiscono il supporto di memorizzazione degli archivi, dalle procedure di interrogazione per la ricerca di informazioni, dalle reti di comunicazione tra i terminali degli operatori etc.*

DATI E INFORMAZIONI: SCHEMI ED ISTANZE

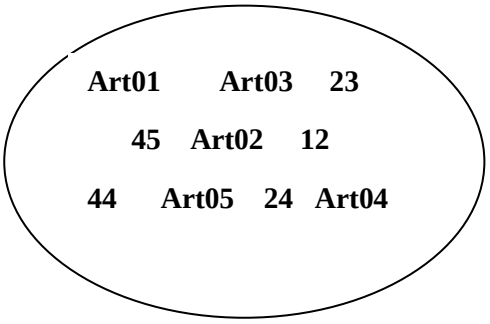
Ribadiamo ancora una volta la differenza tra i termini **dato** ed **informazione**.

DEF: Il termine **dato** deriva dal termine latino *datum* (plurale *data*) ossia *fatto*. Lo scopo dei dati è quello di codificare in vari modi i fatti ritenuti importanti nell'ambito di una organizzazione.

DEF: Una **informazione** invece è l'aumento di conoscenza che può essere acquisita (o *inferita*) dai dati.

Da questa considerazione segue che i dati non possono essere ritenuti utili (ossia non danno informazione) fino a quando non si fornisce una **chiave di interpretazione** (o semplicemente interpretazione) che permetta di comprendere il loro **significato** (o semantica) ossia i fatti che essi codificano.

Esempio:
Consideriamo il seguente insieme di dati:



Essi non forniscono alcuna informazione utile finchè non si definisce cosa ciascun dato rappresenta e le eventuali relazioni esistenti tra i dati stessi.
*Il dato **23** senza chiave di interpretazione che gli dia un significato non è una informazione per nessuno. Lo stesso dato **23** con chiave di interpretazione "Articoli in magazzino" può assumere il significato "quantità disponibile per un certo articolo".*
*Il dato **Art01** senza chiave di interpretazione che gli dia un significato non è una informazione per nessuno. Lo stesso dato **Art01** con chiave di interpretazione "Articoli in magazzino" può assumere il significato "Codice articolo".*

Quindi è possibile dare un significato al precedente insieme (ossia fornire una chiave di interpretazione) attraverso la seguente forma tabellare:

Articoli in magazzino	
Codice articolo	***Quantità***
Art01	*23*
Art02	*45*
Art03	*12*
Art04	*44*
Art05	*24*

DEF: Chiameremo **schema** (o **intensione)** la chiave di interpretazione dei dati ovvero il *significato* (o *semantica*) che si attribuisce al dato per ricavare l'informazione da esso portata.

DEF: Chiameremo **categoria** un insieme di dati aventi la stessa chiave di interpretazione ovvero lo stesso schema

DEF: Chiameremo **istanza di uno schema** o **estensione** i valori assunti da uno schema *in un certo istante di tempo*.

Parleremo di **significato intensionale** dei dati riferendoci al contenuto informativo (significato) dei dati.

Parleremo di **significato estensionale** dei dati riferendoci ai valori che può assumere uno schema. *in un certo istante di tempo*.

Nell'esempio precedente dunque la tabella

Articolo in magazzino	
Codice articolo	***Quantità***

*rappresenta lo **schema** dei dati proposti mentre "Articolo in magazzino" rappresenta la **categoria**. Di essa fa parte la seguente **istanza** dello schema precedente (cinque articoli e le loro rispettive quantità)*

Art01	*23*
Art02	*45*
Art03	*12*
Art04	*44*
Art05	*24*

In sintesi:

Lo **schema** non varia nel tempo.

L'**istanza di uno schema** varia nel tempo poiché possono essere eseguite modifiche, inserimenti, cancellazioni.

N.B. Se ipotizziamo che lo schema non possa essere modificato, dobbiamo considerare la sua modifica come la definizione di un nuovo schema ovvero la definizione di una nuova chiave di interpretazione dei dati, ossia di una nuova base di dati.

Riprendendo il nostro esempio abbiamo che:

categoria		*Articolo in magazzino*	
***schema** (o significato):*		*Codice articolo*	*Quantità*
istanza:	*primo articolo*	*Art01*	*23*
	secondo articolo	*Art02*	*45*
	terzo articolo	*Art03*	*12*
	quarto articolo	*Art04*	*44*
	quinto articolo	*Art05*	*24*

***Il significato intensionale del dato 23** è: "quantità in magazzino dell'articolo Art01"*

In una **base di dati** sono sempre presenti:
- un insieme di **categorie (ossia di schemi)**;
- un insieme di **regole** che le categorie (gli schemi) devono soddisfare;
- un insieme di **autorizzazioni** che definiscono le operazioni consentite a determinati utenti.

DEF: Con il termine **istanza** (o **occorrenza)** di una *base di dati* si intende l'insieme delle *istanze* di tutte le sue *categorie* in un determinato istante di tempo.

N.B. Spesso per base di dati si intende una particolare istanza o occorrenza di una base di dati

IL MODELLO DEI DATI

La più diffusa classificazione delle basi dati si basa sul modello dei dati che essa supporta.

