Lezioni ed esercitazioni di didattica di basi di dati e sistemi informativi (INF/01)

Michelangelo Ceci
Dipartimento di Informatica
Università degli Studi di Bari
Tel: +39 - 080 5443262
Fax: +39 - 080 5443269
http://www.di.uniba.it/~ceci/
mailto:ceci@di.uniba.it

Introduzione

Sistemi informativi e informatici

Basi di dati e DBMS

Modelli dei dati

Linguaggi e utenti delle basi di dati

Vantaggi e svantaggi dei DBMS

I Sistemi Organizzativi

Sistema organizzativo: è l'insieme delle *risorse* e delle *regole* che consentono il funzionamento di una qualunque struttura sociale per il raggiungimento dei suoi obiettivi.

Ogni sistema organizzativo (o semplicemente organizzazione) è dotata di un ***sistema informativo***, che organizza e gestisce i *dati* e le *informazioni* necessarie per perseguire gli scopi dell'organizzazione stessa.

Esempi di sistemi informativi

- Uno *studio medico* mantiene, per ragioni fiscali, dati sui pazienti, sulle visite fatte e sulle parcelle richieste.
- Una *biblioteca* gestisce dati sui materiali raccolti, sui prestiti, sulle persone che prendono in prestito materiali, per svolgere varie attività che si possono raggruppare in tre categorie:
  (a) attività rivolte alla raccolta dei documenti (e.g., abbonamento ai periodici);
  (b) attività rivolte alla conservazione e consultazione dei documenti (e.g., produzione cataloghi)
  (c) attività rivolte alla gestione della biblioteca (e.g., controllo della restituzione dei prestiti e dell'arrivo dei periodici)

Esempi di sistemi informativi

3. Un'*industria manifatturiera* gestisce dati e informazioni per svolgere attività che includono:
   (a) gestione degli ordini e dei pagamenti dei prodotti venduti;
   (b) gestione degli ordini e dei pagamenti ai fornitori di materiali;
   (c) gestione del magazzino;
   (d) pianificazione della produzione;
   (e) controllo di gestione.

Esempi di sistemi informativi

4. Un *comune* gestisce dati e informazioni per svolgere le seguenti attività:
   (a) gestione dei servizi demografici (anagrafe, stato civile, servizio elettorale e vaccinazioni) e della rete viaria;
   (b) gestione dell'attività finanziaria secondo la normativa vigente;
   (c) gestione del personale per il calcolo della retribuzione.

Sistemi Informativi e Sistemi Informatici

Parte di un sistema informativo può essere gestito per mezzo di strumenti automatici, e parte attraverso archivi cartacei e procedure manuali.

Viene definito ***sistema informatico*** la parte automatizzata del sistema informativo. Esso è costituito dall'insieme degli strumenti informatici impiegati per il trattamento delle informazioni di un'organizzazione al fine di agevolare le funzioni del suo sistema informativo.

Il sistema informatico include anche le componenti di confine (*interfacce*) fra la parte automatizzata e la parte manuale.

Sistemi Informativi e Sistemi Informatici

Esempio:

Il sistema informatico di una banca è costituito da:

- Archivi elettronici in cui sono memorizzati i dati dei clienti, dei conti correnti, dei prestiti, ecc.
- I sistemi di calcolo (hardware e software) che consentono la gestione di tali archivi.
- La rete di terminali utilizzati dagli operatori agli sportelli delle agenzie.

Sistemi Informativi e Sistemi Informatici

La parte non automatizzata è invece costituita da:

- operatori,
- eventuali archivi e tabelle non ancora automatizzati (e.g., elenco dei numeri degli assegni rubati),
- da documenti legali (e.g., atti notarili, certificati penali).

Sistemi Informatici

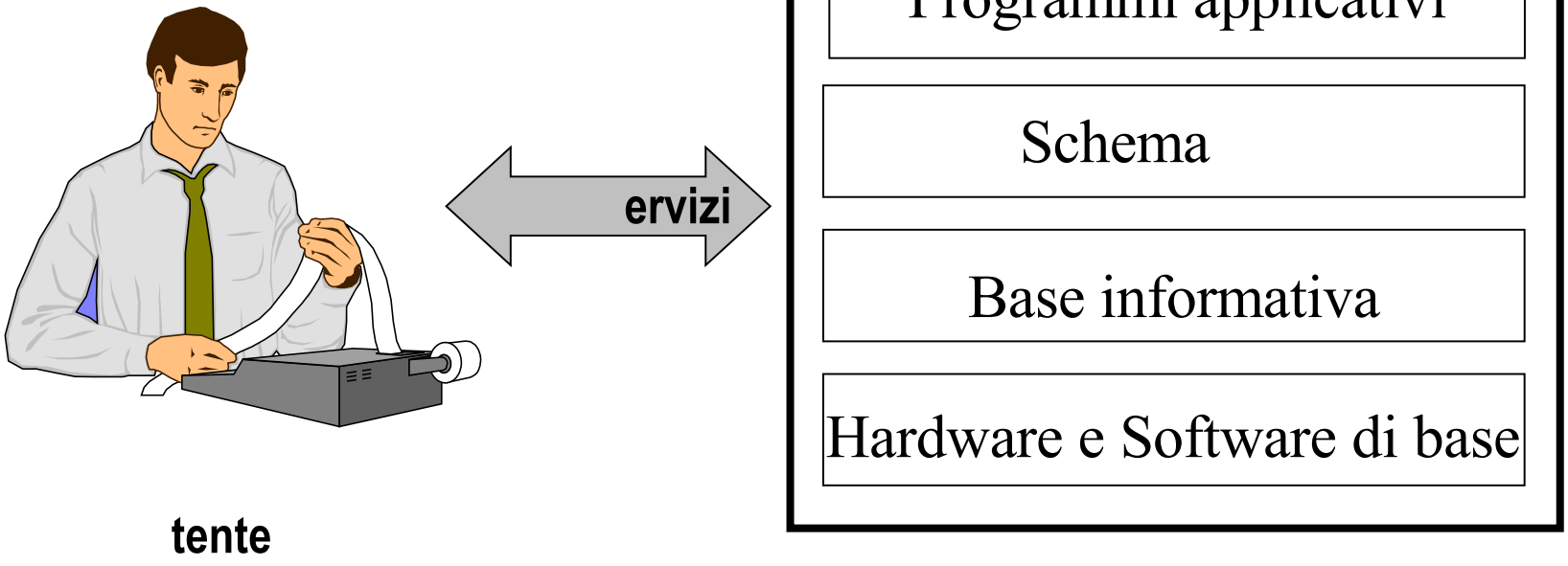
Un sistema informatico è ***un sistema software orientato alla gestione dei dati***, in cui l'aspetto prevalente è rappresentato dai dati che vengono memorizzati, ricercati, e modificati, e che costituiscono il patrimonio informativo di un'organizzazione.

In questo si distingue dal

- Software ***orientato alla realizzazione di funzioni***, in cui la complessità prevalente riguarda le funzioni da realizzare,
- Software ***orientato al controllo***, in cui la complessità prevalente del sistema riguarda il controllo delle attività che si sincronizzano e cooperano durante l'evoluzione del sistema.

Componenti del Sistema Informatico

Per fornire i servizi attesi dagli utenti il sistema informatico prevede quattro componenti:



Componenti del Sistema Informatico

- il ***software*** e ***l'hardware di base***;
- una ***base informativa*** (o base di dati), che contiene una rappresentazione delle informazioni memorizzate;
- uno ***schema***, che descrive la struttura della base informativa, le operazioni per agire su di essa e le restrizioni sui valori memorizzabili nella base informativa e sui modi in cui essi possono evolvere nel tempo (*vincoli d'integrità*).
- I ***programmi applicativi***, che forniscono i servizi agli utenti eseguendo un certo insieme di operazioni sulla base informativa.

Componenti del Sistema Informatico

Nel seguito saranno considerate solo informazioni ***strutturate***, con un formato predeterminato, trascurando i problemi legati al trattamento di informazioni ***non strutturate***, tipo testi e immagini, gestite mediante *sistemi per il recupero dell'informazione* (***information retrieval***).

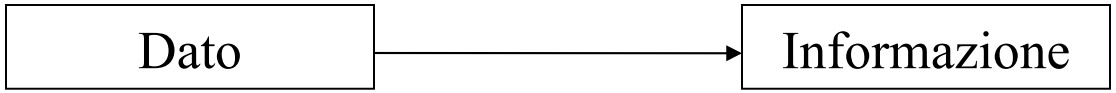
Dati e Informazioni

Prima di procedere con lo studio dei sistemi informatici e delle basi di dati è opportuno chiarire alcuni concetti chiave utilizzati nella definizione di sistema informativo.

dato: ciò che è immediatamente presente alla conoscenza, prima di ogni elaborazione;

informazione: il risultato di un processo di elaborazione e interpretazione di dati, che porta materia precedentemente sconosciuta al ricevente.

