



Corso di **Basi di Dati**

Lezione 2:

Concetti di base ed Architetture dei DBMS

Capitolo 1 del libro di testo

Prof. G. Polese
a.a. 2022/2023

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.
- E' 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 dai dati.

DBMS \neq 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

Stessa differenza esistente tra Word (applicativo) e file .DOC (dati)

Principali DBMS

- > *Oracle*
- >  (DB2, Universal Server)
- > **Microsoft** (Access, SQL Server)
- > 
- > InForm*i*X
- > MySQL (Freeware)

DBMS special-purpose

- Non è necessario usare un DBMS **general-purpose**. E' anche possibile 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.

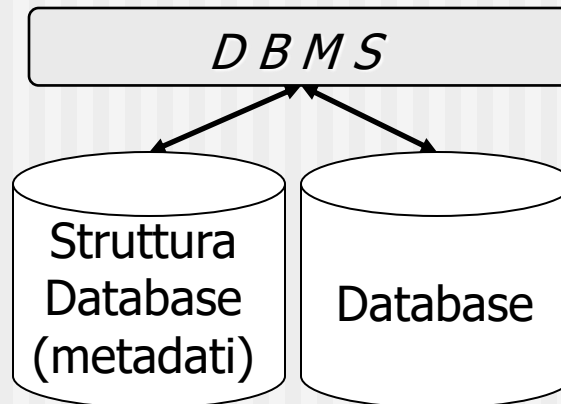
Metadati vs Database

- Il DBMS memorizza non solo i dati, ma anche la **definizione completa** (o descrizione) del 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.

METADATI

Type Studente=
record

```
nome : string;  
matricola : string;  
anno : string;  
end;
```



DATI

Nome	Matricola	Anno
Rossi	056/000484	2 f.c.
Bianchi	056/100084	5
Verdi	011/120579	3
...

Proprietà di un DataBase

- *Dati persistenti*: hanno un tempo di vita indipendente dall'esecuzione dei programmi che li utilizzano
- *Dati condivisi*: i vari settori 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)

Transazioni, due accezioni

■ Per l'utente:

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

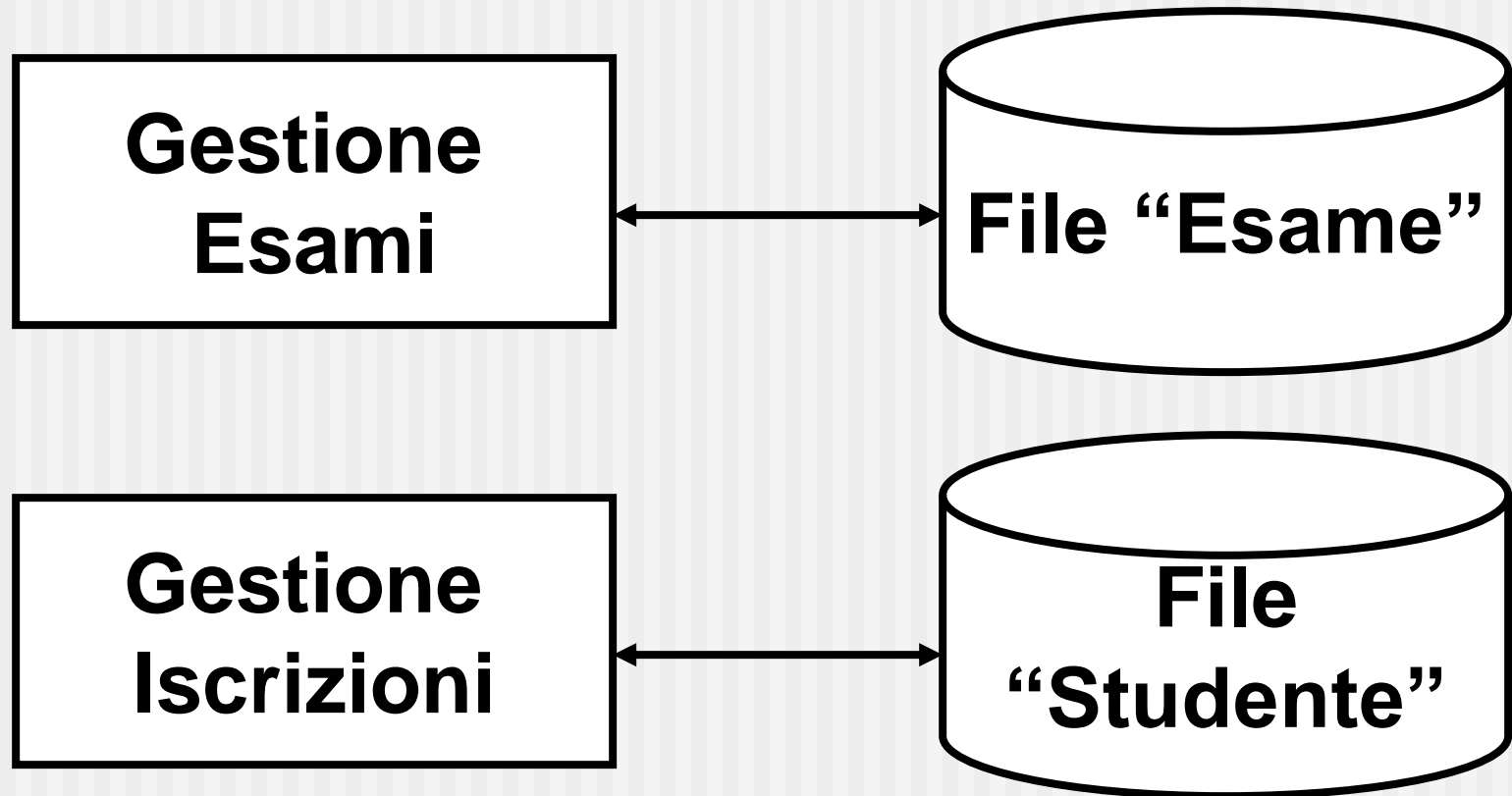
■ Per il sistema:

- sequenza indivisibile di operazioni

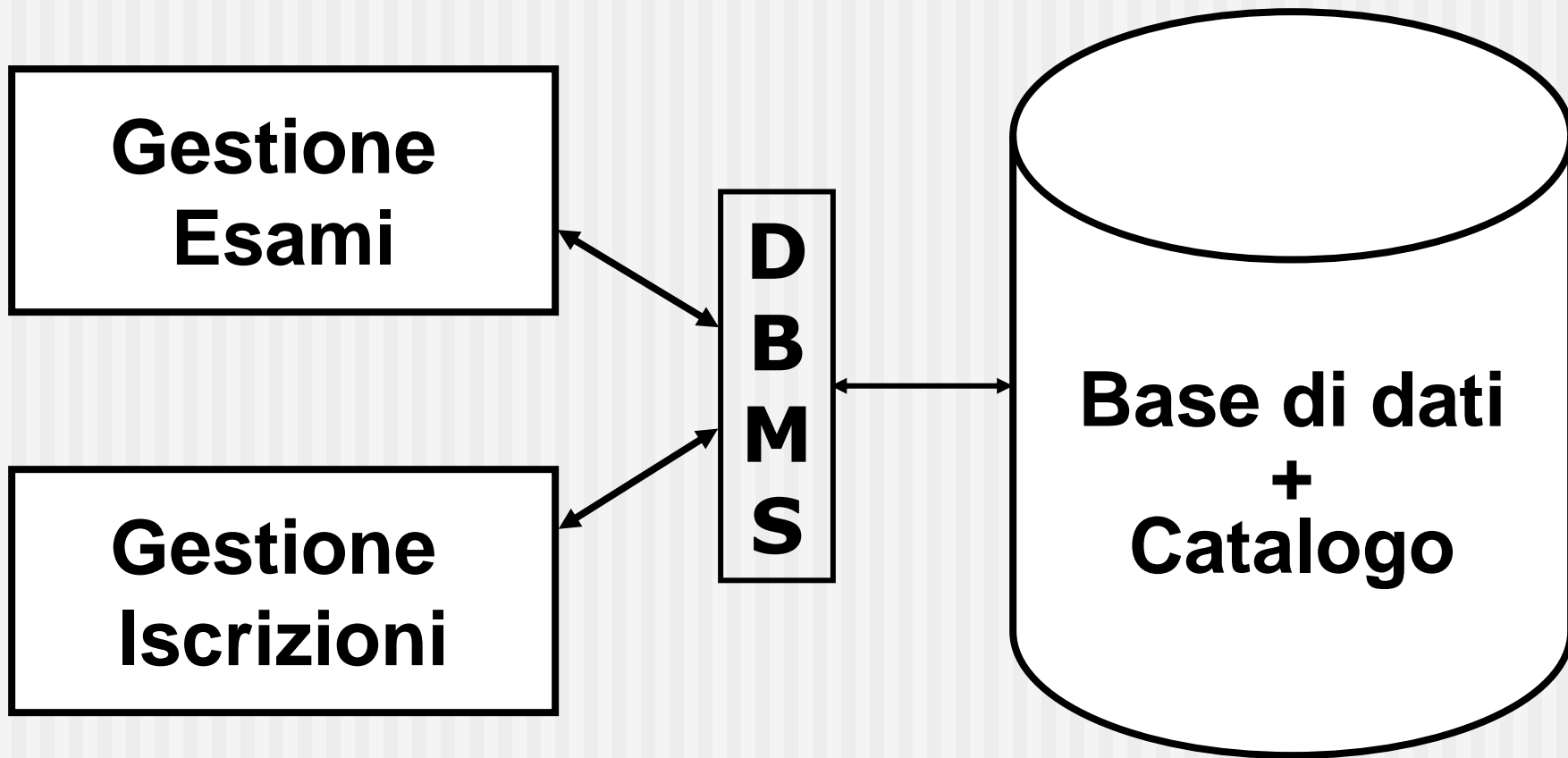
DBMS vs File Processing

- 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 (Meno efficienza) 16

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

- Sappiamo rappresentare i dati nei file. Ma come si rappresentano 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. Ma 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

Cos'è un modello (in genere)

Un modello deve:

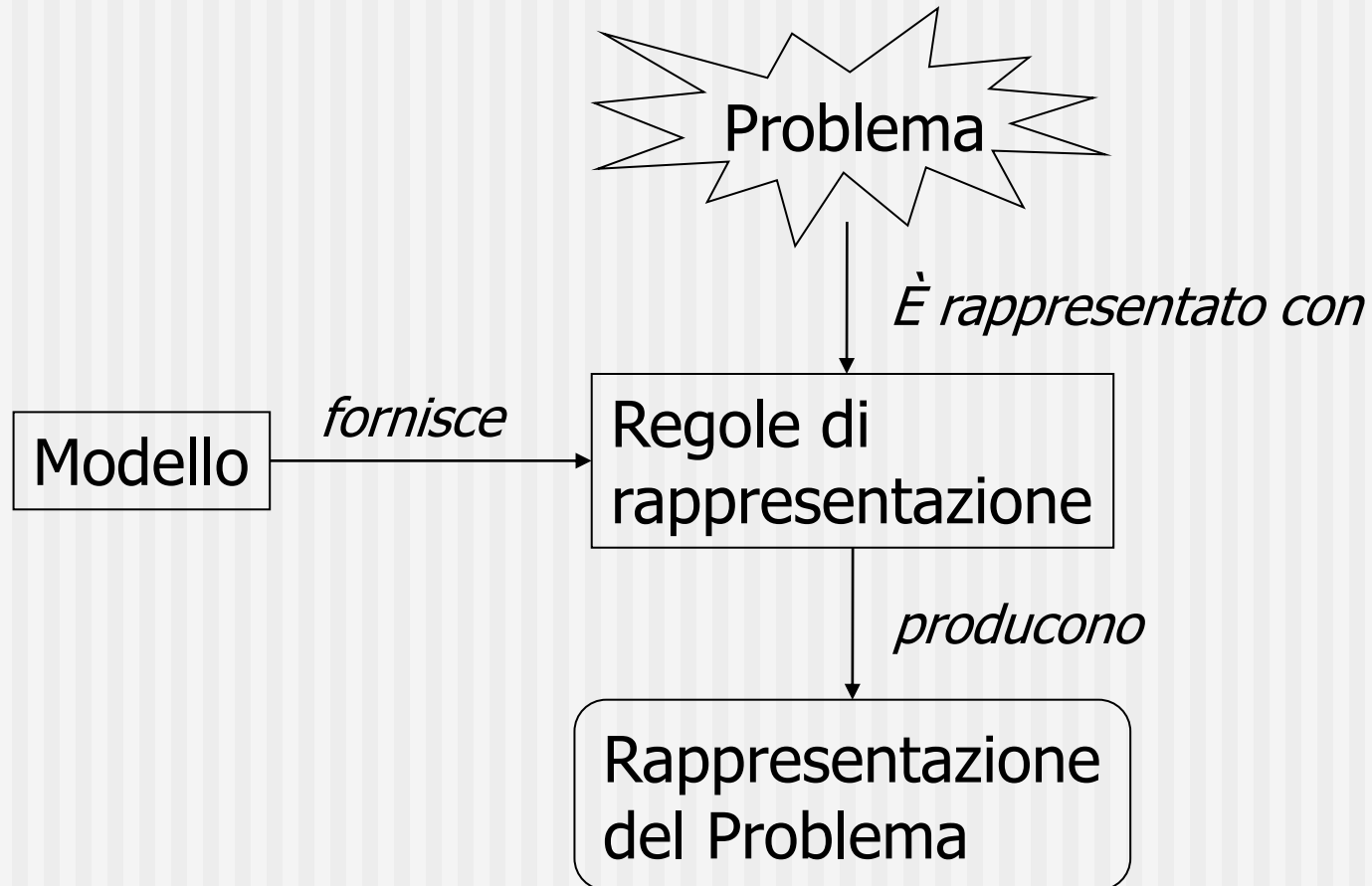
- Rappresentare una certa realtà
 - Es. Una mappa rappresenta una porzione di territorio. E' quindi un modello del territorio, che rappresenta alcune caratteristiche, nascondendone altre
- Fornire un insieme di strutture simboliche per rappresentare una realtà, nascondendo alcuni dettagli.
 - Es. Una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

Modello dei dati

- 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
...