Un **modello di dati** è un insieme di concetti e di costrutti utilizzati per organizzare i dati di interesse e descriverne la struttura e la dinamica (associazioni e vincoli che devono rispettare)

Nella teoria delle basi dati i modelli si distinguono in

Modelli concettuali: Essi permettono di **rappresentare i concetti** (dati) in modo indipendente da ogni sistema cercando di descrivere i concetti del mondo reale. Quindi mettono in evidenza i concetti presenti in una base di dati piuttosto che la struttura con cui tali concetti possono essere rappresentati nella memoria dell'elaboratore. Uno dei più noti è il **modello ER** (*Entity-Relationship*)

Modelli logici: Essi consentono una specifica **rappresentazione dei dati**. Sono detti logici per sottolineare il fatto che le strutture utilizzate in questi modelli, pur essendo astratte, riflettono una particolare organizzazione (*tabelle*, *alberi*, *grafi*, *oggetti*, etc.) di rappresentare. I principali modelli logici sono:

(*) Modello gerarchico: Ha caratterizzato i primi DBMS verso la metà degli anni 60. I dati sono organizzati in record collegati tra loro mediante la classica struttura ad albero (*n-ario*). Ogni record del database deve avere un unico padre. Possono esserci più record, su alberi diversi, che rappresentino la stessa informazione andando a creare così vari problemi quali ad

esempio la *ridondanza* dei dati e rendendo necessari continui controlli di *consistenza*. La manutenzione appare pertanto molto complessa (ad esempio per cancellare un dato occorre cancellare tutti i record che ne contengono l'informazione ed anche per aggiornare un dato occorre provvedere all'aggiornamento di tutti i record che ne contengono l'informazione).
N.B. In questo modello i programmi dipendono dalle strutture dati utilizzate e quindi non è possibile modificare tali strutture dati senza modificare i programmi che le utilizzano.

(*) Modello reticolare: E' detto anche *modello a rete* o *modello Codasyl* ed è stato ideato nel 1973 e perfezionato nel 1978**.** Si basa sulle strutture dati a reticolo (ad esempio i *grafi*). Deriva da quello gerarchico ma ne supera la rigidità della struttura ad albero. I record sono legati tra loro tramite strutture ad anello che permettono all'utente di accedere ai dati in maniera più semplice senza i vincoli rigidi della struttura gerarchica. Ogni nodo può essere un punto di partenza per raggiungere un campo. Un record può avere uno o più record padre permettendo di risolvere il problema della ridondanza dei dati.
N.B. Anche in questo modello i programmi dipendono dalle strutture dati utilizzate ed inoltre le stesse strutture dati utilizzate sono di una complessità superiore. Addirittura per modificare un database con questa struttura occorre cancellarlo per poi ricrearlo ex-novo.

(*) Modello relazionale: Inventato da Codd nel 1970 si bada sul concetto matematico di *"relazione tra insiemi"* intesa come sottoinsieme finito del loro prodotto cartesiano.
Studieremo dettagliatamente in seguito proprio il modello relazionale.

(*) Modello ad oggetti: nasce come evoluzione del modello relazionale estendendo alla basi di dati il concetto di programmazione ad oggetti. Moderno e promettente è stato introdotto per superare alcune caratteristiche dei database relazionali che si sono mostrati inadatti in alcune sfere applicative quali le applicazioni multimediali.

Indipendentemente dai modelli appena descritti e che verranno utilizzati quando si parla di basi dati è possibile incontrare sempre i seguenti quattro macroargomenti:

1) **metodologie di progettazione** di una base dati: essi permettono di definire lo *schema* di una base di dati a partire dalle *specifiche dell'utente*;

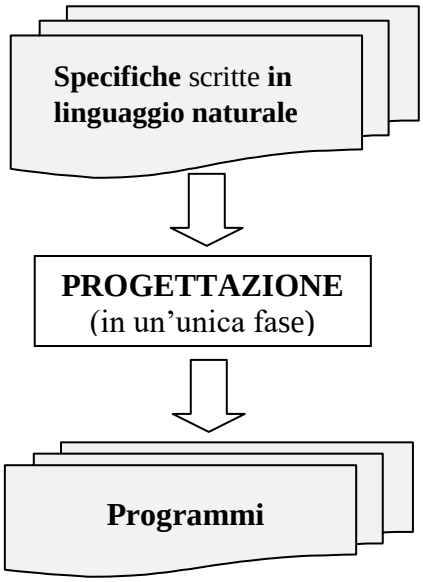
2) **sistemi di gestione** delle basi dati **o DBMS** (**D**ata **B**ase **M**anagement **S**ystem): sono prodotti software che permettono di *interagire* con una base dati consentendo opportune *operazioni agli utenti autorizzati* nel rispetto di *regole* prestabilite. Tali richieste non devono violare alcun *vincolo* sui dati.