L'informazione deriva da un processo di interpretazione e correlazione dei dati.



In altri termini, tutto ciò che produce variazione nel patrimonio conoscitivo di un soggetto è informazione.

Dati e Informazioni

Esempio: per un operatore di borsa, sono informazioni gli articoli letti su un quotidiano.

I dati sono fatti noti che possono essere registrati su un qualche supporto in una forma simbolica.

Esempio: le parole scritte negli articoli di un quotidiano.

I dati da soli non hanno alcun significato, ma una volta interpretati e correlati opportunamente, essi forniscono informazioni, che consentono di arricchire la nostra conoscenza del mondo.

Il concetto di informazione fa riferimento al suo percettore, al suo utilizzatore, in altre parole al contesto che le conferisce significato.

Dati e Informazioni

Esistono dati che non possono produrre informazioni (e.g, i tabulati inutilizzati dagli impiegati) e informazioni che difficilmente possono essere trasferite in un dato (e.g., l'espressione del medico che esamina i risultati di un esame clinico).

Evoluzione dei sistemi informatici

Un sistema organizzativo è generalmente composto da più sottosistemi tra loro coordinati tramite la struttura organizzativa.

Esempio: I sottosistemi nei quali una azienda manifatturiera si articola sono detti ***settori***.

Produzione Vendite Acquisti Personale Magazzino

Evoluzione dei sistemi informatici

In ognuno di questi settori viene generato un insieme di informazioni specifiche di quel settore.

La generazione di informazioni avviene in seguito al verificarsi di eventi.

Esempio:

Settore	*Evento*	*Dati generati*
Vendite	Ricezione ordine	Codice ordine Codice cliente Codice articolo Quantità Data ordine
Produzione	Preparazione del piano	Codice piano di produzione Budget del prodotto
Amministrazione	Fatturazione	Codice fattura Data fattura Data pagamento Importo

Sistemi Informatici Settoriali

Per queste ragioni si stabiliscono tra i settori flussi di documenti che permettono ad ogni settore di procurarsi i dati di suo interesse dal settore originante.

Conseguenze:

1. *Proliferazione dell'informazione*, che nel caso di informazioni soggette a frequenti aggiornamenti obbliga a definire regole organizzative per garantire la *consistenza delle informazioni*.

Sistemi Informatici Settoriali

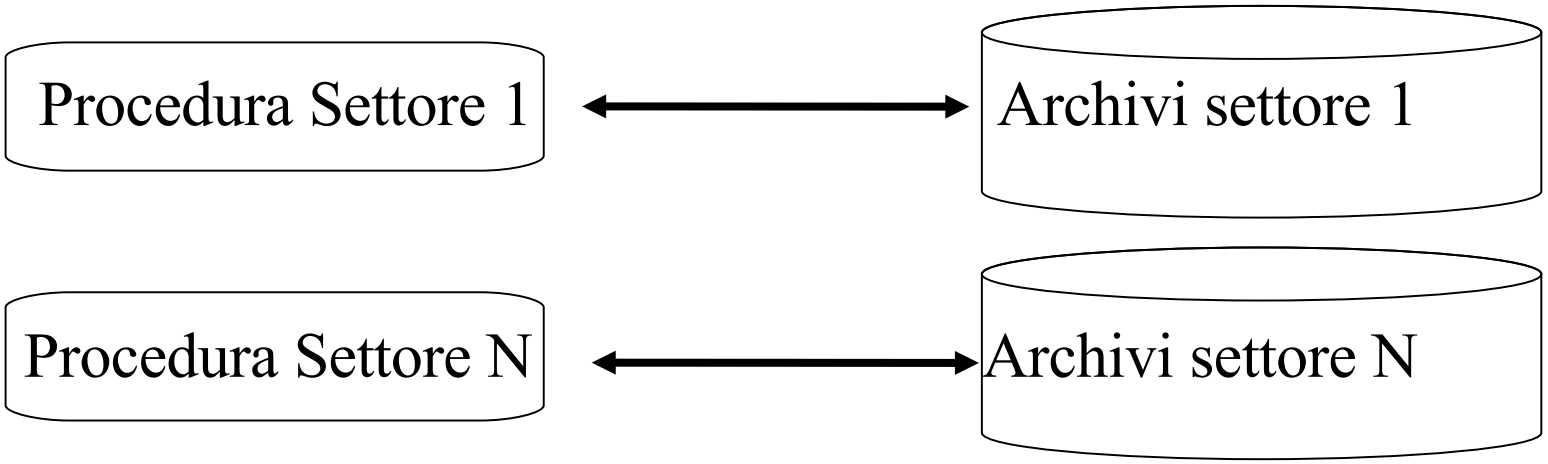
2. *Relazione gerarchica* che si instaura fra i settori aziendali coinvolti nello scambio informativo: il settore che cede l'informazione, può in qualche modo controllare quantità e qualità dell'informazione fornita al settore ricevente.

Tutto ciò indipendentemente dall'automazione.

Nella fase iniziale del processo di automazione dei vari settori aziendali, le procedure tendono ad essere prodotte separatamente per ogni settore.

Sistemi Informatici Settoriali

Questa situazione ha prodotto e produce varie conseguenze sulla gestione degli archivi dei dati, sugli scambi informativi tra i settori aziendali e sull'efficienza stessa del sistema automatizzato di produzione delle informazioni.



Sistemi Informatici Settoriali

Vantaggi:

Riduzione dei costi.

Maggiore produttività settoriale.

Svantaggi:

Alta ridondanza dei dati. Tutti i dati comuni ai diversi archivi devono essere duplicati.

Esistenza di diversi cicli di vita per lo stesso dato. Gli eventi di nascita e morte di un dato variano da settore a settore.

Sistemi Informatici Settoriali

Diversità delle caratteristiche fisiche. Ogni settore sceglie il supporto di memorizzazione, la struttura di memorizzazione, la chiave di accesso, ecc. secondo criteri di ottimizzazione locali.

Varietà dei vincoli di integrità. Le proprietà che il dato deve rispettare possono variare da settore a settore.

Sistemi Informatici Settoriali

Le conseguenze negative provocate dall'architettura non integrata dei sistemi informatici settoriali sono spesso imputate all'uso di ***tecnologie*** arretrate di trattamento e di elaborazione dei dati, e in particolare all'affidamento della gestione dei dati ai linguaggi di programmazione *tradizionali* (Cobol, Fortran, C, ecc.), i quali usano i ***file*** per memorizzare i dati in modo permanente.

Approccio convenzionale: uso dei file

Gli svantaggi più evidenti dell'uso dei file sono:

- I dati sono organizzati in insiemi *indipendenti* (le relazioni fra i dati non sono rappresentate).
- Gli operatori disponibili sui file dipendono dal *tipo* di archivio (sequenziale, relativo, associativo)

3 Per effettuare semplici operazioni sui dati è necessario scrivere programmi.

4 La struttura logica di un archivio deve essere dichiarata in tutti i programmi che ne fanno uso.

Approccio convenzionale: uso dei file

1. Le strutture ausiliarie per agevolare il recupero dei dati dalla memoria secondaria vanno realizzate esplicitamente quando queste sono diverse da quelle del tipo di archivio disponibile.
2. Una modifica della struttura dei dati, del tipo di archivio o dei meccanismi di accesso ai dati comporta una modifica dei programmi che fanno uso dell'archivio.
3. I meccanismi di protezione dei dati da mal-funzionamenti del sistema di elaborazione, o dei programmi che li usano, vanno realizzati esplicitamente.

Queste limitazioni sono intollerabili per sistemi informatici complessi.

Requisiti di un sistema informatico complesso

- *Integrazione dei dati*. Si deve disporre di un'unica raccolta di dati comuni, che costituiscono le informazioni di base, e tanti programmi che realizzano le applicazioni operando solo sui dati di loro interesse. In questo modo si limitano le ridondanze.

  Esempi:

  a) per le USL della Lombardia i propri assistiti sono 10.804.412, mentre la popolazione, alla stessa data, è di 8.884.930 (dicembre 1985).

  b) istituendo l'anagrafe sanitaria al Comune di Bologna si è scoperto che c'erano 7000 doppioni, ovvero cittadini assistiti due volte (agosto, 1990).

Requisiti di un sistema informatico complesso

2. *Flessibilità della realizzazione*. Un sistema informatico di supporto ai sistemi informativi deve essere progettato in modo incrementale: si parte automatizzando un nucleo di funzioni essenziali e, successivamente, si amplia il sistema. Inoltre con l'uso del sistema, si possono scoprire nuove possibilità d'impiego.

Requisiti di un sistema informatico complesso

I *sistemi di gestione di basi di dati* (***DBMS***, *Data Base Management Systems*) o, più semplicemente, i *sistemi per basi di dati*, vengono considerati strumenti avanzati di archiviazione e reperimento di informazioni in grado di soddisfare i requisiti di un sistema informatico complesso.