Come si usa un modello



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)
 - es.: le intestazioni delle tabelle
 - l'**istanza**, rappresenta i valori memorizzati e possono quindi cambiare anche molto rapidamente (aspetto estensionale)
 - es.: il "contenuto" di una tabella

Schema e Istanza

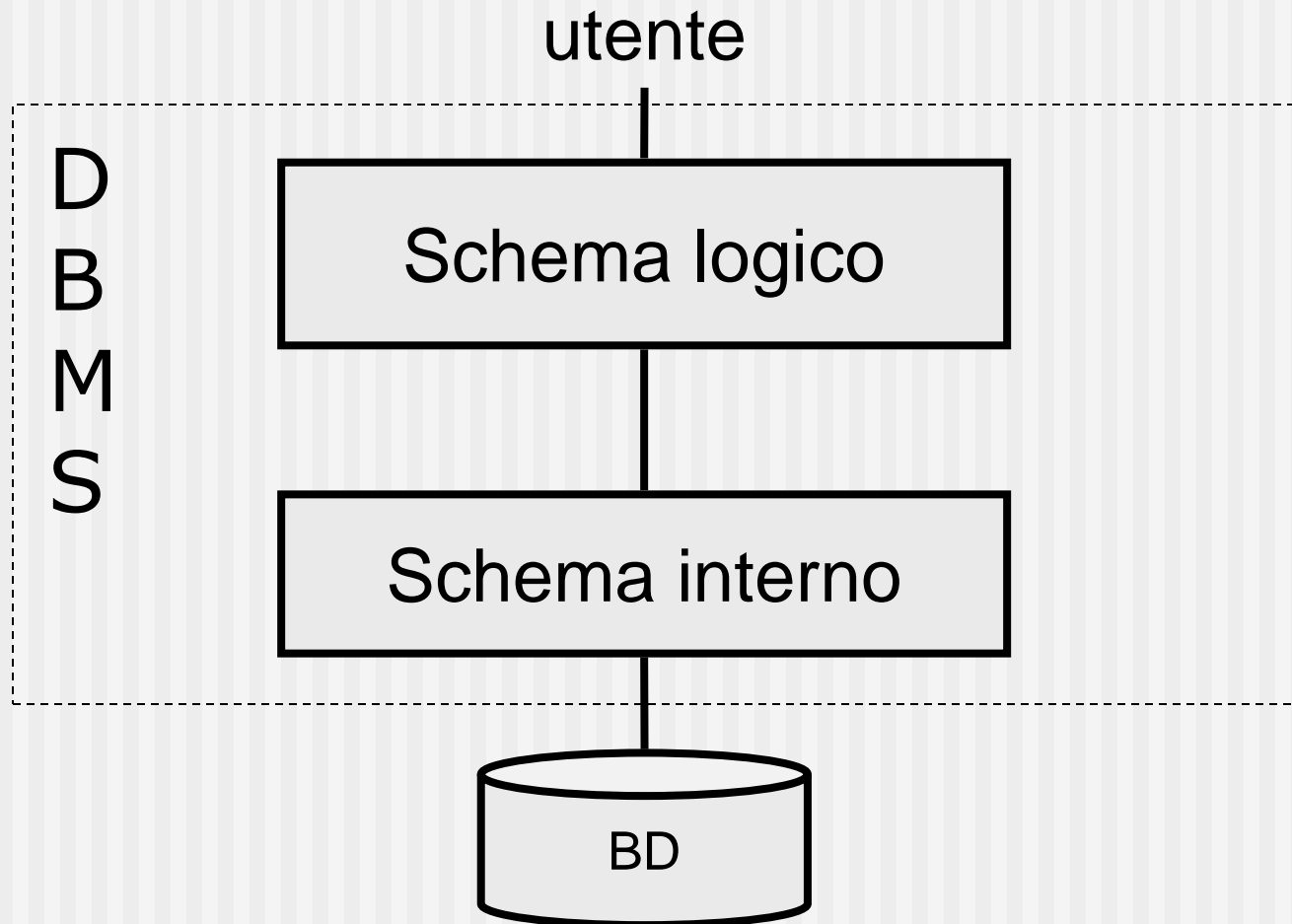
Schema

ESAME

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

Istanza

Architettura semplificata di un DBMS



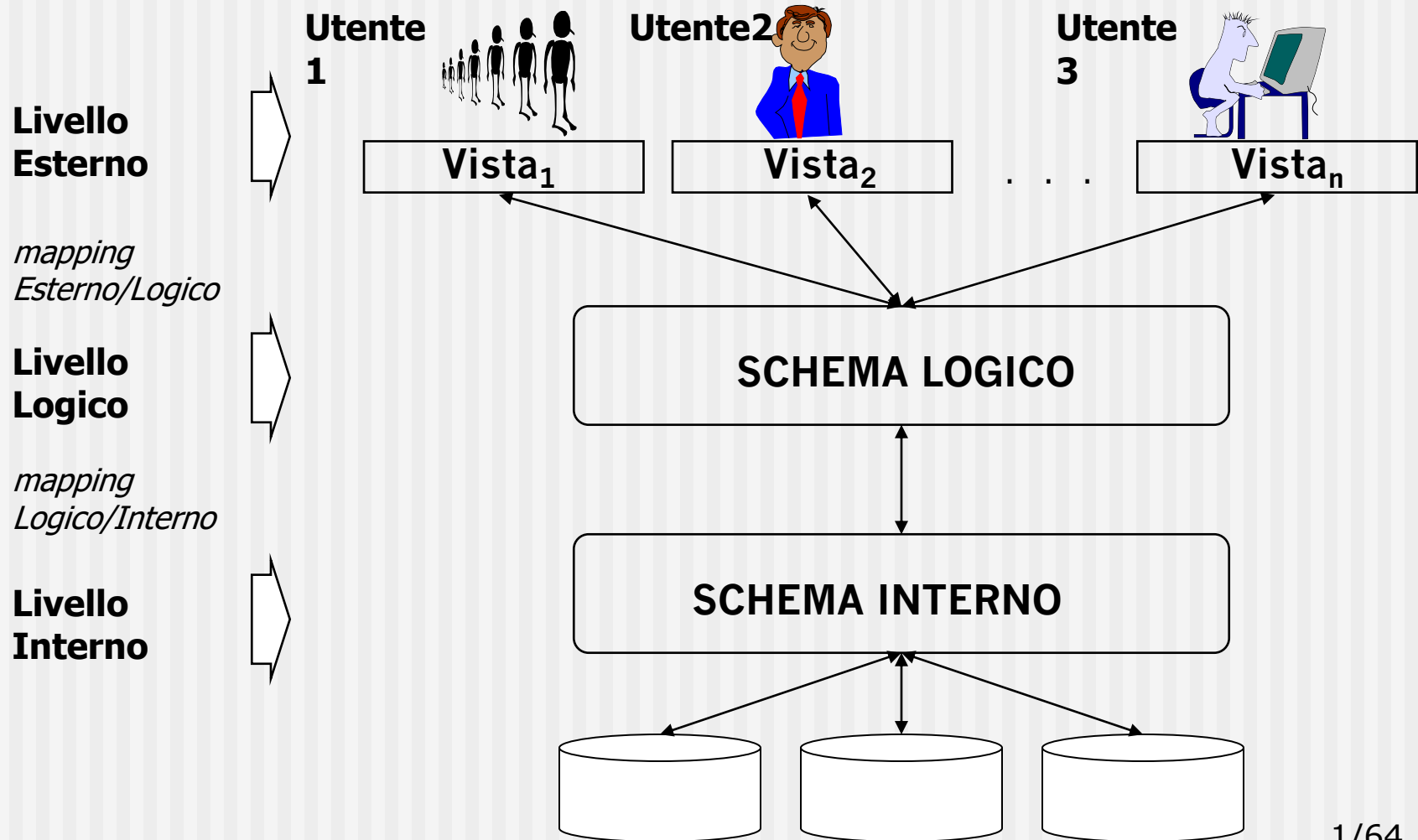
Architettura semplificata di un DBMS: schemi

- **schema logico**: descrizione della base di dati secondo il modello logico (ad es., attraverso tabelle):
 - Type Studente= record
 - nome* : string;
 - matricola* : string;
 - anno* : string;
 - end;
- **schema interno** (o **fisico**): rappresentazione dello schema logico per mezzo di strutture di memorizzazione (ad es., 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)
- Perciò in questo corso vedremo solo il livello logico e non quello fisico.