3) **linguaggi** per le basi di dati: sono quei linguaggi con i quali *creare* le basi di dati, *interrogare* le basi di dati, *modificare* sia gli *schemi* sia le *istanze* di una base di dati;

4) **utenti** delle basi di dati: sono coloro che possono operare sui dati secondo le autorizzazioni ed i privilegi loro assegnati

LA PROGETTAZIONE DI UNA BASE DATI

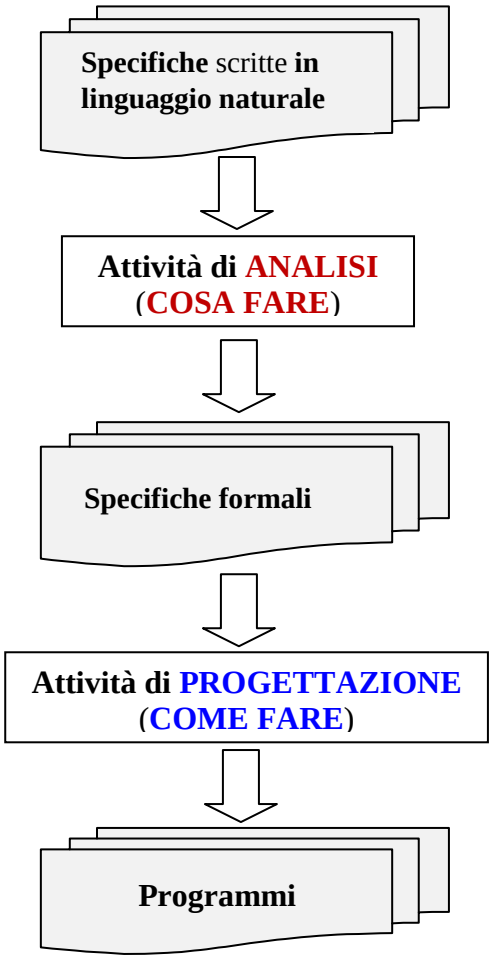
Una **metodologia di progettazione di una base dati** può essere vista come un insieme di:
- **attività o fasi** tra loro collegate;
- **prodotti** intermedi e finali di tali *attività*;
- **criteri** di verifica della qualità di tali *attività* e *prodotti*



In un primo momento della storia delle basi di dati avveniva la *trasformazione delle specifiche (scritte in linguaggio naturale) in programmi attraverso una sola fase di progettazione*.

Vi erano però notevoli problemi dovuti sia alla scarsa documentazione della struttura del programma sia alla mancanza di una visione di insieme del progetto con conseguenti gravi difficoltà nella manutenibilità dello stesso

---



In un secondo momento della storia delle basi di dati le *specifiche della realtà di interesse vengono trasformate in specifiche formali dopo una prima fase di analisi*.

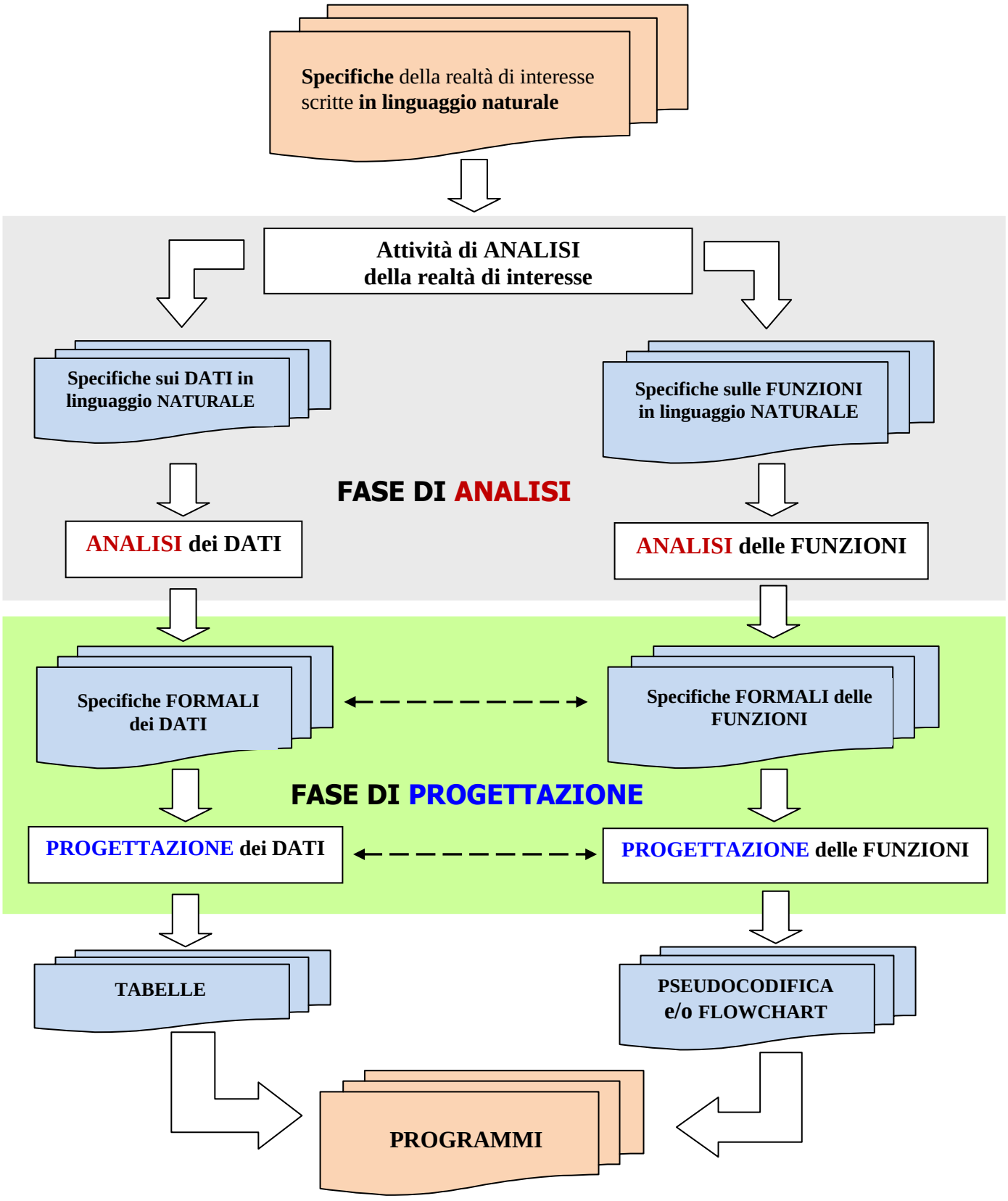
In questa fase di analisi si determina **COSA ciascun programma deve fare indipendentemente da come lo farà** e lo si formalizza nel documento di specifiche formali.

