😷

Capitolo 1 (Introduzione)

Sistema organizzativo

Insieme di risorse e regole per lo svolgimento coordinato delle attività (processi) al fine del perseguimento di obiettivi

Le risorse di un’azienda (o ente, amministrazione) sono:

- Persone
- Denaro
- Materiali
- Informazioni

Sistema informativo

Il sistema informativo è parte del sistema organizzativo

Il sistema informativo esegue/gestisce processi informativi (cioè i processi che coinvolgono informazioni)

Esso è un componente (sottosistema) di un’organizzazione che gestisce (acquisisce, elabora, conserva, produce) le informazioni di interesse (cioè utilizzate per il perseguimento degli scopi dell’organizzazione)

Il concetto di “sistema informativo” è indipendente da qualsiasi forma di automazione:

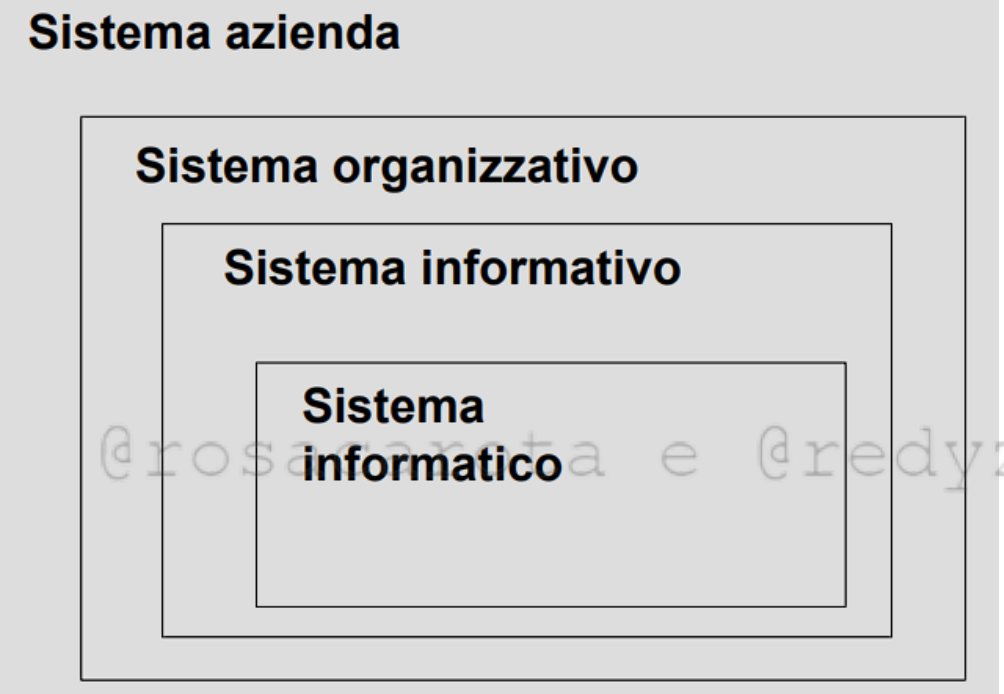
- Esistono organizzazioni la cui ragion d’essere è la gestione di informazioni (es. servizi anagrafici e banche) e che operano da secoli

Sistema Informatico

Porzione automatizzata del sistema informativo:

- La parte del sistema informativo che gestisce informazioni con tecnologia informatica

Contesto di un Sistema Informatico:



Database

Definizione 1: un database (db o base di dati) è una collezione di dati **correlati**

- Esempio: una rubrica telefonica

Definizione 2: insieme organizzato di dati per il supporto allo svolgimento di attività (di un ente, azienda, ufficio, persona)

Per “dati” si intendono dei fatti noti, con un significato **implicito**, che possono essere memorizzati

Informazioni e dati

Informazione: notizia, dato o elemento che consente di avere conoscenza più o meno esatta di fatti, situazioni, modi di essere

Dato: ciò che è immediatamente presente alla conoscenza, prima di ogni elaborazione; (in informatica) elementi di informazione costituiti da simboli che devono essere elaborati

A partire dalle elaborazioni sui dati costituisco le informazioni

Proprietà dei database

Un database deve presentare le seguenti proprietà:

- Deve essere una collezione di dati logicamente correlati con qualche significato inerente
  - Un assortimento casuale di dati non può correttamente essere considerato un database
- Deve essere progettato, costruito e riempito di dati per un utilizzo specifico. Ha una tipologia ben definita di utenti

Dimensione di un database

Un database può avere **qualsiasi dimensione** e **complessità**

Esempi:

- Una rubrica telefonica personale può avere poche centinaia di voci
- Il database dei contribuenti italiani e delle relative dichiarazioni dei redditi ha delle dimensioni notevoli:
  - Circa 20 milioni di contribuenti
  - Mediamente 5 moduli per ciascuna dichiarazione
  - 200 byte per ogni modulo ($20 * 10^6 * 200 * 5 = 2 * 10 * 10^6 * 100 * 2 * 5 = 2 * 10 * 10^6 * 10^2 * 10 = 2 * 10^{10}$ byte)
  - Ciò per un solo anno fiscale