Architettura a 3 livelli per DBMS standard ANSI/SPARC



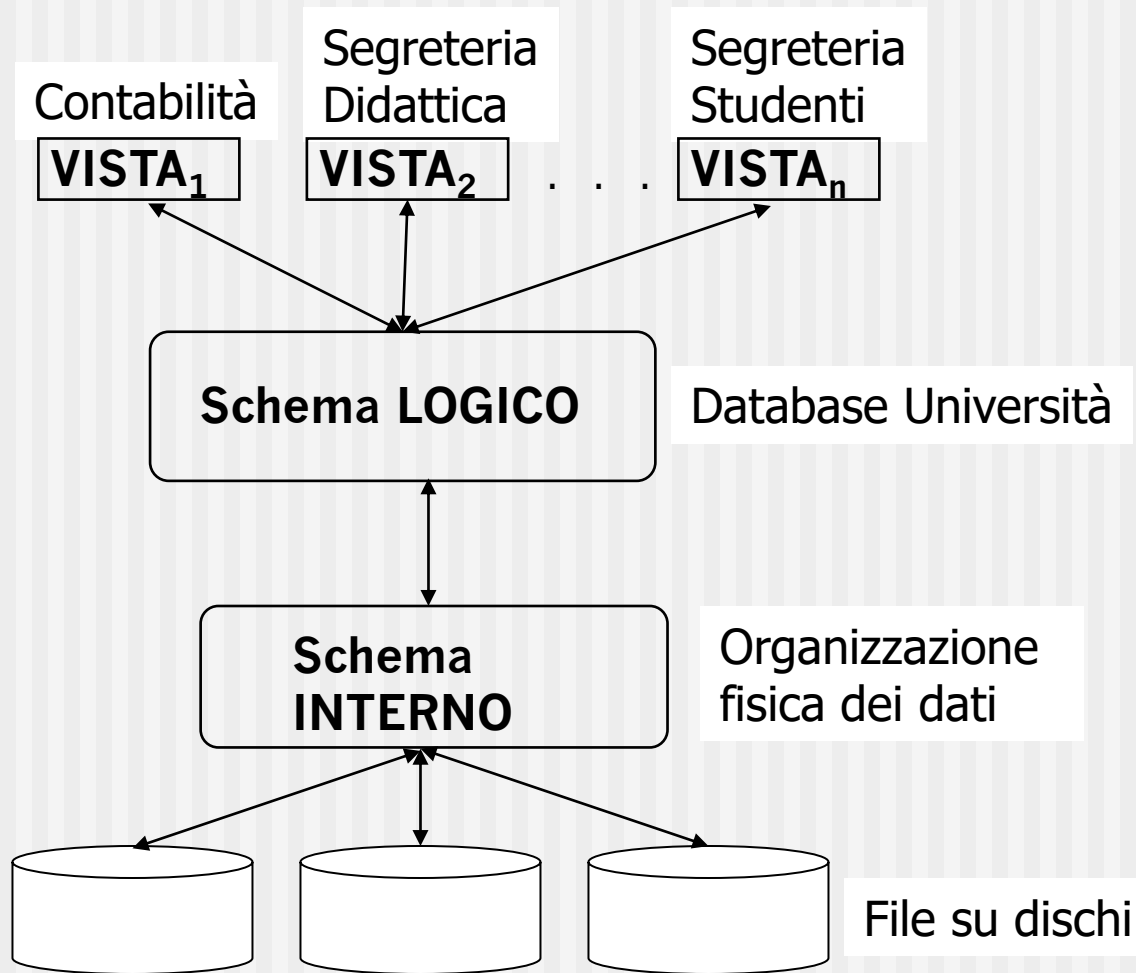
Architettura ANSI/SPARC: schemi

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 in un modello dati di alto livello (viste).

L'ARCHITETTURA a 3 livelli: ESEMPIO



Proprietà del Livello Esterno

- Definisce un sottoinsieme del db 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 DB cui è interessato un particolare gruppo di utenti e nasconde il resto del DB 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 3 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

- **Indipendenza logica dei dati:**

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.

INDIPENDENZA LOGICA: 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

- Indipendenza fisica dei dati:
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.

INDIPENDENZA FISICA: 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.

ARCHITETTURA A 3 Livelli: VANTAGGI E SVANTAGGI

VANTAGGI

- L'architettura a 3 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 di 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

I Professionisti 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

- E' 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.

I Professionisti DB (2)

■ Utenti finali

- utenti finali casuali: 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

I Professionisti DB (3)

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 hw e sw del sistema di database. Spesso queste figure appartengono allo staff del DBA.

LINGUAGGI PER I DATABASE

Linguaggi di DBMS

- 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.

Linguaggi di DBMS (2)

- 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 3 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.

Linguaggi di DBMS (3)

- 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 concettuali
 - per la definizione di viste
 - per la manipolazione dei DB e
 - per 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

I DML (2)

- 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 singolo statement DML (es. l'SQL):
 - Sono detti dichiarativi perché una query di un DML high-level spesso specifica quale dato deve essere ritrovato piuttosto che come ritrovarlo.