In questo caso con il termine *"formali"* si intende *"astratte"* ossia si descrivono in modo **astratto** (ossia in modo indipendente dalla tecnologia e dal particolare linguaggio di programmazione con il quale si realizzeranno i programmi) sia le tipologie di **dati** sia le **funzioni** che opereranno su tali dati.

Nella fase di progettazione si determinerà **COME ciascun programma dovrà fare quanto stabilito in fase di analisi** ossia, in altre parole, in che modo il documento di specifiche formali verrà trasformato in programmi.

Inizialmente la distinzione tra *analisi* e *progettazione* si era affermata più per le *funzioni* che per i dati (approccio asimmetrico) e solo in un secondo tempo ci si è accorti che la risorsa centrale erano proprio i **dati**.

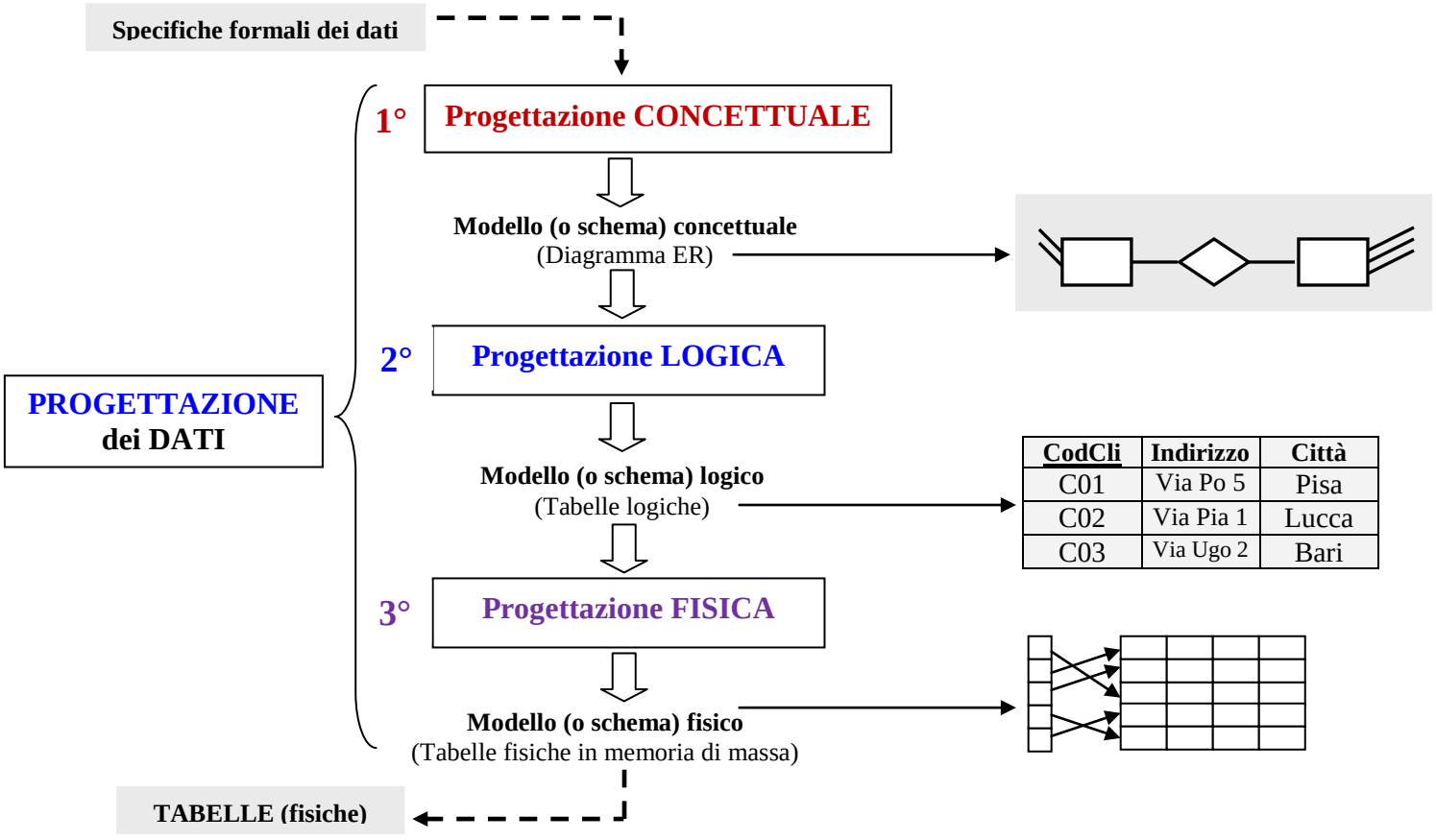
Ecco che si è giunti **all'attuale approccio progettuale**, detto **con pari dignità,** proprio perché vengono attivate due fasi distinte con una *verifica di completezza reciproca* (rappresentata dalla doppia freccia tratteggiata) tra funzioni e dati.