Un esempio di database per una università

Vogliamo realizzare il database UNIVERSITÀ per gestire studenti, corsi (con prerequisiti) ed esami:

- Comprende quattro file:
  - STUDENTE: contiene i dati su ciascuno studente iscritto
  - CORSO: contiene i dati relativi a ciascun corso
  - PREREQUISITI: contiene i prerequisiti di ciascun corso
  - ESAME: contiene i voti riportati dagli studenti nei vari esami

Ogni file memorizza record di dati dello stesso tipo

Per definire il database occorre specificare **la struttura dei record** di ciascun file

Occorre cioè:

- Specificare i **campi/attributi (data element)** di ogni record
- Specificare **il tipo di ogni data element** in ciascun record

I data element: un record del file STUDENTE contiene i dati per rappresentare il **nome** dello studente, il numero di **matricola**, e **l'anno** di iscrizione corrente

STUDENTE

Cognome	Nome	Matricola	Anno
Rossi	Giuseppe	056/000484	2 f.c.
Bianchi	Antonio	056/100084	5
Verdi	Enrico	011/120579	3
…		…	…

Il tipo dei data element: gli elementi COGNOME, NOME, MATRICOLA ed ANNO sono tutti definiti come stringhe di caratteri

La definizione dell'intero database UNIVERSITÀ:

CORSO

DENOMIN.	SEMESTRE	TITOLARE
Diritto Privato	1	Amato
Procedura Civile	2	Renzi
Procedura Penale	1	Esposito
…	…	…

PREREQUISITI

DENOMIN.	PROPEDEUTICITA'
Diritto Privato	nessuna
Procedura Civile	Diritto Privato
Procedura Civile	Diritto Civile
…	…

ESAME

STUDENTE	CORSO	VOTO
056/000484	Diritto Privato	24
056/000484	Procedura Civile	28
011/120579	Procedura Penale	30
…	…	…

Per costruire il database UNIVERSITÀ memorizziamo dati di ogni studente, corso, prerequisito ed esame nel file appropriato

I record nei vari file possono essere **correlati**

Esempio:

- Il record di “Rossi” nel file STUDENTE è in relazione con due record nel file ESAME
- Il record per “Procedura Civile” nel file CORSO è in relazione con due record nel file PREREQUISITI e con un record nel file ESAME

Manipolare il database significa interrogare ed aggiornare i dati

Esempio:

- Quanti esami ha sostenuto “Rossi”? (interrogazione)
- Inserire un nuovo studente (manipolazione)
- Registrare un esame (manipolazione)
- …

Database Management System

Un **Database Management System** (DBMS) o anche **Sistema per la Gestione di Basi di Dati** è una collezione di programmi che permette di gestire basi di dati

Si tratta di un software *general-purpose* che facilita la creazione, costruzione e gestione di database per diverse applicazioni

Permette di memorizzare informazioni in strutture di dati efficienti, scalabili e flessibili

Funzionalità di un DBMS

Un DBMS permette di:

- **Definire** un database
  - Specificando i dati da memorizzare nel database: i tipi di dati, le strutture, i vincoli
- **Costruire** il database
  - Memorizzando i dati stessi su qualche memoria di massa controllata dal DBMS
- **Manipolare** il database
  - Interrogare il database per ritrovare dati specifici
  - Aggiornare il database per riflettere cambiamenti nel mondo reale
  - Generare dei report dei dati

DBMS ≠ Database

Un DBMS è un applicativo per gestire database

- Esempio: MS Access, Oracle, MySQL

Un database è una collezione di dati correlati

- Esempio: file con estensione .MDB (con Access)

DBMS special-purpose

Non è necessario usare un DBMS **general-purpose**

Si può anche scrivere un proprio insieme di programmi per gestire i dati, creando un DBMS **special-purpose**

Tale approccio può essere vantaggioso nello sviluppo di piccole applicazioni, evitando lunghi tempi di **overhead** e ottenendo tempi di risposta più brevi

Metadati vs Database

Un **metadato** è un dato che descrive un altro dato, ad esempio, "DD/MM/YYYY" è un formato di data, costituito da due caratteri numerici per il giorno, uno / separatore, due caratteri per il mese, uno / separatore e quattro caratteri per l'anno

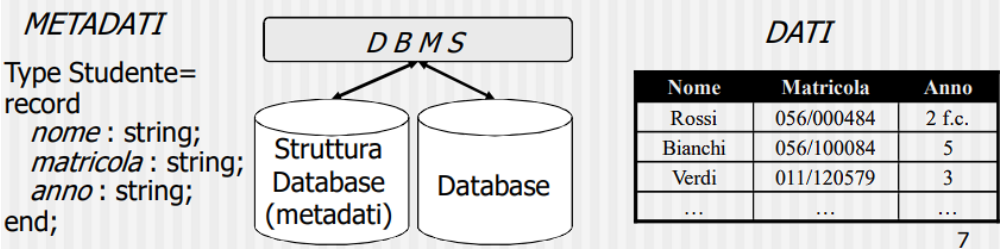
Altro esempio è la descrizione di un cognome come una stringa alfabetica di 20 caratteri