I ***DBMS*** sono sistemi software in grado di gestire collezioni di dati che siano *grandi*, *condivise* e *persistenti*, assicurando la loro *affidabilità* e *privatezza*.

Base di Dati

Il termine ***base di dati*** viene inteso nel suo significato concettuale di insieme integrato dei dati di un sistema organizzativo, ovvero come collezione di dati utilizzati per rappresentare le informazioni di interesse per un sistema informativo.

I dati di una base di dati sono gestiti da un elaboratore elettronico, e sono suddivisi in due categorie:

Base di Dati

- I ***metadati***, ovvero lo *schema* della base dei dati (*database schema*), una raccolta di definizioni che descrivono la struttura dei dati, le restrizioni sui valori ammissibili dei dati (vincoli d'integrità), le relazioni esistenti tra gli insiemi, e a volte anche alcune operazioni eseguibili sui dati.
- I ***dati***, le rappresentazioni di fatti conformi alle definizioni dello schema, con le seguenti caratteristiche:
  - Organizzazione in insiemi omogenei. La struttura dei dati e le relazioni dipendono dal *modello* dei dati utilizzato.
  - b) Grandi quantità, non gestibili in memoria centrale.

Base di Dati

c) Permanenza.

d) Accessibilità mediante *transazioni* (*transactions*), unità di lavoro atomiche che non possono avere effetti parziali.

e) Protezione da accessi non autorizzati e da malfunzionamenti hardware/software.

f) Utilizzabilità contemporaneamente da diversi utenti

Lo schema è la parte *invariante* nel tempo, mentre i dati o ***istanze*** sono la parte variabile nel tempo.

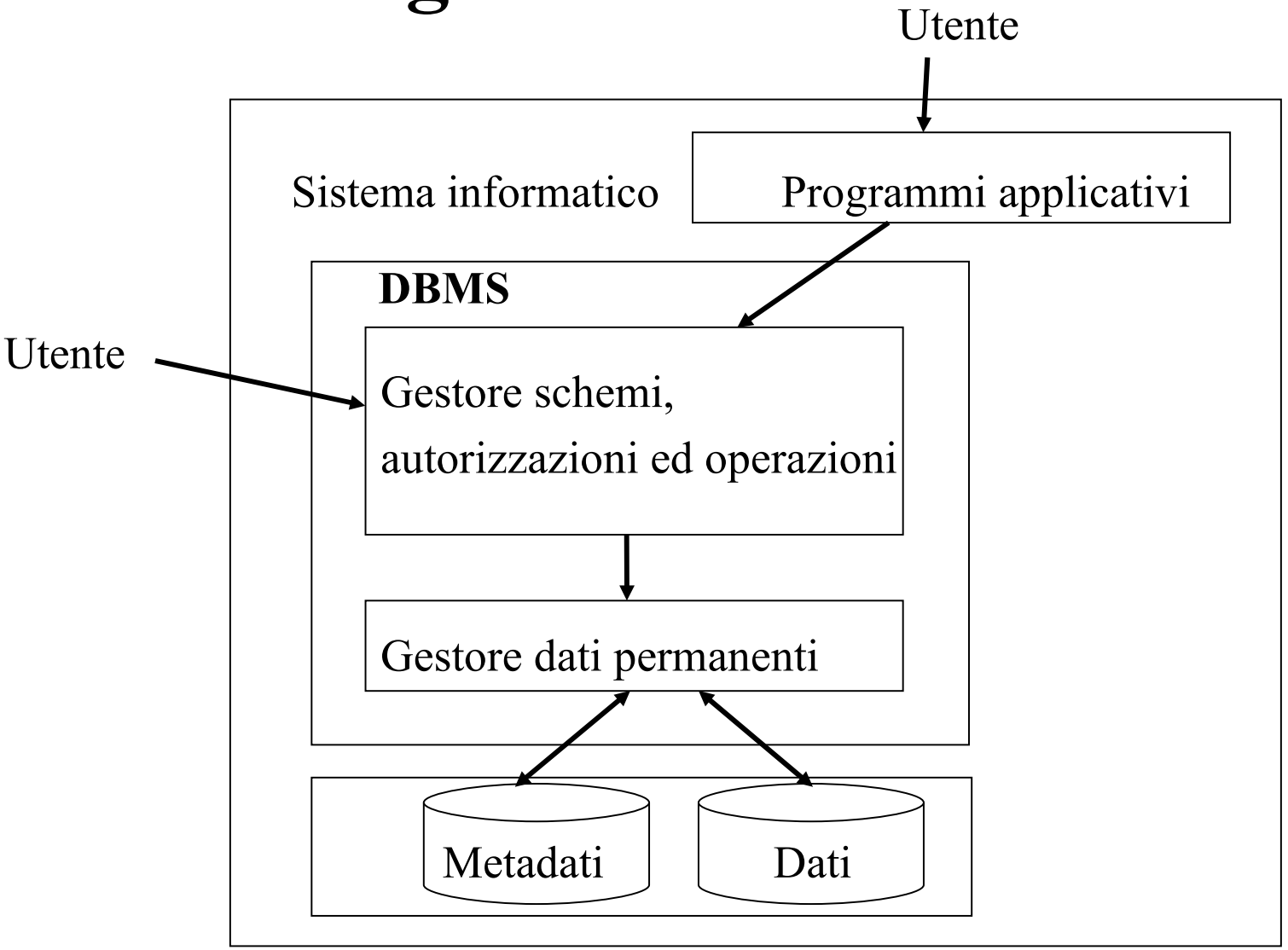
Sistemi di gestione di basi di dati

Un DBMS consente

- Di definire schemi di basi di dati;
- Di scegliere le strutture dati per la memorizzazione e l'accesso ai dati;
- Di memorizzare, recuperare e modificare i dati, interattivamente o da programmi, ad utenti autorizzati e rispettando i vincoli definiti nello schema.

Per estensione, si definisce ***base di dati*** una collezione di dati gestita da un DBMS.

Sistemi di gestione di basi di dati



Modelli dei Dati

Un ***modello dei dati*** è un insieme di concetti utilizzati per organizzare i dati di interesse e descriverne la struttura in modo che essa risulti comprensibile ad un elaboratore.

Ogni modello di dati fornisce meccanismi di astrazione per definire nuovi tipi sulla base di tipi (elementari) predefiniti e costruttori di tipo.

Modelli dei Dati

Un buon modello di dati è caratterizzato da:

Espressività: rappresentazione in modo naturale e diretto del significato di ciò che si sta modellando.

Semplicità d'uso: pochi meccanismi, semplici da usare e da comprendere.

Efficienza di realizzabilità.

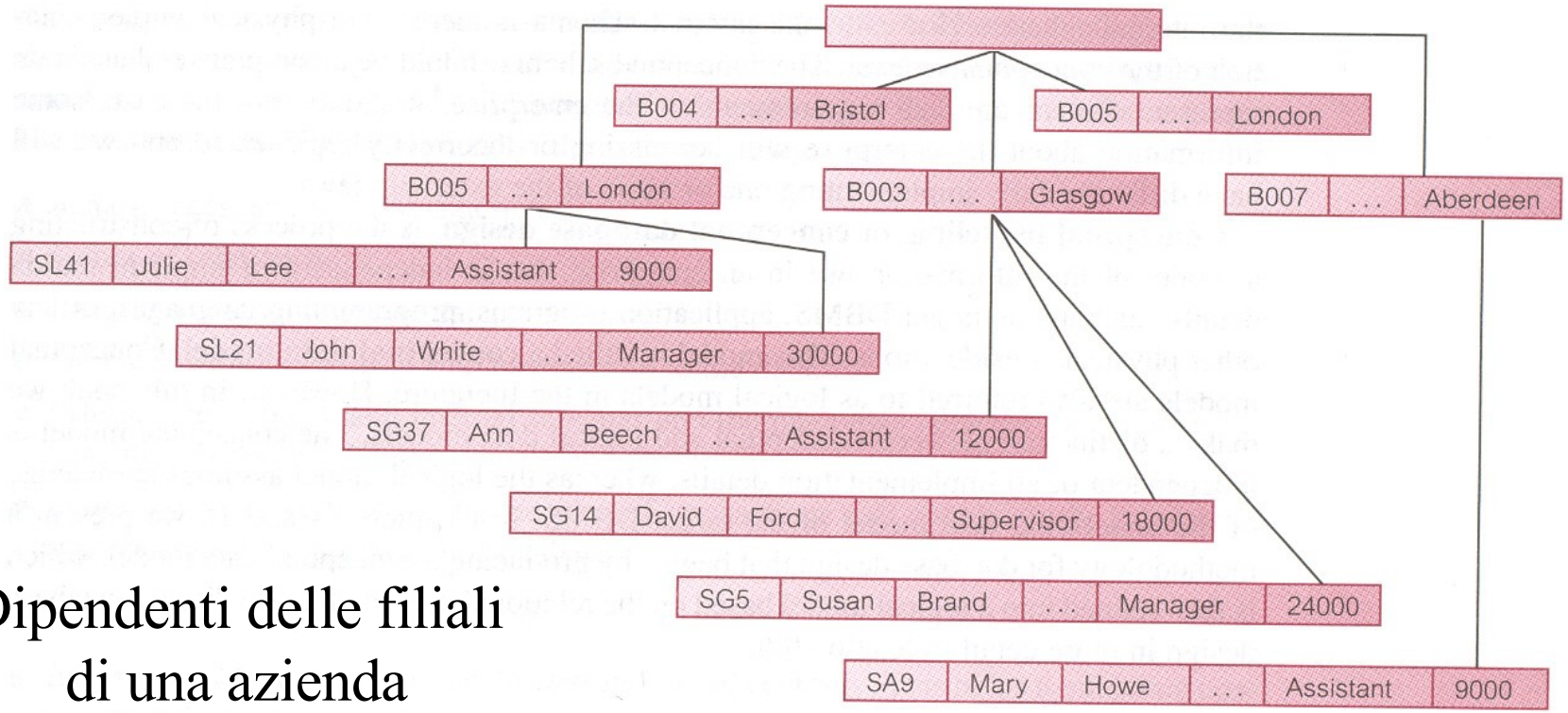
I modelli dei dati adottati dai sistemi commerciali sono distinguibili in due grandi categorie:

- Modelli dei dati basati su record
- Modelli dei dati basati su oggetti

Modelli dei Dati

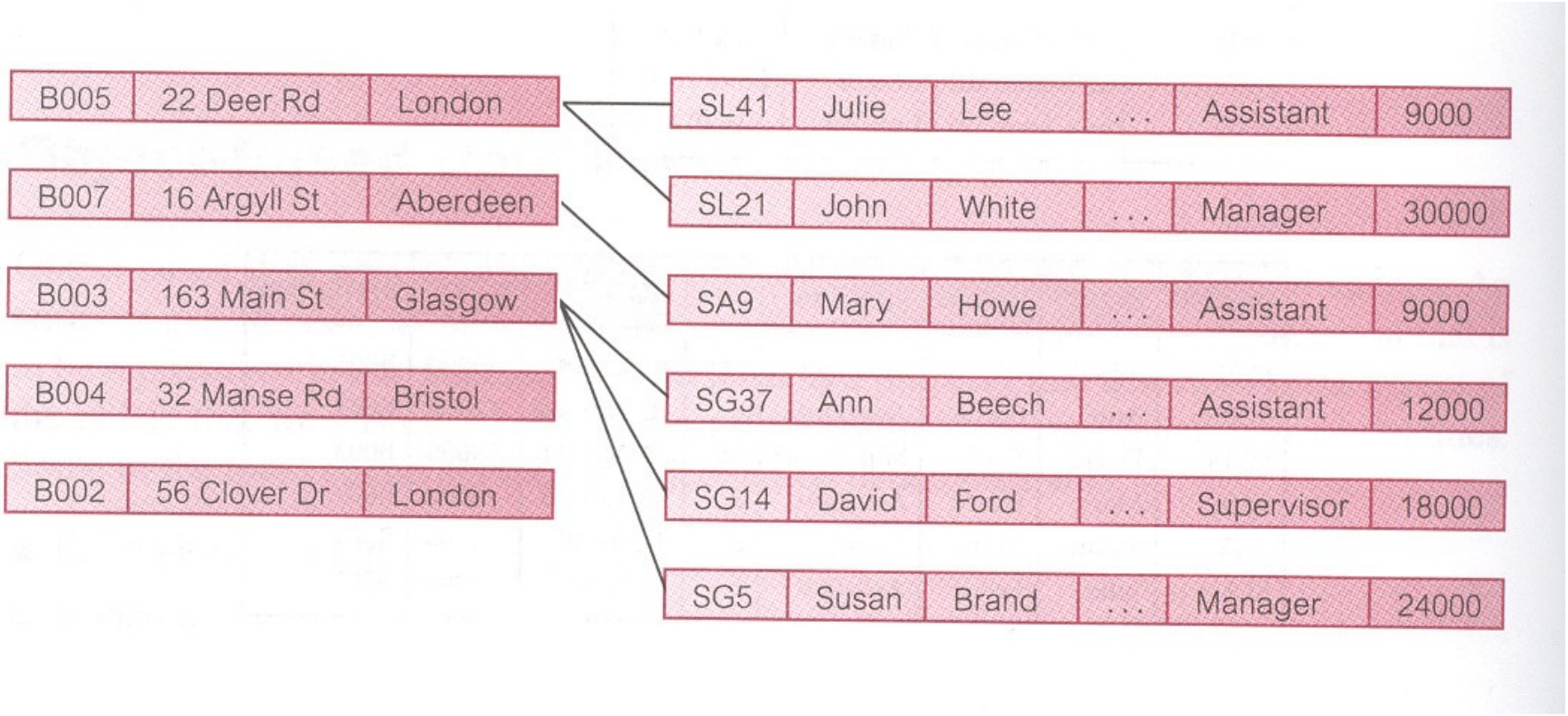
Basati su record	***modello gerarchico***	basato sull'uso di *strutture ad albero*	'60	IMS System 2000
	modello reticolare	basato sull'uso di *grafi*	'70 Proposta CODASIL	IDMS IDS II DM IV
	modello relazionale	basato sull'uso di *relazioni*, insiemi di record a struttura fissa con campi di tipo primitivo	'80	System R Ingres Oracle DB2
Basati su oggetti	***modello relazionale a oggetti***	basato sull'uso di *relazioni*, insiemi di oggetti con identità	'90	Postgres Illustra
	modello a oggetti	basato sull'uso di *classi di oggetti* e *istanze*	'90	O_2 ObjectStore

Il modello gerarchico



Dipendenti delle filiali di una azienda

Il modello reticolare



Il modello relazionale

Branch

branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

È il modello che vedremo nel corso

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Modelli dei Dati vs. Modelli Concettuali

I modelli dei dati precedentemente elencati sono detti

- **Logici**, per evidenziare il fatto che le strutture utilizzate da questi modelli, pur essendo astratte, riflettono una particolare organizzazione (ad alberi, a grafi, a tabelle o a oggetti).

Modelli dei Dati vs. Modelli Concettuali

Essi si distinguono dai modelli

- **Concettuali**, utilizzati per descrivere i dati in maniera completamente indipendente dalla scelta del modello logico. Essi tendono a descrivere i *concetti* del mondo reale, piuttosto che i dati utili a rappresentarli. Questi modelli vengono utilizzati nella fase preliminare del processo di progettazione di una base di dati, in modo da modellare la realtà senza legarsi a questioni inerenti la realizzazione. Un modello concettuale è il modello **Entità-Relazioni**.

I modelli concettuali, a differenza di quelli logici, non sono disponibili su DBMS commerciali.

Livelli di astrazione nei DBMS

Nei DBMS la base di dati è articolata in tre livelli distinti di descrizione dei dati, per ciascuno dei quali esiste uno schema logico.

- ***Livello fisico***: a questo livello si descrive il modo in cui vanno organizzati fisicamente i dati nelle memorie permanenti e quali strutture dati ausiliarie definire per facilitarne l'uso. La descrizione di questi aspetti viene chiamata ***schema fisico*** o ***interno*** (*physical* o *internal schema*).