Alla fine di tale attività sarà indispensabile verificare (la sopra citata *verifica di completezza*)che:
- **per ogni funzione** definita nel documento delle specifiche formali relativo siano rappresentate le tipologie di dati da essa manipolate;
- **per ogni tipologia** dei dati definita nel documento delle specifiche formali relativo siano definite tutte le funzioni che su di essa operano.

LE TRE FASI DELLA PROGETTAZIONE DI UNA BASE DATI

L'insieme di attività che costituiscono l'attività di **progettazione** di una base di dati consta di tre distinte attività di progettazione:
1) **progettazione concettuale**: ha lo scopo di *costruire* e *definire* una rappresentazione astratta corretta e completa della realtà di interesse in modo assolutamente indipendente dalla sua successiva implementazione (quindi indipendente dal DBMS che poi verrà più avanti scelto)
L'input di tale fase è il documento delle *specifiche formali dei dati.*
L'output di tale fase è uno *schema concettuale* ossia una rappresentazione astratta ed il più possibile formale della realtà (nel nostro caso è il **diagramma ER**).
2) **progettazione logica**: ha lo scopo di *trasformare* lo schema concettuale (ancora astratto ed indipendente da un DBMS) in uno *schema logico* ovvero in una rappresentazione efficiente rispetto alle strutture di un DBMS (un esempio è una descrizione tramite tabelle del modello relazionale).
L'input di tale fase è lo *schema concettuale* (nel nostro caso *il diagramma ER)* della fase di progettazione concettuale.
L'output di tale fase è uno *schema logico* (nel nostro caso riassumibile con *relazioni* rappresentate da tabelle logiche).
3) **progettazione fisica**: ha lo scopo di *implementare* lo schema logico *definendo tutti gli aspetti fisici di memorizzazione e rappresentazione in memoria di massa.*
L'input di tale fase è lo *schema logico* (nel nostro caso sono le tabelle logiche individuate nella fase di progettazione logica).
L'output di tale fase è lo *schema fisico* ossia l'*implementazione in memoria di massa* di tali tabelle.
Spesso si parla di **modello logico-fisico** intendendo questa come un'unica fase realizzativa.



IL DBMS

DEF. Il **DBMS (Data Base Management System)** è il software che offre, sulla base delle specifiche utente, partendo da un progetto concettuale tradotto poi in un modello logico, la possibilità di costruire e gestire una base di dati su una memoria di massa.

Nel caso specifico di database relazionali si parla di **RDBMS** (**Relational Data Base Management System**.)

Quindi il **DBMS** rappresenta **un'interfaccia** tra gli sviluppatori e gli utenti del data base ed il sistema di elaborazione.
Il **DBMS** è un sistema **attivo** mentre la base di dati è l'oggetto **passivo** su cui opera il DBMS.

Prima di analizzare le caratteristiche di un DBMS occorre rispondere alla seguente domanda:

Perché utilizzare i database piuttosto che gli archivi?

Le tecniche di gestione delle basi di dati sono nate per superare i limiti posti dalle tradizionali tecniche di organizzazione degli archivi informatici.

Primo problema: i dati contenuti negli archivi tradizionali come sappiamo non sono organizzati in modo integrato tra loro.
A causa di ciò, quindi, esistono dati ripetuti in più archivi (**ridondanza dei dati**).
A causa della ridondanza dei dati nascono problemi di **incongruenza dei dati** (ad esempio se modifico un record tale modifica dovrà essere apportata su tutti gli archivi ove esso è presente)
A causa dell'incongruenza dei dati nascono problemi di **inconsistenza dei dati** (essi diventano inaffidabili poiché si potrebbe anche non risalire al dato corretto).
Secondo problema: nella gestione tradizionale degli archivi, i classici linguaggi di programmazione richiedono che all'interno di ciascun programma vengano specificati gli archivi utilizzati e la struttura del loro record. Una qualunque modifica della struttura di un record comporta la modifica di tutti i programmi che lo utilizzano.
Terzo problema: nella gestione tradizionale degli archivi, l'operazione di accesso ai dati è strettamente correlata alla organizzazione assegnata agli stessi. Tale organizzazione vincola il programmatore a sviluppare le sole operazioni consentite per quel tipo di organizzazione.
Quarto problema: nella gestione tradizionale degli archivi vi erano i problemi connessi alla **concorrenza**: in uno stesso istante vi possono essere attivi vari programmi che operano su uno stesso file accedendo ai suoi dati. Se due programmi leggono uno stesso valore e lo modificano in contemporanea solo l'ultima operazione di modifica sarà registrata con il risultato che una operazione di scrittura andrà persa.