Il DBMS memorizza non solo i dati, ma anche la **definizione completa** (o descrizione) nel database

Le informazioni sulla definizione, dette **metadati**, sono memorizzate nel **catalogo di sistema**

Il DBMS accede al catalogo per conoscere la struttura dei file dello specifico database

Stiamo quindi utilizzando più spazio per la memorizzazione dei metadati



Proprietà di un Database

Dati persistenti: hanno un tempo di vita indipendente dall'esecuzione dei programmi che li utilizzano. I dati devono persistere anche in presenza di errori

Dati condivisi: i vari settore di un'azienda possono condividere dati. Questo può comportare i seguenti problemi:

- **Ridondanza** (informazioni ripetute)
- Rischio di **incoerenza** (diverse versioni degli stessi dati possono non coincidere)

DBMS Multiutente

Un DBMS multiutente deve consentire accesso al database a più utenti contemporaneamente

Deve includere software per:

- Il **controllo dell'accesso** (chi è autorizzato ad accedere o modificare quali dati)
- Il **controllo della concorrenza**, per garantire l'aggiornamento corretto (esempio: il problema della prenotazione di posti per una compagnia aerea)

Affidabilità

I DBMS devono garantire **affidabilità** (per le basi di dati)

- Resistenza a malfunzionamenti hardware e software

Una base di dati è una risorsa preziosa e quindi deve essere conservata a lungo termine

Tecnica fondamentale:

- Gestione delle **transazioni**

Transazioni (per l’utente)

Programmi che realizzano attività frequenti e predefinite, con poche eccezioni, previste a priori

Esempi:

- Versamento presso uno sportello bancario
- Emissione di certificato anagrafico
- Dichiarazione presso l’ufficio di stato civile
- Prenotazione aerea

Le transazioni sono di solito realizzate in linguaggio **host** (tradizionale o ad hoc), possono tuttavia essere realizzate anche con linguaggi general-purpose

Transazioni, due accezioni

Per l’utente:

- Programma a disposizione, da eseguire per realizzare una funzione di interesse

Per il sistema:

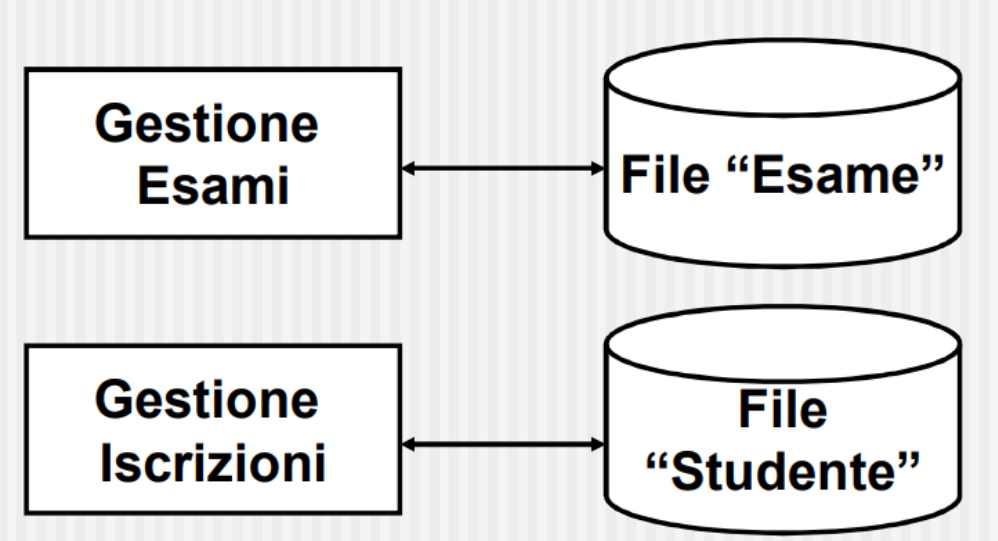
- Sequenza indivisibile (atomica) di operazioni che deve disporre di determinate caratteristiche - tutte le operazioni devono essere necessariamente svolte, in caso di errore tutti i progressi devono essere annullati