Livelli di astrazione nei DBMS

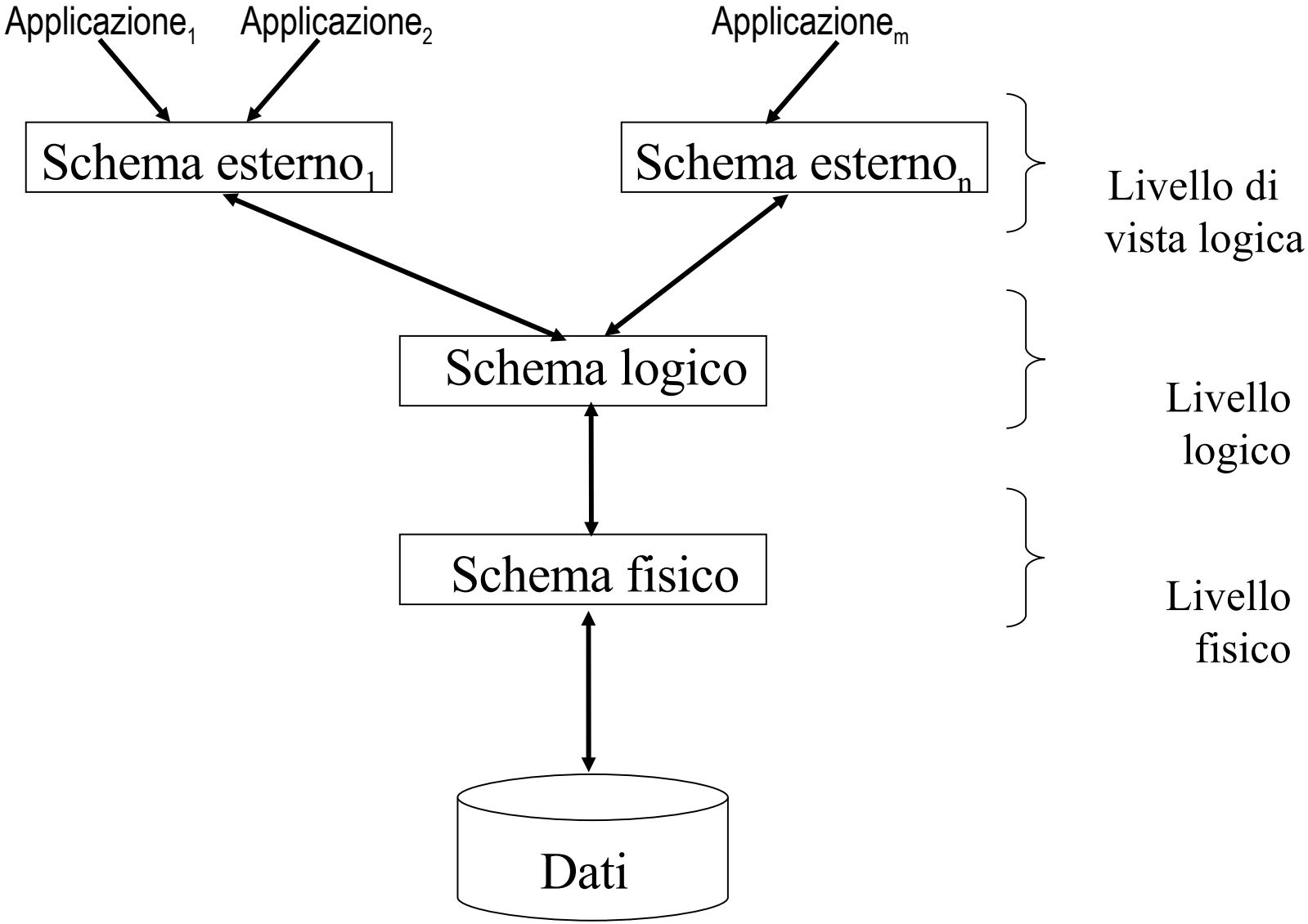
- ***Livello logico***: a questo livello viene descritta la struttura degli insiemi di dati e delle relazioni fra loro, secondo un certo modello dei dati, senza nessun riferimento alla loro organizzazione fisica nella memoria permanente. La descrizione della struttura della base di dati viene chiamata ***schema logico*** (*logical schema*).

Livelli di astrazione nei DBMS

- ***Livello di vista logica***: a questo livello si definisce come deve apparire la struttura della base di dati ad una certa applicazione (o a un certo utente). Questa descrizione viene spesso chiamata ***schema esterno*** o ***vista*** (*external schema* o *user view*), per evidenziare il fatto che essa si riferisce a ciò che un utente immagina che sia la base di dati.

Mentre lo schema logico è unico, esistono in genere più schemi esterni, uno per ogni applicazione (o gruppo di applicazioni correlate).

Livelli di astrazione nei DBMS



Livelli di astrazione nei DBMS

Questa architettura garantisce l'***indipendenza dei dati***.

- ***Indipendenza fisica***: è possibile modificare le strutture fisiche (e.g., le modalità di organizzazione dei file gestiti dal DBMS o la allocazione fisica dei file), senza influire sulle descrizioni dei dati ad alto livello e quindi sui programmi che utilizzano i dati stessi.
- ***Indipendenza logica***: consente di interagire con il livello esterno della base di dati in modo indipendente dal livello logico (e.g., aggiungere un nuovo schema esterno senza modificare lo schema logico).

Livelli di astrazione nei DBMS

Esempio.

Si consideri una base di dati per gestire informazioni sui docenti di un'università, di supporto alle attività dell'ufficio stipendi e della biblioteca.

Livello di vista logica:

Ufficio stipendi: nome e cognome, codice fiscale, parametro, stipendio.

Biblioteca: nome e cognome, recapito telefonico.

Livelli di astrazione nei DBMS

Livello logico

I dati sui docenti sono descritti da un unico insieme di registrazioni (record). Lo schema logico di una base di dati relazionale è dichiarato come segue:

```
CREATE TABLE Personale
(    Nome          CHAR(30),
     CodiceFiscale CHAR(15),
     Stipendio     INTEGER,
     Parametro     CHAR(6),
     Recapito      CHAR(8) )
```

Livelli di astrazione nei DBMS

Per creare le due viste logiche si scriverà:

```
CREATE VIEW PersonalePerUfficioStipendi AS
    SELECT   Nome, CodiceFiscale, Stipendio, Parametro
    FROM Personale
CREATE VIEW PersonalePerLaBiblioteca AS
    SELECT    Nome, Recapito
    FROM Personale
```

Livelli di astrazione nei DBMS

Livello fisico

Il progettista del DB potrà decidere di memorizzare l'insieme in modo sequenziale, oppure con una tecnica hash usando come chiave il nome e cognome:

MODIFY Personale TO HASH ON Nome

Livelli di astrazione nei DBMS

Se in seguito si dovesse scoprire che l'ufficio stipendi esegue con frequenza l'operazione di recupero dei dati riguardanti i docenti che abbiano un certo parametro, si potrebbe pensare di modificare lo schema fisico aggiungendo un indice sul campo Parametro:

```
CREATE INDEX IndiceParametro ON Personale(Parametro)
```

Garantendo l'indipendenza delle applicazioni dalla organizzazione fisica dei dati, si eviterà di modificare i programmi che fanno uso dei dati relativi al personale.

Livelli di astrazione nei DBMS

In futuro si potrebbe decidere di cambiare l'organizzazione logica dei dati sul personale memorizzandoli su due tabelle, TecniciAmministrativi e Docenti. Per rendere le applicazioni che usano la tabella Personale indipendenti da questa modifica, la base di dati si ridefinisce come segue:

Livelli di astrazione nei DBMS

```
CREATE TABLE Docenti
(       Nome            CHAR(30),
        CodiceFiscale   CHAR(15),
        Stipendio       INTEGER,
        Parametro       CHAR(6),
        Recapito        CHAR(8) )
```

```
CREATE TABLE TecniciAmministr.
(       Nome            CHAR(30),
        CodiceFiscale   CHAR(15),
        Stipendio       INTEGER,
        Parametro       CHAR(6),
        Recapito        CHAR(8))
```

```
CREATE VIEW Personale AS
SELECT FROM Docenti UNION SELECT FROM TecniciAmministrativi
```

Linguaggi per Database

Negli usuali linguaggi di programmazione, le istruzioni dichiarative e quelle eseguibili fanno parte dello stesso linguaggio.

Nel mondo dei database, è invece normale separare in due diversi linguaggi le funzioni *dichiarative* e quelle di *elaborazione*.

Il motivo è che, mentre in un normale programma i dati esistono solo mentre esso è in esecuzione, in un sistema di database i dati sono permanenti e possono essere dichiarati una volta per tutte. Può essere allora significativo darne una definizione separata.

Linguaggi per Database

Occorre quindi distinguere:

- ***Linguaggi di definizione dei dati*** o *data definition language* (*DDL*), utilizzati per definire gli schemi logici, esterni e fisici e le autorizzazioni per l'accesso.
- ***Linguaggi di manipolazione dei dati*** o *data manipulation language* (*DML*), utilizzati per l'interrogazione e l'aggiornamento delle istanze di basi dei dati.

Linguaggi per Database

Esempio:

SQL è un *linguaggio di definizione dei dati* per DB con modello relazionale.

```
CREATE TABLE VOLI(NUMERO:INT, DATA:CHAR(6),
    POSTI:INT, DA:CHAR(3), A:CHAR(3));
CREATE INDEX FOR VOLI ON NUMERO;
```

Le prime due righe descrivono la relazione, la terza precisa che deve essere creato un indice sui numeri di volo come parte dello schema fisico.

Linguaggi per Database

Esempio:

SQL è anche un *linguaggio di manipolazione dei dati* per DB con modello relazionale.

```
UPDATE VOLI
SET POSTI = POSTI - 4
WHERE NUMERO=123 AND DATA='AGO 31';
```

In questo modo si decrementa il numero di posti disponibili sul volo 123 del 31 agosto.