Le funzioni principali cui un **DBMS** deve assolvere e garantire sono:
a) Gestione della base dati: Il DBMS deve permettere le operazioni di creazione, inserimento, aggiornamento, cancellazione ed interrogazione della base dati. Deve anche permettere l'interfacciamento tra i programmi scritti con i classici linguaggi di programmazione e la base di dati attraverso le sue specifiche funzionalità e garantire un accesso ai dati attraverso semplici interfacce grafiche semplici ed intuitive anche per utenti non specialistici.

b) Persistenza e consistenza dei dati: Il DBMS deve esser in grado di conservare intatto il contenuto della base di dati, permettendone anche la ricostruzione, in caso di malfunzionamento del sistema di elaborazione sul quale è in funzione (*persistenza*). A tale scopo quasi tutti i DBMS sono dotati di funzionalità di *backup* e *restore* ossia salvataggio e ripristino dei dati.
Deve anche garantire l'integrità dei dati nei casi potenzialmente pericolosi di accesso concorrente in lettura/scrittura (*consistenza*).

c) Privatezza e sicurezza dei dati: Il DBMS deve permettere che ciascun utente sia reso identificabile univocamente attraverso opportune credenziali con le quali effettuare l'accesso alla base dati. Inoltre deve essere in grado di fornirgli i giusti privilegi.
Pertanto, una volta riconosciuto, l'utente sarà abilitato a svolgere solo specifiche azioni sui dati attraverso opportuni meccanismi di autorizzazione.

d) Supporto alle transazioni: Innazitutto una **transazione** è una sequenza di operazioni "fisiche" da effettuare su di una base di dati che dal punto di vista funzionale devono essere considerate come facenti parte di un'unica operazione "logica".
Esempio di transazioni possono essere l'effettuazione di un bonifico o di un acquisto on line.

Una **transazione** può ovviamente concludersi con un successo o con un insuccesso.

Nel caso di **successo** (ossia in caso di esito positivo di tutte le operazioni fisiche costituenti) le modifiche alla base dati contenute nella transazione **devono** essere rese **permanenti** altrimenti, in caso di **insuccesso** (ossia in caso di fallimento di una delle operazioni fisiche costituenti) la base dati **deve ritornare allo stato precedente** l'avvio della transazione stessa garantendo così la sua integrità.

Il DBMS deve pertanto garantire che tutte le operazioni che costituiscono una transazione siano correttamente e completamente eseguite (*COMMIT*) oppure che non ne sia eseguita alcuna (*ROLLBACK*).

e) Gestione del dizionario dei dati: Il DBMS deve permettere di gestire il dizionario o catalogo di una base di dati contenente i **metadati** ossia tutte quelle informazioni "a corredo" che descrivono gli oggetti di una base dati (esempio il nome del database, i nomi delle tabelle che ne fanno parte, i nomi degli attributi di ogni tabella, i vincoli da realizzare, le autorizzazione agli accessi, gli utenti, etc.).
Tali oggetti sono organizzati e gestiti anch'essi in modo relazionale.
Gli utenti autorizzati possono accedere ai *metadati* con le stesse modalità con cui operano per accedere ai dati.

Un **DBMS** deve inoltre essere progettato per essere:

- **efficiente**: ossia capace di svolgere le operazioni richieste minimizzando l'insieme di risorse (tempo e spazio) a sua disposizione. L'efficienza dipende non solo dall'implementazione del DBMS ma anche dalla bontà di realizzazione della base di dati da parte dei progettisti;

- **efficace:** ossia deve essere in grado di rendere produttive e semplici le attività richieste dagli utenti.

I LINGUAGGI PER LE BASI DI DATI

L'utente usa un **DBMS** attraverso appropriati **comandi** grazie ai quali viene instaurata una **comunicazione** con il sistema di elaborazione che gestisce la basi di dati o database.
L'insieme di tali comando costituisce un vero e proprio linguaggio, anzi costituisce **un insieme di vari linguaggi.**

- **DDL** (**D**ata **D**efinition **L**anguage ossia **Linguaggio di definizione dei dati**).
E' il linguaggio utilizzato per la descrizione dei dati, delle tabelle, delle interfacce utente e delle *viste* (sottoschemi). Permette all'utente di costruire la struttura fisica del database partendo dallo schema logico.

- **DMCL** (**D**evice **M**edia **C**ontrol **L**anguage ossia **Linguaggio di controllo dei supporti di memorizzazione dei dati**).
E' il linguaggio che permette alla struttura fisica del database di far riferimento alle particolari unità di memoria di massa utilizzate dal sistema.

- **DML** (**D**ata **M**anipulation **L**anguage ossia **Linguaggio per il trattamento dei dati**).
E' il linguaggio utilizzato per accedere al database per effettuare inserimenti, modiche e cancellazioni.

- **DCL** (**D**ata **C**ontrol **L**anguage ossia **Linguaggio di controllo dei dati**)
E' il linguaggio utilizzato per stabilire i vincoli di integrità, oltre che per stabilire accessi e permessi.