DBMS vs File Processing

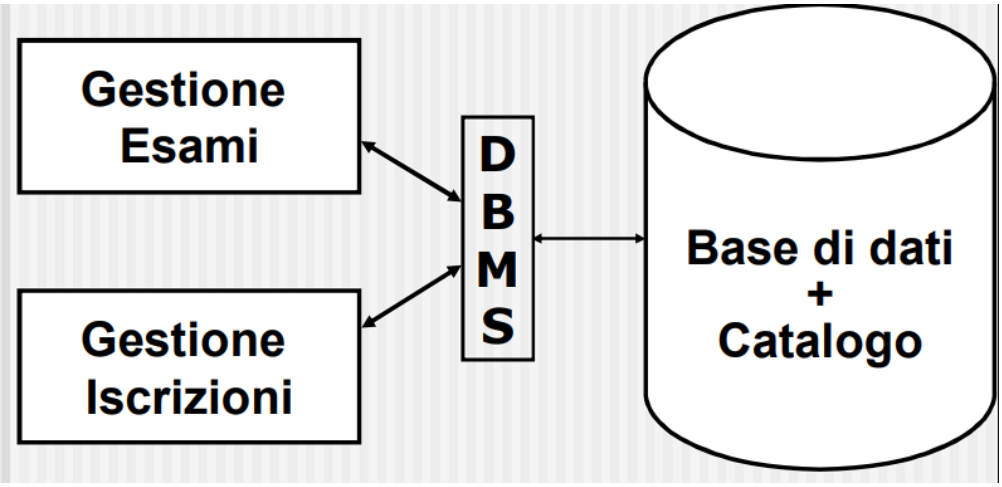
Nei programmi tradizionali i dati vengono conservati in un file. Ogni programma contiene una descrizione della struttura del file stesso, con conseguenti rischi di incoerenza fra le descrizioni (ripetute in ciascun programma) e i file stessi

Nei DBMS, esiste una porzione della base di dati (il **catalogo** o **dizionario**) che contiene una descrizione centralizzata dei dati **(metadati)** che può essere utilizzata dai vari programmi

Approccio File-Processing:



Approccio DBMS:



DBMS: Pro e Contro

Pro:

- Gestione centralizzata con possibilità di standardizzazione ed “economia di scala”
- Disponibilità di servizi integrati
- Riduzione di ridondanze ed inconsistenze
- Minor accoppiamento programma/dati (favorisce sviluppo e manutenzione delle applicazioni)
- Manutenzione semplificata

Contro:

- Costo del DBMS
- Overhead computazionale

File Processing: Pro e Contro

Pro:

- Evita l’acquisto del DBMS (difficile da ammortizzare per piccole applicazioni)
- Possibilità di scrivere programmi efficienti

Contro:

- Ridondanza dei dati, comporta elevato rischio di inconsistenze nei dati
- Forte accoppiamento programma/dati (maggiori difficoltà di sviluppo e manutenzione delle applicazioni)

Rappresentazione dei dati con i DBMS

Ciascun DBMS supporta un determinato **modello dei dati**

Occorre quindi rappresentare i dati in modo conforme al modello supportato dal DBMS scelto

Sarà cura del DBMS memorizzare in file i dati espressi secondo il modello

I dati sono quindi rappresentati a vari livelli di astrazione. Tuttavia il database vero e proprio è solo quello residente su file

Applicazioni e DBMS

I programmi applicativi fanno riferimento alle strutture del modello

Ciò consente di modificare la rappresentazione dei dati a livello più basso senza dover modificare i programmi **(indipendenza fisica)**

Modelli dei dati

Un modello dei dati è un insieme di costrutti utilizzati per descrivere i dati e le strutture atte a memorizzarli, nascondendo alcuni dettagli di memorizzazione

- Ad esempio, il **modello relazionale** fornisce il costrutto **relazione (tabella)** che permette di definire insiemi di record omogenei

ESAME

STUDENTE	CORSO	VOTO
056/000484	Diritto Privato	24
056/000484	Procedura Civile	28
011/120579	Procedura Penale	30
…	…	…

Un modello deve:

- Rappresentare una certa realtà
  - Es. una mappa rappresenta una porzione di territorio. Quindi è un modello del territorio, che rappresenta alcune caratteristiche, nascondendone altre
- Fornire un insieme di strutture simboliche per rappresentare una realtà, nascondendo alcuni dettagli
  - Es. una mappa ha una serie di simbolismi grafici per rappresentare gli aspetti salienti di un territorio

Due principali tipi di modelli

Modelli **logici**:

- Adottati nei DBMS per l’organizzazione dei dati

- Esempi: modello **relazionale**, **reticolare**, **gerarchico**, **ad oggetti**