Linguaggi per Database

Il termine *query language* viene spesso usato come sinonimo di DML. In senso stretto, solo poche istruzioni di un DML sono interrogazioni, ovvero quelle che non modificano il database.

Originariamente la distinzione fra DDL, DML e query language era netta. Solo recentemente sono stati proposti dei linguaggi che integrano alcune delle funzionalità suddette, come SQL.

Linguaggi per Database

DDL e DML sono linguaggi specializzati per un DB. Per essi è disponibile un'interprete e una interfaccia che ne consente l'uso interattivo.

Tuttavia accade spesso che la manipolazione di un database si ottiene tramite un *programma applicativo*, che fa qualcosa di più che manipolare il DB.

Esempio

Un programma per la prenotazione dei posti deve interrogare e aggiornare il DB, come pure stampare i biglietti, controllare la disponibilità di posti, ecc.

Linguaggi per Database

Perciò di solito i programmi per manipolare il database sono scritti in un *linguaggio ospite (host language)*, che è un linguaggio di programmazione tradizionale, come C, C++, COBOL e FORTRAN.

I comandi del linguaggio per la manipolazione dei dati vengono chiamati dai programmi scritti in linguaggio host in uno dei due modi possibili:

- I comandi del linguaggio di manipolazione dei dati vengono chiamati tramite "call" del linguaggio ospite a *procedure* fornite dal DBMS.

Linguaggi per Database

- Il linguaggio ospite è esteso opportunamente in modo da "ospitare" comandi scritti nel linguaggio per basi di dati. Sarà compito di un *preprocessor* o dello stesso compilatore quello di trattare le istruzioni di manipolazione dei dati, traducendole in richiami a procedure fornite dal DBMS.

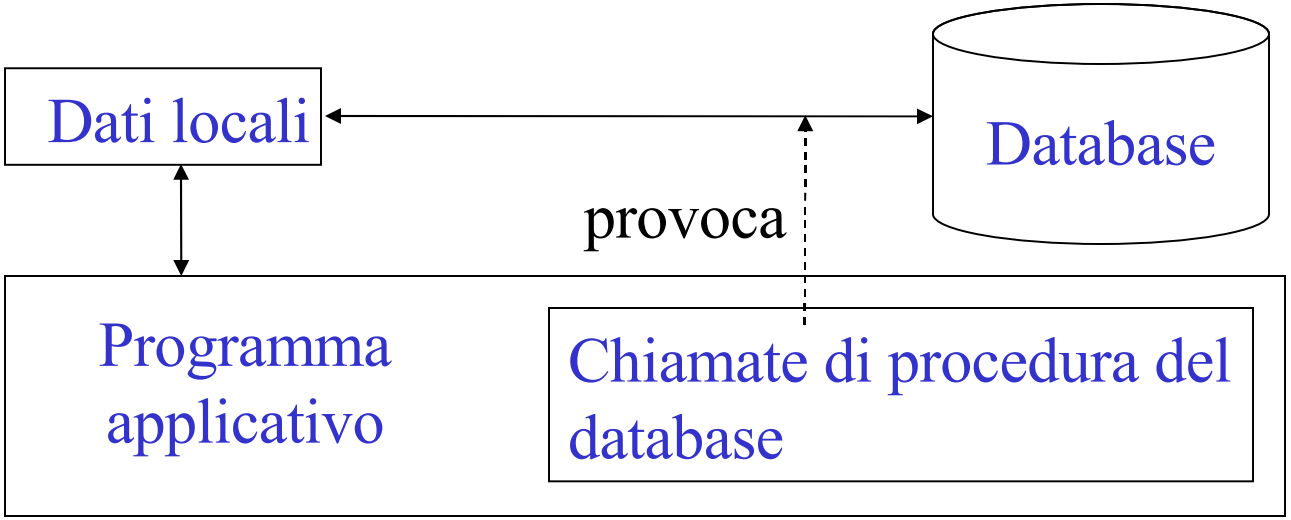
La differenza fra i due approcci non è poi grande.

Esempi:

Richiamo di procedure	*Linguaggio esteso*
CALL GET(B) A := B+1 CALL STORE(A)	##GET(B) A := B+1 ##STORE(A)

Linguaggi per Database

L'interazione fra il programma applicativo e il DB è schematizzato così:



Linguaggi per Database

Una query che richieda dei dati fa sì che la risposta venga trasferita dal database a variabili che sono nell'area dei dati locali; se vi è più di una risposta, allora esse sono recuperate una per volta quando l'applicativo richiama una procedura di *fetch*.

Se si inseriscono o modificano dei dati, i valori vengono trasferiti dalle variabili locali al database, sempre in risposta a chiamate ad opportune procedure.

Linguaggi per Database

Va detto che ci sono due tipi di DML:

- quelli di ***alto livello*** o ***non procedurali*** e
- quelli di ***basso livello*** o ***procedurali***.

Una query in un DML di alto livello, come SQL, consente di specificare quali dati (***what***) cercare, piuttosto che come (***how***) reperirli.

Per questo i DML di alto livello sono anche detti dichiarativi.

Linguaggi per Database

I DML procedurali devono essere inclusi in un linguaggio ospite, e consentono tipicamente il trattamento di un record alla volta.

Per questa ragione sono anche detti ***record-at-a-time*** DML, in modo da distinguerli dai DML di alto livello, denominati ***set-at-a-time*** o ***set-oriented***.

I linguaggi record-at-a-time devono essere incorporati in linguaggi ospite in modo da poter usufruire dei costrutti per i cicli in modo tale da trattare insiemi di dati.

Interfacce per DBMS

Gli utenti occasionali di un DB tipicamente usano un linguaggio ad alto livello per specificare le richieste, mentre i programmatori usano linguaggi ospite.

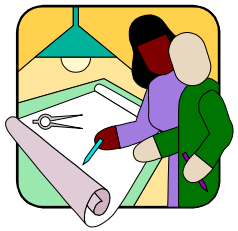
Infine per utenti del tutto inesperti, sono spesso disponibili ***interfacce user-friendly*** per interagire con il DB.

- *Interfacce basate su menu*: la query è composta passo dopo passo mediante scelte di un menu. Un esempio sono i ***browser***, che consentono di navigare nel DB in modo non strutturato.

Utenti della Base di Dati

Amministratore di DBMS (***database administrator***, ***DBA***): è il responsabile dell'autorizzazione all'accesso al DB, del coordinamento e al monitoraggio dell'uso. Acquista le risorse software e hardware ed è diretto responsabile dei problemi di sicurezza e dei tempi di risposta. In grandi organizzazioni è coadiuvato da uno staff.

Utenti della Base di Dati



Progettisti del DB: sono responsabili della identificazione dei dati da memorizzare nel DB e della scelta delle strutture appropriate per rappresentare e memorizzare questi dati. I progettisti di DB tipicamente interagiscono con ciascun gruppo di potenziali utenti e sviluppano una vista del database che rispecchia le richieste informative ed elaborative del gruppo. Quindi le viste sono analizzate ed integrate per definire il livello logico e fisico.

Utenti della Base di Dati

Utenti finali: distinguibili in utenti *occasionali*, utenti *parametrici* (come agenti di viaggio, terminalisti e impiegati di banca), utenti *sofisticati* (ingegneri, scenziati, ecc.), e utenti *stand-alone* che utilizzano pacchetti pronti all'uso mediante interfacce grafiche.

Programmatori di applicazioni e *analisti di sistemi*: definiscono le richieste degli utenti finali, scrivono le specifiche e producono i programmi necessari.

Utenti dietro le quinte

- ***Progettisti di DBMS*** e ***implementatori***: un DBMS è un sistema software complesso che consiste di molti moduli per il linguaggio di interrogazione, per i processori di interfaccia, per l'accesso ai dati, per la memorizzazione e per la sicurezza. Il DBMS deve interfacciarsi con altri sistemi software complessi, come i sistemi operativi e i traduttori di vari linguaggi di programmazione.

Utenti dietro le quinte

2. ***Sviluppatori di tool***: i *tool* sono pacchetti software che facilitano il progetto di un DB e aiutano a migliorarne le prestazioni. Esempi di tool sono quelli che consentono il monitoraggio delle risorse, la prototipazione rapida, la generazione di dati di test, la generazione di report, ecc.

Controllo della base di dati: Integrità

I DBMS prevedono dei meccanismi per controllare che i dati inseriti, o modificati, siano conformi alle definizioni date nello schema, in modo da garantire che la base di dati si trovi sempre in uno stato *consistente*.

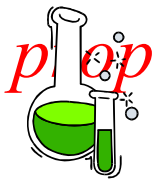
La tendenza è di avere linguaggi per la definizione dello schema logico che consentano di definire anche le condizioni a cui essi devono sottostare per essere significativi (*vincoli d'integrità*), quando queste condizioni vanno verificate e cosa fare in caso di violazioni.