- **QL** (**Q**uery **L**anguage ossia **Linguaggio di interrogazione dei dati**)
E' il linguaggio interattivo che permette di interrogare il database per ritrovare le informazioni relative alla chiave di ricerca impostata dall'utente.

N.B.

La diffusione attuale del modello relazionale ha favorito la nascita del linguaggio per basi di dati ossia di un insieme di comandi che consentono la gestione globale del database raggruppando in se le funzioni dei linguaggi DDL, DMCL, DML, DCL e QL.

GLI UTENTI DI UNA BASE DI DATI

E' possibile individuare le seguenti **classi di utenza** di una base di dati:

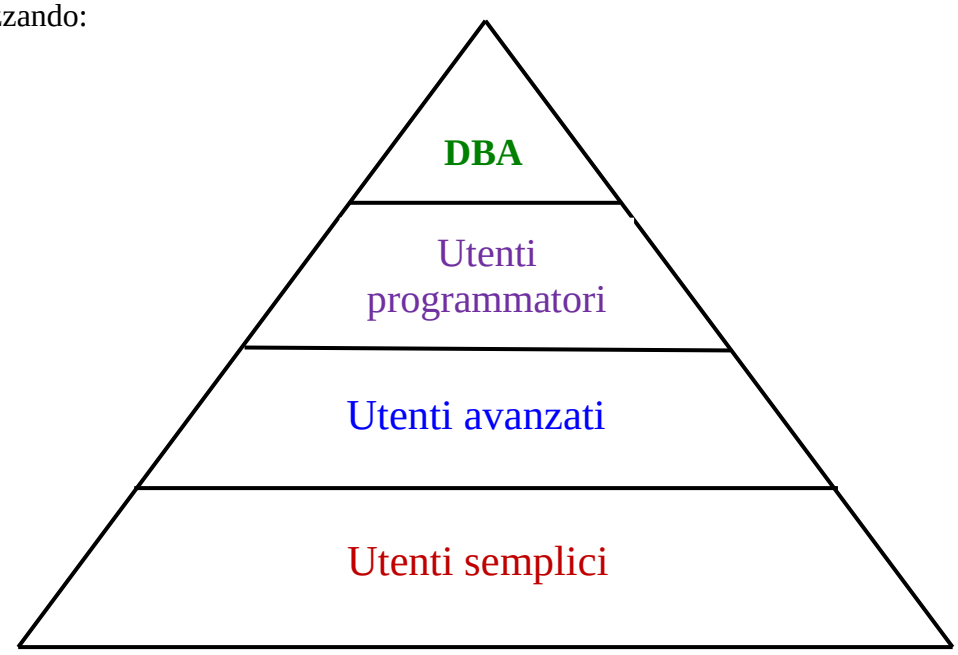
- **utenti semplici:** detti anche operatori sono coloro che utilizzano il database per inserire, modificare e cancellare in modo *"protetto"* seguendo i programmi applicativi creati dagli utenti programmatori;

- **utenti avanzati:** sono coloro che accedono alla base di dati per effettuare interrogazioni anche complesse senza che vi sia bisogno dell'intervento di un utente programmatore che scriva un programma ad hoc. Tali interrogazioni vengono effettuate grazie alla conoscenza approfondita di un linguaggio di interrogazione non procedurale come *l'SQL*. Tipicamente sono persone che fanno parte dello staff dirigenziale e/o di livello medio di un'azienda.

- **utenti programmatori:** sono coloro che costruiscono specifiche applicazioni per permettere agli altri utenti un'interazione *"controllata"* sulla base di dati. Utilizzano il DML del DBMS.
La distinzione tra utente e programmatore sta via via scomparendo a causa della sempre maggiore interattività e facilità di uso dei DBMS che permettono anche ad un utente meno esperto di creare tabelle, effettuare interrogazioni, etc.

- **amministratore o DBA (DataBase Administrator):** è colui che progetta e si occupa della manutenzione della base di dati. Gestisce inoltre:
  (-) la definizione iniziale e la modifica dello **schema** della base di dati (è responsabile della *correzione*, dell'*ampliamento* e dell'*adeguamento* dello schema logico alle nuove esigenze degli utenti della base di dati);
  (-) la definizione iniziale e la modifica delle **viste logiche** sui dati (è responsabile della *protezione* e della *riservatezza* dei dati);
  (-) la definizione iniziale e la modifica delle **politiche di accesso** ai dati (è responsabile dell'*assegnazione dei privilegi di accesso* agli utenti della base di dati);
  (-) la definizione iniziale e la modifica delle **politiche di salvataggio e ripristino** (**backup** e **restore**) ai dati (è responsabile dell'immediato *ripristino dei dati* a seguito di eventi disastrosi).