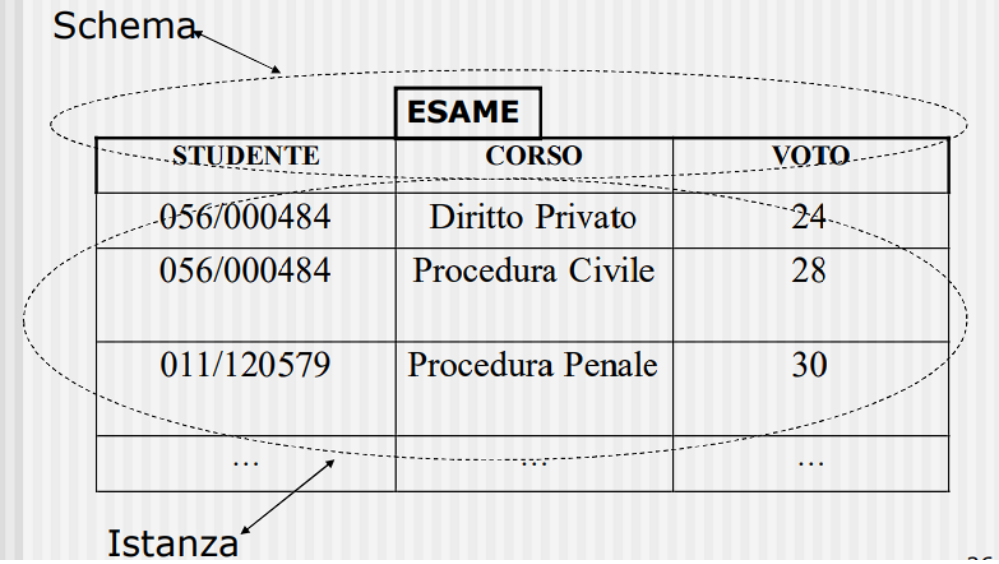
Modelli **concettuali**:

- Permettono di rappresentare i dati in modo indipendente da come verranno memorizzati
- Descrivono i concetti del mondo reale
- Sono utilizzati nelle fasi preliminari di progettazione
- Il più diffuso è il modello **Entità-Relazioni**

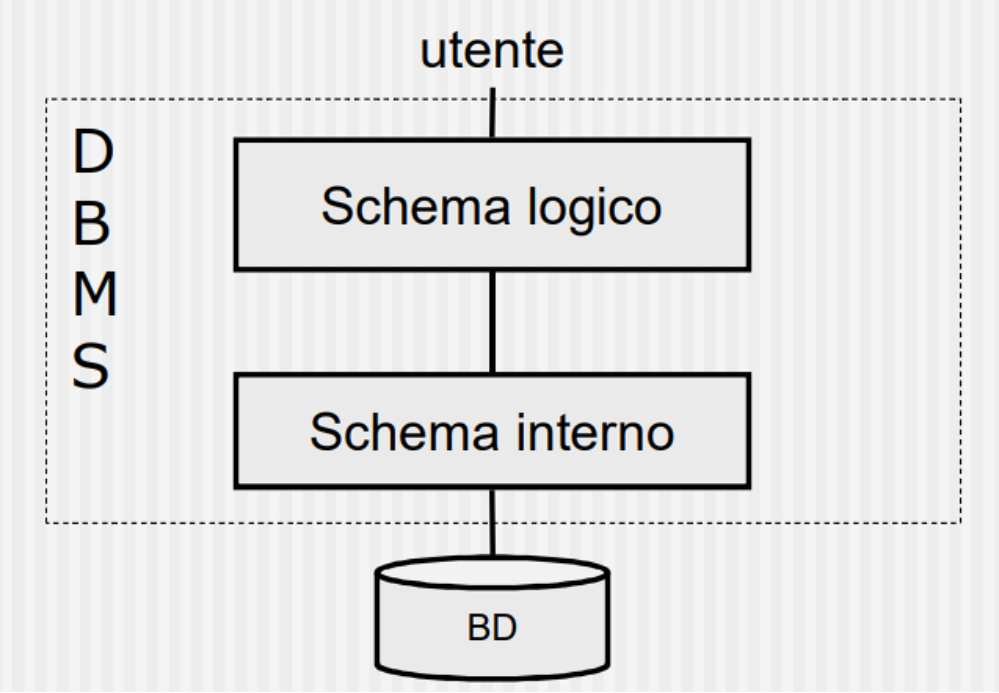
Schema vs Istanza di un DB

In ogni base di dati esistono:

- Lo **schema**, sostanzialmente invariante nel tempo, che ne descrive la struttura (aspetto intensionale), viene fatto seguendo la rappresentazione di un modello
  - Es. le intestazioni delle tabelle
- L'**istanza**, rappresenta i valori memorizzati e possono quindi cambiare anche molto rapidamente (aspetto estensionale), prende anche il nome di **aspetto estensionale**
  - Es. il "contenuto" di una tabella



Architettura semplificata di un DBMS



Schema logico: cioè che vede anche l'utente, descrizione della base di dati secondo il modello logico (ad esempio attraverso tabelle):

- Type Studente = record

  nome: string;

  matricola: string;

  anno: string;

  end;

- È una descrizione dell'intera base di dati per mezzo del modello logico adottato dal DBMS (ad es. relazionale)

Schema interno (o fisico): rappresentazione dello schema logico per mezzo di strutture di memorizzazione (ad esempio file, record con puntatori ordinati in un certo modo)