Controllo della base di dati: Affidabilità

I DBMS devono disporre di meccanismi per proteggere i dati da malfunzionamenti hardware o software e da interferenze indesiderate dovute ad accessi concorrenti ai dati da parte di più utenti.

A tal fine un DBMS prevede che le interazioni con la base di dati avvengano per mezzo di *transazioni*.

Una ***transazione*** è una sequenza di azioni di lettura e scrittura della base di dati e di elaborazioni di dati in memoria temporanea, che il DBMS esegue garantendo le seguenti proprietà (***ACID properties***):



Controllo della base di dati: Affidabilità

Atomicity: la transazione è eseguita nella sua interezza (*committed transaction*) oppure non è eseguita affatto. Le transazioni che terminano prematuramente (*aborted transaction*) sono trattate come se non fossero mai iniziate.

Consistency preservation: una esecuzione corretta della transazione deve portare il DB da uno stato consistente all'altro.

Isolation: una transazione non dovrebbe rendere gli aggiornamenti visibili ad altre transazioni finché non termina normalmente.

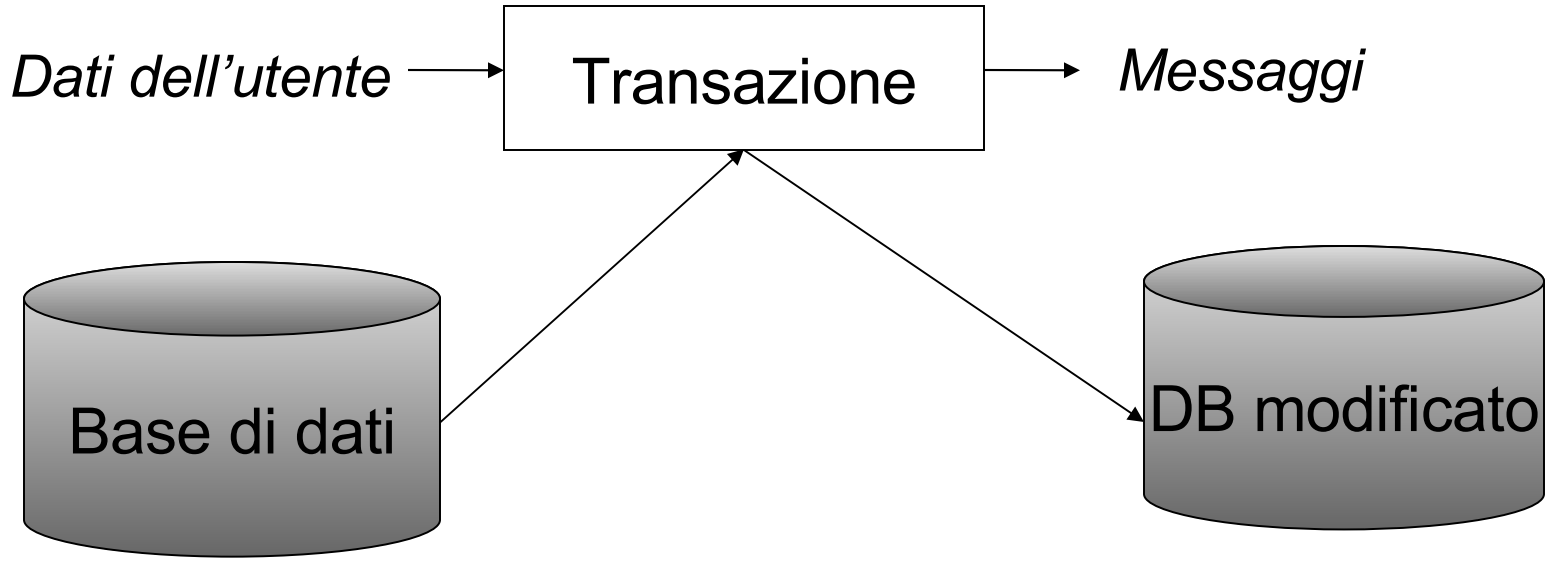
Durability (*persistenza*): le modifiche sulla base di dati di una transazione terminata normalmente sono permanenti, cioè non sono alterabili da eventuali malfunzionamenti successivi alla terminazione.

Controllo della base di dati: Affidabilità

Una transazione può essere espressa come:

- Un insieme di espressioni in un DML.
- Una parte di un programma sequenziale.

In entrambi i casi esiste un qualche meccanismo per segnalare al sistema il punto di inizio ed il punto finale di una transazione.



Controllo della base di dati: Affidabilità

Un ***malfunzionamento*** (*failure*) è un evento che porta il DB in uno stato inconsistente. Se ne distinguono tre tipi:

- ***Fallimenti di transazioni*** (*transaction failure*): sono interruzioni di transazioni che non comportano perdite di dati né in memoria temporanea (buffer) né in memoria permanente. Sono dovuti a condizioni già previste nei programmi. Esempi sono la violazione di vincoli di integrità e il tentato accesso a dati protetti.
- ***Fallimenti di sistema*** (*system failure*): sono interruzioni del funzionamento del DBMS che fanno perdere il contenuto della memoria temporanea ma non della memoria permanente. Sono dovuti a malfunzionamenti hardware/software.
- ***Disastri*** (*disk/media failure*): danneggiano la memoria permanente contenente il DB.

Controllo della base di dati: Affidabilità

L'interruzione istantanea di una transazione o dell'intero sistema causa l'attivazione di opportune procedure che permettano di riportare la base di dati allo stato corretto precedente alla manifestazione del malfunzionamento (procedure di ***ripristino*** o *recovery*).

Per poter eseguire queste procedure un DBMS mantiene una copia di sicurezza della base di dati e tiene traccia di tutte le modifiche fatte sulla base di dati dal momento in cui è stata eseguita l'ultima copia di sicurezza.

Controllo della base di dati: Affidabilità

Grazie a questi dati ausiliari, quando si verifica un malfunzionamento il DBMS può ricostruire una versione corretta dei dati utilizzando l'ultima copia e rieseguendo tutte le operazioni che hanno modificato i dati e di cui ha mantenuto traccia.

Controllo della base di dati: Affidabilità

Per poter garantire l'affidabilità, un DBMS deve anche *controllare la concorrenza*, in modo da evitare la perdita di modifiche.

Esempio: lettura e aggiornamento simultaneo di un saldo.

Un modo semplice di risolvere il problema sarebbe quello di eseguire le transazioni isolatamente, cioè in modo tale che, per ogni coppia di transazioni T_i e T_j, tutte le azioni di T_i precedono quelle di T_j (*esecuzione seriale*).

I DBMS hanno dei meccanismi più sofisticati per gestire la concorrenza.

Controllo della base di dati: Sicurezza

I DBMS prevedono meccanismi simili a quelli presenti nei sistemi operativi per controllare che ai dati accedano solo persone autorizzate.

- Identificazione degli utenti autorizzati.
- Protezione da furti mediante crittografia.

Inoltre consentono di porre delle *restrizioni*.

- Restrizioni sui dati leggibili e modificabili.
- Restrizioni sulle ore di accesso al DB.
- Accesso solo a informazioni aggregate (statistiche), senza la possibilità di prendere visione di dati singoli.

Classificazione dei DBMS

Il criterio principale utilizzato per classificare è il ***modello dei dati*** sul quale su fonda il DBMS, distinguendo le basi di dati gerarchiche da quelle reticolari (*network*), relazionali, orientate a oggetti, ecc.

Altri criteri sono:

- **Il numero di utenti**

  Sistemi ***single-user*** supportano solo un utente per volta e sono per lo più usati con personal computer.

  La maggior parte dei DBMS sono ***multi-user***.