I DML (3)

- Quando il linguaggio DML (high-level o low-level) è inglobato in un linguaggio general-purpose (l'host language) esso è detto data **sublanguage**.
- Se un linguaggio DML high-level è usato da solo in maniera interattiva, esso è detto query language. Un query language high-level 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 DI 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 menu che viene visualizzata dal sistema
 - Non c'è necessità di memorizzare i comandi specifici e la sintassi di un query language
 - I menu pull-down sono spesso usati nelle interfacce browsing, che consentono all'utente di guardare i contenuti del DB in maniera non strutturata

INTERFACCE DI DBMS (2)

■ Interfacce Forms-based

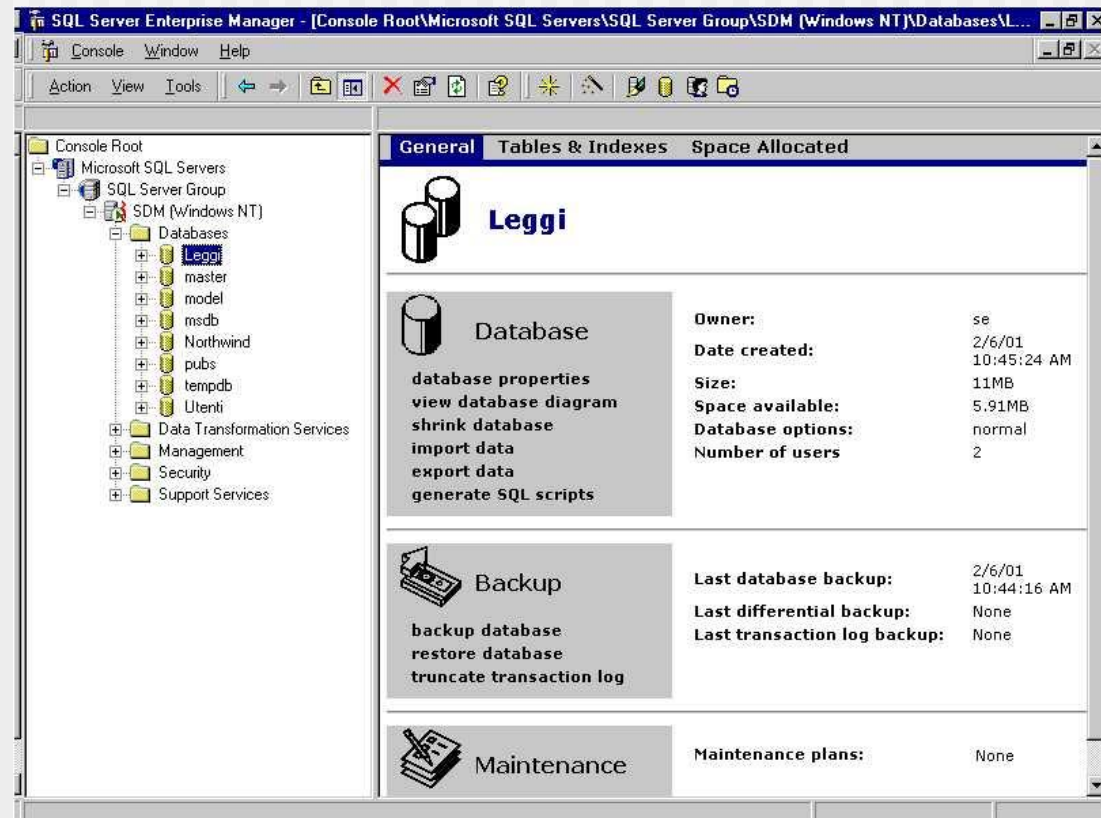
Un'interfaccia forms-based visualizza un form per ciascun utente. Gli utenti possono

- Riempire tutte le entrate del form completamente per inserire nuovi dati
- 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 DI DBMS (3)