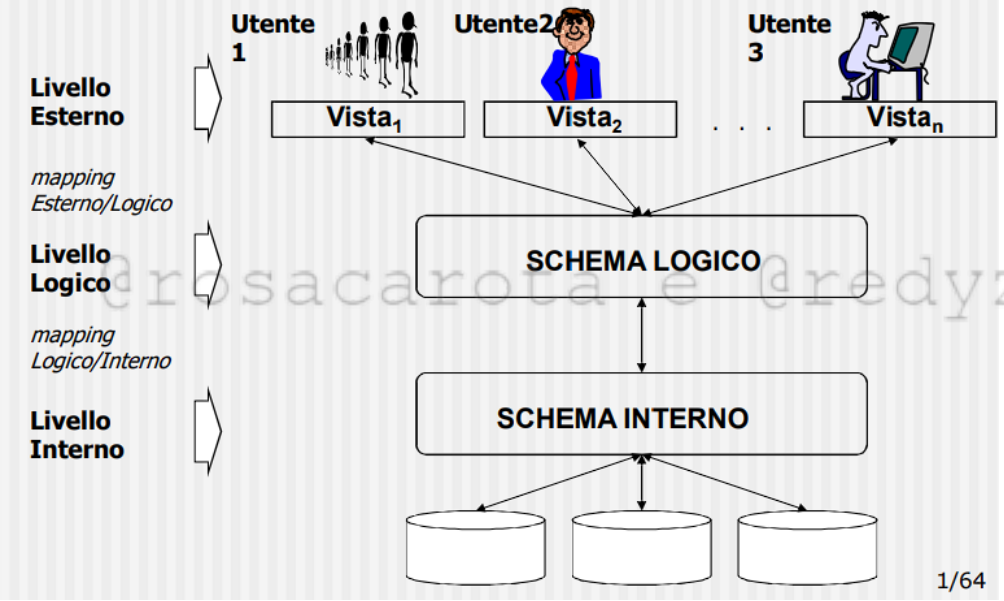
Indipendenza dei dati

Il livello logico è indipendente da quello fisico **(indipendenza fisica)**:

- Una tabella è utilizzata nello stesso modo qualunque sia la sua realizzazione fisica (che può anche cambiare nel tempo)

Posso cambiare il livello fisico senza dover cambiare il livello logico che avevo costruito

Architettura a 3 livelli per DBMS standard ANSI/SPARC



Posso permettere ad utenti diversi di visualizzare porzioni diverse di database, in maniera spesso semplificata

Schema logico: descrizione dell’intera base di dati nel modello logico del DBMS

Schema interno (o fisico): rappresentazione dello schema logico per mezzo di strutture fisiche di memorizzazione

Schema esterno: descrizione di una porzione della base di dati, per mezzo del modello logico in un modello dati di alto livello **(viste)**

Esempio:

Proprietà del Livello Esterno

Definisce un sottoinsieme del database per una particolare applicazione

Include più schemi esterni o viste utente

Usa un modello dati di alto livello

Ogni schema esterno descrive la parte del database a cui è interessato un particolare gruppo di utenti e, nasconde il resto del database a quel gruppo

Esempio:

- La segreteria didattica ha necessità di vedere tutte le informazioni sugli esami, ma non deve vedere informazioni sugli stipendi

Esempio di vista:

ESAME-DOCENTE

STUDENTE	DOCENTE	VOTO
056/000484	Rossi	24
056/000484	Bianchi	28
011/120579	Rossi	30
…	…	…

I dati per costruire la vista sono stati estratti dalle tabelle **esame** e **corso**

I Mapping tra i livelli

I tre schemi sono solo delle **descrizioni** di dati: gli unici dati che realmente esistono sono a livello fisico

Un mapping è un processo di trasformazione delle richieste e dei risultati

Il DBMS trasforma:

- Una richiesta specificata su uno schema esterno
    - In una richiesta secondo schema logico e poi
        - In una richiesta sullo schema interno per procedere sul database memorizzato
- I risultati seguono un percorso inverso

Indipendenza dei dati

L’architettura a tre livelli è utile per evidenziare il concetto di indipendenza dei dati, ovvero la capacità di cambiare lo

schema a un livello del database senza dover cambiare lo schema al livello superiore

Si definiscono due tipi di indipendenza dei dati:

- **Indipendenza logica** dei dati:
  - Lo schema logico può essere cambiato senza dover cambiare gli schemi esterni o i programmi applicativi
- **Indipendenza fisica** dei dati:
  - Lo schema interno può essere cambiato senza dover cambiare gli schemi logici (o esterni)

Indipendenza logica

Lo schema logico può essere cambiato per modificare, espandere o ridurre il database (aggiungendo o rimuovendo, rispettivamente, un tipo di record o data item)

Se si elimina un tipo di record, gli schemi esterni che si riferiscono ai soli dati restanti, non devono essere alterati