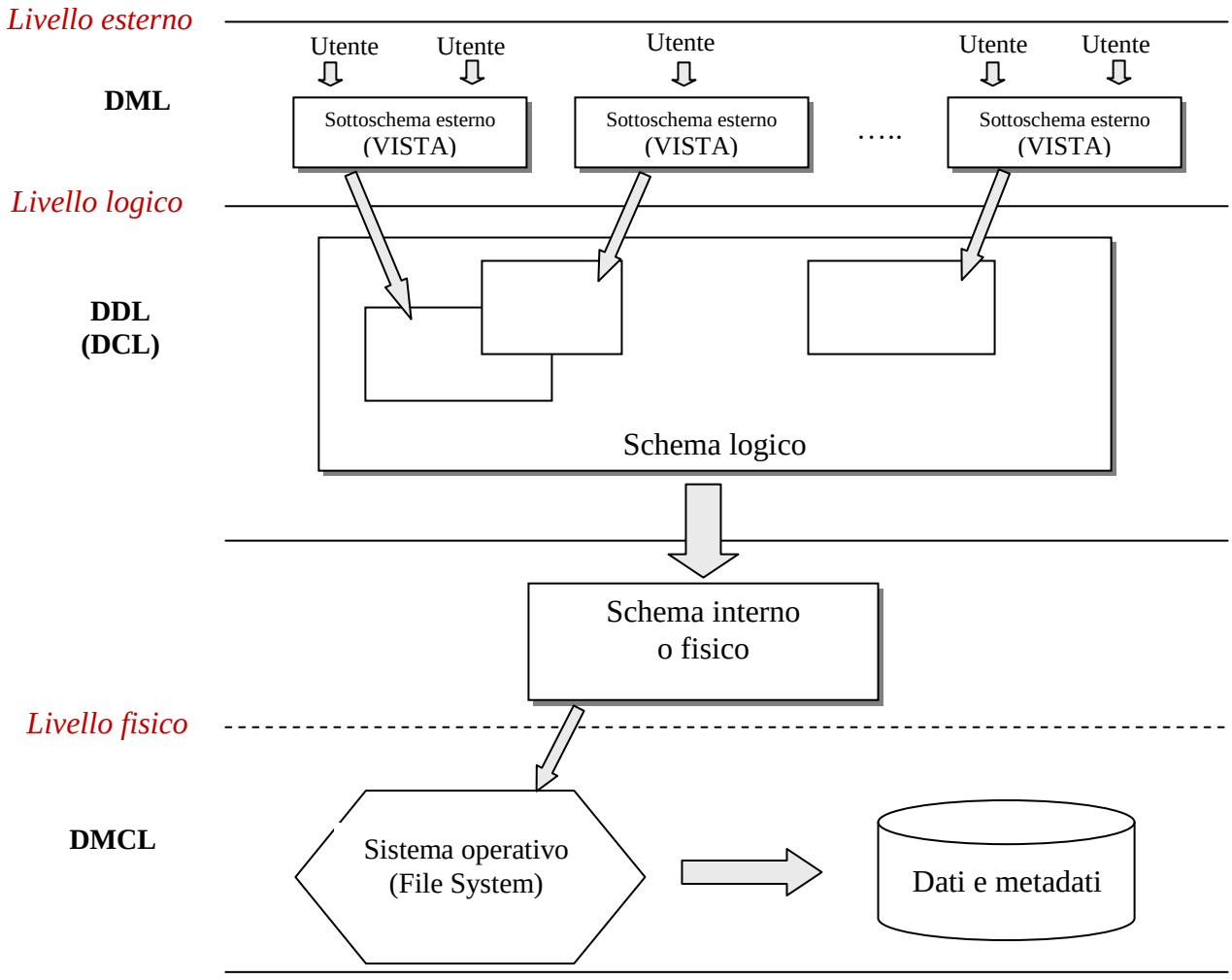
Schematizzando:



ARCHITETTURA DI UN DBMS

Gli utenti di un **DBMS** lo vedono come una *macchina astratta* che permette loro di compiere specifiche operazioni per descrivere e/o modificare i dati contenuti nel data base utilizzando appositi linguaggi.

Lo schema riepilogativo di questa **macchina DBMS** è il seguente:



Il **livello esterno** è il livello con il quale interagiscono i singoli utenti della base di dati attraverso specifiche applicazioni che fornisce gli strumenti utili per modificre e vedere i dati. Ogni utente può essere interessato a vedere solo una parte della base di dati ed è quindi compito del DBA fornire aq ciascun utente un sottoschema (o vista) logico del database che contenga tutto e solo ciò che l'utente desidera ed è abilitato ad usare.

Il **livello concettuale o logico** troviamo il database logico costituita dalla rappresentazione astratta della base di dati in quanto indipendente dall'implementazione fisica e può essere vista come la definizione dell'intero schema dei dati.

Il **livello interno o fisico** troviamo il database fisico costituita dall'implementazione dedel databse logico. Considera i tipi dei dati, i formati, le strutture di memorizzazione ed i metodi di accesso. Ossia rappresenta la forma in cui la base di dati viene memorizzata ed utilizzata

Infine un **DBMS** deve permettere, secondo la teoria delle basi di dati:
- l'**indipendenza** dalla *struttura fisica* dei dati;
- l'**indipendenza** dalla *struttura logica* dei dati;

L'**indipendenza fisica** dei dati consiste nella possibilità di modificare lo schema fisico ossia la struttura fisica dei dati senza dover modificare l'organizzazione logica ed i programmi applicativi che usano i dati.
L'**indipendenza logica** dei dati consiste nella possibilità di modificare lo schema concettuale dei dati senza dover modificare i programmi applicativi non interessati alla modifica.