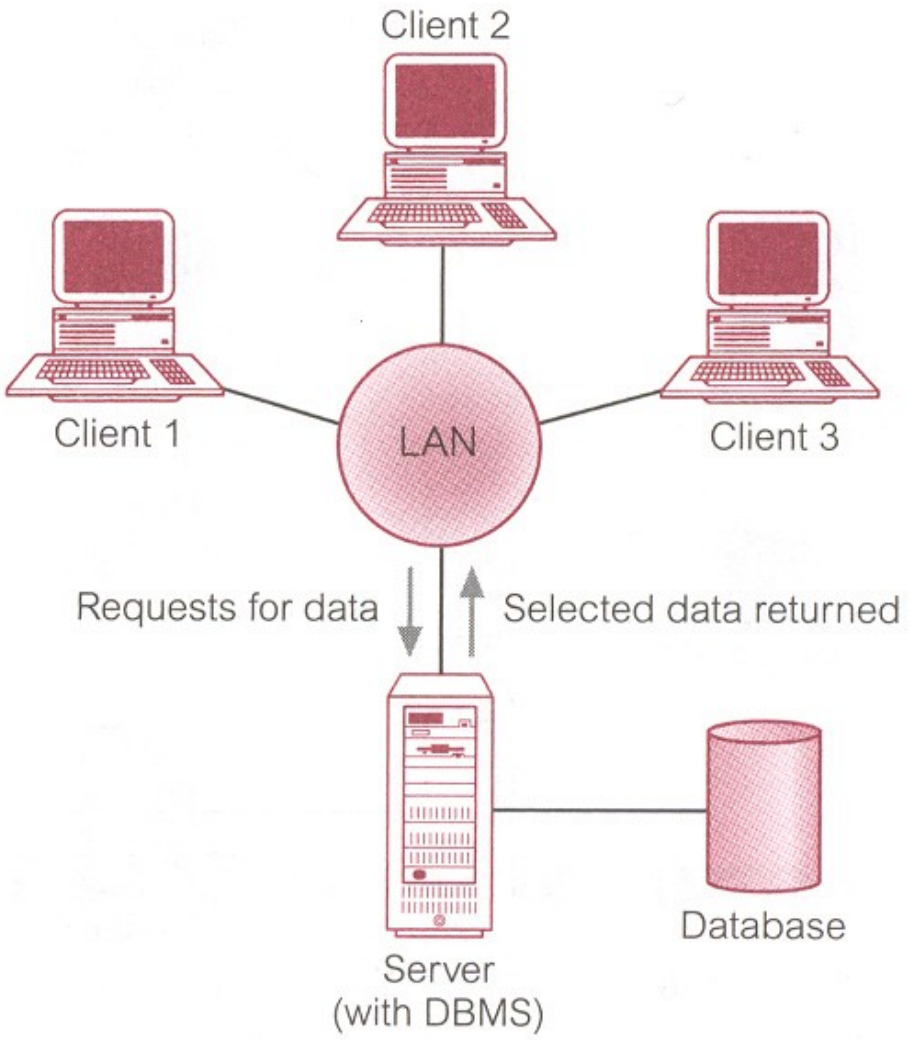
Classificazione dei DBMS

- **Il numero di centri (site) in cui è distribuito il DB.**

  La maggior parte dei DBMS sono ***centralizzati***, intendendo che i dati sono memorizzati in un unico centro. Un DBMS centralizzato può supportare più utenti, eventualmente remoti.

  Un DBMS ***distribuito*** può avere dati e software distribuito in più centri connessi da una rete locale o geografica. Tipicamente i database distribuiti hanno un'architettura *client-server*.

Classificazione dei DBMS



Classificazione dei DBMS

I DBMS distribuiti ***omogenei*** usano lo stesso software.

Una tendenza recente è quella di sviluppare software per accedere a diversi database autonomi preesistenti memorizzati in DBMS ***eterogenei***.

Ciò porta a un DBMS ***federato*** (***multidatabase***) dove i DBMS partecipanti sono accoppiati in modo lasco e hanno un alto grado di autonomia.

- **Costo**

  Single-user, 100$ ÷ 3000$.

  Multi-user, 10.000$ ÷ 100.000$.

Classificazione dei DBMS

- **General-purpose / special-purpose**

  Quando le prestazioni sono importanti, si può progettare e costruire un sistema *special-purpose* per l'applicazione specifica. Molti sistemi di prenotazione delle linee aeree come pure il sistema per il prelievo dal Bancomat usano DBMS special-purpose. Questi ricadono nella categoria dei sistemi ***OLTP*** (*on-line transaction processing*), che supportano un gran numero di transazioni concorrenti senza imporre eccessivi ritardi.

Classificazione dei DBMS

Al contrario, per quelle applicazioni in cui si vogliono elaborare enormi volumi di dati per estrarne informazioni di carattere statistico, come nel caso dell'elaborazione dei dati relativi agli acquisti di una catena di grandi magazzini allo scopo di individuare le tendenze di mercato, si potranno utilizzare altri DBMS special-purpose, i ***datawarehouse*** o i ***data mart***.

Vantaggi e problemi nell'uso dei DBMS

Oltre ai vantaggi di integrazione dei dati e di flessibilità della base di dati, la tecnologia delle basi di dati ne offre anche altri, in particolare

- quello di stabilire degli standard riguardo alla strutturazione e nomenclatura dell'informazione;
- quello di ridurre notevolmente i tempi di sviluppo delle applicazioni rispetto a quelli richiesti usando la tecnologia degli archivi.

Tuttavia si hanno anche

Vantaggi e problemi nell'uso dei DBMS

- **Problemi gestionali ed organizzativi**
  1. I DBMS più sofisticati richiedono un notevole impegno hardware e software per la loro messa a punto ed il loro funzionamento.
  2. Il costo di questi sistemi ed i costi di esercizio possono essere facilmente sottostimati.
  3. È importante acquisire e mantenere personale qualificato e con competenze specifiche sul sistema impiegato.
  4. L'introduzione del sistema ha un impatto sulla struttura organizzativa.

Vantaggi e problemi nell'uso dei DBMS

- **Problemi di produzione del software**
  1. Il progetto di base di dati e la messa a punto delle applicazioni richiedono personale qualificato e strumenti opportuni, che non sono messi a disposizione da tutti i sistemi.
  2. L'impiego di una base di dati richiede una ristrutturazione dei dati in archivi già esistenti e la riscrittura dei programmi applicativi.
  3. L'impiego di un DBMS può aumentare la dipendenza da ditte esterne.

Vantaggi e problemi nell'uso dei DBMS

- **Problemi di funzionamento**
  1. Alcuni sistemi hanno meccanismi inadeguati per il ripristino e la protezione dei dati da malfunzionamenti hardware e software.
  2. Alcuni sistemi hanno meccanismi inadeguati per prevenire accessi non autorizzati ai dati.
  3. Aumento del rischio di interruzione dei servizi, come conseguenza della centralizzazione.
  4. Massima attenzione all'organizzazione dei dati, per non degradare i tempi di risposta.
  5. La riorganizzazione periodica dei dati può richiedere l'interruzione di accesso al sistema.

Vantaggi e problemi nell'uso dei DBMS

- **Problemi di pianificazione**
  - I costi di acquisizione della tecnologia delle basi di dati e di sviluppo delle applicazioni sono tali da ***non*** rendere praticabile la sostituzione di un sistema con un nuovo prodotto non compatibile. Pertanto la scelta di un DBMS va fatta pianificando l'intervento accuratamente.

Esercizi

Quali delle seguenti affermazioni sono vere?

- l'indipendenza dei dati permette di scrivere programmi senza conoscere le strutture fisiche dei dati
- l'indipendenza dei dati permette di modificare le strutture fisiche dei dati senza dover modificare i programmi che accedono alla base di dati
- l'indipendenza dei dati permette di scrivere programmi conoscendo solo lo schema concettuale della BD
- l'indipendenza dei dati permette di formulare interrogazioni senza conoscere le strutture fisiche

Esercizi

Quali delle seguenti affermazioni sono vere?

- il fatto che le basi di dati siano condivise favorisce l'efficienza dei programmi che le utilizzano
- il fatto che le basi di dati siano condivise permette di ridurre ridondanze e inconsistenze
- il fatto che le basi di dati siano persistenti ne garantisce l'affidabilità
- il fatto che le basi di dati siano persistenti favorisce l'efficienza dei programmi
- il fatto che le basi di dati siano condivise rende necessaria la gestione della privatezza e delle autorizzazioni

Esercizi

Quali delle seguenti affermazioni sono vere?

- la distinzione fra DDL e DML corrisponde alla distinzione fra schema e istanza
- le istruzioni DML permettono di interrogare la base di dati ma non di modificarla
- le istruzioni DDL permettono di specificare la struttura della base di dati ma non di modificarla
- non esistono linguaggi che includono sia istruzioni DDL sia istruzioni DML
- SQL include istruzioni DML e DDL
- le istruzioni DML permettono di interrogare la base di dati e di modificarla

Esercizi

Quali delle seguenti affermazioni sono vere?

- gli utenti casuali utilizzano transazioni predefinite
- i terminalisti utilizzano transazioni predefinite
- gli utenti casuali progettano la base di dati
- i progettisti del DBMS realizzano le transazioni che saranno utilizzate dai terminalisti
- i progettisti della base di dati realizzano il DBMS
- i progettisti delle applicazioni utilizzano la base di dati come progettata dal progettista del DBMS
- i progettisti delle applicazioni utilizzano la BD come progettata dal progettista della BD

Esercizi

- Illustrare, in modo sintetico ma chiaro, supponendo di rivolgersi ad un non esperto,le caratteristiche fondamentali delle basi di dati e il ruolo che esse giocano nei sistemi informativi.
- Discutere brevemente (meno di mezza pagina) la seguente affermazione: "i dati sono una risorsa per una organizzazione, e come tali vanno considerati anche separatamente dalle applicazioni che li utilizzano."
- Illustrare brevemente (non più di mezza pagina) il concetto di indipendenza dei dati.