In caso di modifiche, solo le definizioni delle viste ed i mapping devono essere cambiati; i programmi applicativi che fanno riferimento agli schemi esterni sono indifferenti alle modifiche

Esempio:

Lo schema esterno

ESAMI SUPERATI	Nome	Matricola	ESAMI	
			DENOMIN.	VOTO
	Rossi	056/000484	Diritto Privato	24
			Procedura Civile	28
	Verdi	011/120579	Procedura Penale	30
			Diritto Civile	27

non dovrebbe essere alterato cambiando il file ESAME da

ESAME

NOME	DENOMIN.	VOTO
Rossi	Diritto Privato	24
Rossi	Procedura Civile	28
Verdi	Procedura Penale	30
...	...	...

a

ESAME

NOME	MATRICOLA	DENOMIN.	VOTO
Rossi	056/000484	Diritto Privato	24
Rossi	056/000484	Procedura Civile	28
Verdi	056/100084	Procedura Penale	30
...		...	...

38

Indipendenza fisica

Lo schema fisico (interno) può essere cambiato senza dover cambiare lo schema logico (e quindi anche quello esterno)

Un cambiamento dello schema fisico può essere dovuto alla riorganizzazione di qualche file (es. creando ulteriori strutture di accesso), per migliorare l'esecuzione del ritrovamento o dell'aggiornamento

Se i dati non subiscono alterazioni non è necessario cambiare lo schema logico

L'indipendenza fisica si ottiene più facilmente di quella logica

Esempio:

- Fornire un percorso di accesso per migliorare il ritrovamento dei record del file *corso* con *Denominazione* e *Semestre* non dovrebbe richiedere variazioni ad una query del tipo "ritrova

tutti i corsi tenuti nel secondo semestre”, sebbene la query possa essere eseguita in maniera più efficiente

Indipendenza dei dati (libro)

L’architettura a livelli garantisce l’indipendenza dei dati

- Questa proprietà permette a utenti e programmi applicativi che utilizzano la base di dati di interagire a un elevato livello di astrazione

Può essere caratterizzata come **indipendenza fisica** e **indipendenza logica**

- L’indipendenza fisica consente di interagire con il DBMS in modo indipendente dalla struttura fisica dei dati. In base a questa proprietà è possibile modificare le strutture fisiche senza influire sulle descrizioni dei dati ad alto livello e quindi sui programmi che utilizzano i dati stessi
- L’indipendenza logica consente di interagire con il livello esterno della base di dati in modo indipendente dal livello logico. Per esempio è possibile aggiungere uno schema esterno senza dover modificare lo schema logico; dualmente è possibile modificare il livello logico, mantenendo inalterate le strutture esterne (modificandone ovviamente la definizione in termini delle strutture logiche)

Architettura a tre livelli: vantaggi e svantaggi

Vantaggi:

- L’architettura a tre livelli consente facilmente di ottenere una reale indipendenza dei dati
- Il database risulta più flessibile e scalabile

Svantaggi:

- Il catalogo del DBMS multi-livello deve avere dimensioni maggiori, per includere informazioni su come trasformare le richieste e i dati tra i vari livelli
- I due livelli di mapping creano un overhead durante la compilazione o esecuzione di una query di un programma, causando inefficienze nel DBMS

Le professioni relative ai DB

Database Administrator (DBA):

- Il DBA è responsabile per autorizzare l'accesso al database, coordinare e monitorare il suo uso, acquisire nuove risorse hardware e software
- In grosse organizzazioni è assistito da uno staff

Progettista:

- Responsabile dell'individuazione dei dati da memorizzare nel DB e della scelta delle strutture
- Deve capire le esigenze dell'utenza del DB e giungere a un progetto che soddisfi questi requisiti
- Sviluppa viste dei dati
- Il DB finale deve essere in grado di supportare i requisiti di tutti i gruppi di utenti

Utenti finali:

- Utenti finali **causali**: accedono occasionalmente al DB e lo interrogano attraverso linguaggi sofisticati
- Utenti finali **naive** o **parametrici**: rappresentano una parte considerevole di utenza. Usano aggiornamenti e query standard

- Utenti finali **sofisticati**: ingegneri, scienziati e analisti di affari che hanno familiarità con le facility del DBMS per richieste complesse

Analisti di sistema e programmatori di applicazioni:

- Gli analisti di sistema determinano i requisiti degli utenti finali e sviluppano specifiche per le transazioni
- I programmatori di applicazioni implementano le specifiche come programmi

Altre figure:

- **Progettisti e implementatori di DBMS:**
  - Disegnano e implementano moduli e interfacce di DBMS come pacchetti software. Un DBMS è un complesso sistema software che consiste di molti moduli
- **Sviluppatori di Tool:**
  - Implementano pacchetti software per il progetto, il monitoraggio, l'interfaccia, la prototipazione, ecc. di database
- **Operatori e personale per la manutenzione**
  - Personale che si occupa dell'amministrazione del sistema e della manutenzione hardware e software del sistema di database. Spesso queste figure appartengono allo staff del DBA