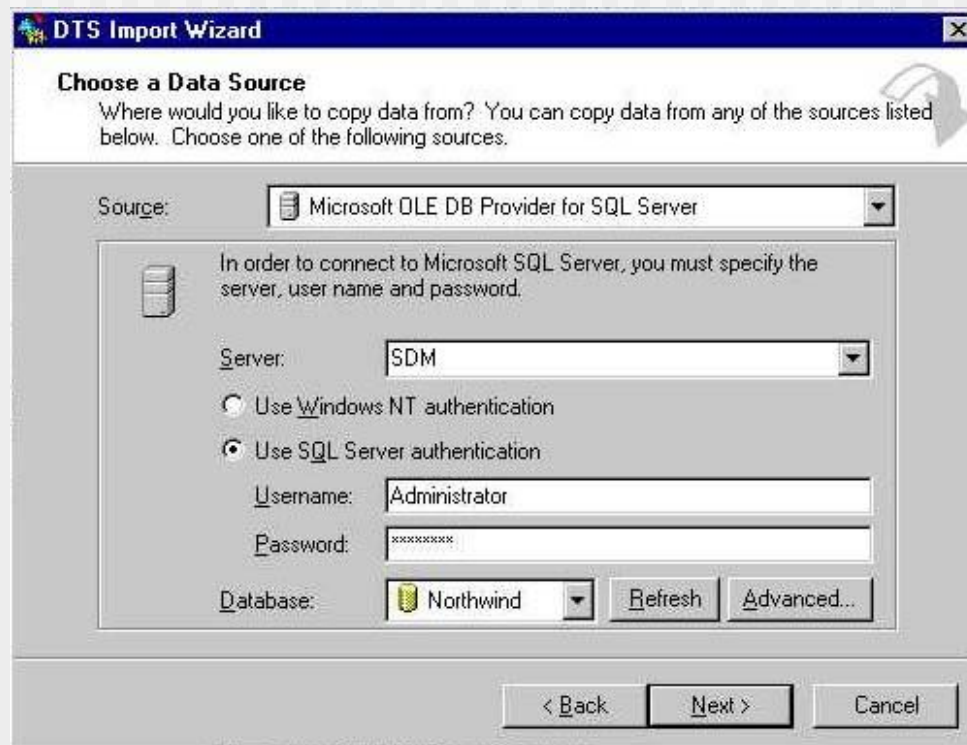
■ Interfacce per DBA

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



Utility di un DBMS

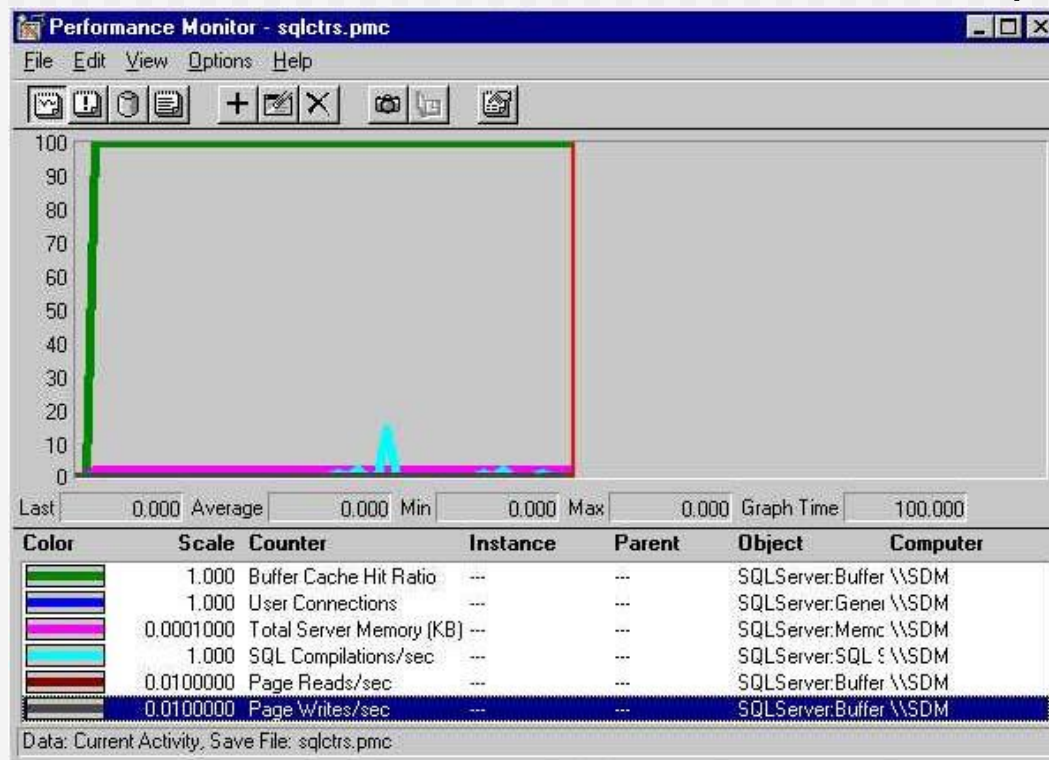
- Loading utilities
 - Per la conversione di dati da formati esterni al formato di specifico del DBMS



Microsoft SQL Server Import Wizard

Utility di un DBMS (2)

- Performance Monitor utilities
 - Per monitorare il db e costruire statistiche per il DBA



Utility di un DBMS (3)

- 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