Linguaggi per i database

Un DBMS deve fornire ad ogni categoria di utenti interfacce e linguaggi adeguati per comunicare gli schemi di database progettati e per manipolare i dati. Inoltre, nei DBMS **multilivello**, occorre specificare il mapping tra gli schemi

Nei DBMS dove non c’è una netta separazione tra livelli, progettisti e DBA usano un **linguaggio di definizione dati** o **data definition language (DDL)** per definire tutti gli schemi

Il compilatore del DDL (contenuto nel DBMS) ha la funzione di:

- Trattare gli **statement DDL** per identificare descrizioni dei costrutti di schema
- Memorizzare la descrizione dello schema nel catalogo del DBMS

Dove c’è una chiara distinzione tra livello logico ed interno si usa:

- Uno **storage definition language (SDL)** per specificare lo schema interno
- Il DDL per specificare soltanto lo schema logico

I mapping tra i due schemi possono essere specificati in uno qualsiasi dei due linguaggi

Nelle architetture a tre livelli viene usato un **linguaggio di definizione viste** o **view definition language (VDL)** per specificare le viste utente e i loro mapping sullo schema logico

Una volta che gli schemi sono compilati ed il DB viene popolato (riempito) con i dati, il DBMS fornisce un **linguaggio di manipolazione dati** o **data manipulation language (DML)** per consentire agli utenti di manipolare i dati (cioè recuperare, inserire, cancellare ed aggiornare)

Nei DBMS moderni si tende ad utilizzare un unico linguaggio integrato che comprende costrutti per:

- La definizione di schemi logici
- La definizione di viste
- La manipolazione dei DB
- La definizione di strategie di memorizzazione

Esempio:

- Il linguaggio dei database relazionali **SQL** rappresenta una combinazione di DDL, VDL, DML e SDL

Tipi di DML

Esistono due tipi di DML:

- DML di alto livello o non procedurali
  - Consentono di specificare in maniera concisa operazioni complesse sui database
- DML di basso livello o procedurali
  - Devono essere inglobati in un linguaggio di programmazione general-purpose
  - Sono detti **record-at-a-time** perché tipicamente ritrovano record individuali nei DB e li trattano separatamente; hanno quindi bisogno di usare costrutti di programmazione come l'iterazione per trattare il singolo record da un insieme di record

In molti DBMS gli statement di DML di alto livello presentano le seguenti caratteristiche:

- Possono essere specificati interattivamente da terminale
- Possono essere inglobati in un linguaggio di programmazione general-purpose
- Sono detti **set-at-a-time** o **set-oriented** perché possono specificare e ritrovare molti record in un singolo statement DML (es. SQL)
- Sono detti dichiarativi perché una query di un DML ad alto livello spesso specifica quale dato deve essere ritrovato piuttosto che come ritrovarlo

Quando il linguaggio DML (di alto o basso livello) è inglobato in un linguaggio general-purpose **(l'host language)** esso è detto data **sublanguage**

Se un linguaggio DML di alto livello è usato da solo in maniera interattiva, esso è detto **query language**. Un query language di alto livello è usato da utenti occasionali per specificare le proprie richieste

Gli utenti naive o parametrici hanno in genere a disposizione delle interfacce amichevoli per interagire col DB

Interfacce per DBMS

Interfacce Menu-based:

- Presentano all'utente liste di opzioni, dette menù, che guidano l'utente nella formulazione di una richiesta. La query è composta un passo alla volta scegliendo opzioni da una lista di menù che viene visualizzata dal sistema
- Non c'è necessità di memorizzare i comandi specifici e la sintassi di un query language
- I menù pull-down sono spesso usati nelle interfacce browsing, che consentono all'utente di guardare i contenuti del DB in maniera non strutturata

Interfacce Forms-based:

- Un'interfaccia forms-based visualizza un form per ciascun utente. Gli utenti possono:
  - Riempire tutte le entrate del form completamente per inserire nuovi dai
  - Possono riempire solo alcune entrate, nel qual caso il DBMS ritroverà dati che fanno matching per il resto delle

entrate

- I form sono in genere progettati e programmati per utenti naive
- Il **form specification language**, che molti DBMS hanno, aiuta i programmatori a specificare i form

Interfacce per DBA:

- La maggior parte dei DBMS contengono comandi privilegiati che possono essere usati solo dallo staff del DBA (es. creare account, impostare dei parametri, cambiare uno schema, riorganizzare la struttura del DB)

Utility di un DBMS

Loading utilities:

- Per la conversione di dati da formati esterni al formato di specifico del DBMS

Performance Monitor utilities:

- Per monitorare il DB e costruire statistiche per il DBA

Backup utility:

- Per creare copie di backup del DB su dispositivi di streaming

File Organization utility:

- Per riorganizzare i file del database, allo scopo di migliorare le prestazioni