

BASI DI DATI I

- Modelli dei Dati
- Architettura dei R-DBMS
- Indipendenza logica e fisica dei dati
- Dati e Metadati
- Architettura Client Server

Prof. Adriano Peron

Prof. Silvio Barra

MODelli DEI DATI



MODELLI ED ASTRAZIONE DATI

- L'approccio database ha la caratteristica fondamentale di fornire una forte astrazione dei dati, nascondendo tutti i dettagli di memorizzazione non necessari agli utenti.
- L'astrazione dei dati si ottiene per mezzo di un data model.
- Un insieme di concetti che possono essere usati per descrivere la struttura di una base di dati.



MODELLI DEI DATI

- Un data model è un insieme di concetti per descrivere
 - La **struttura** di un database
 - *Es:* costrutti per definire gli elementi ed il loro tipo, entità, record e relazioni
 - Le **operazioni** per manipolare le strutture
 - *Es:* Inserzione, cancellazione, modifica, ritrovamento di un oggetto.
 - *Es:* Un'operazione COMPUTE_GPA che calcola la media dei voti e che può essere applicata a un oggetto studente.
 - Alcuni **vincoli** cui il database deve sottostare
 - I vincoli specificano restrizioni sui dati per renderli validi



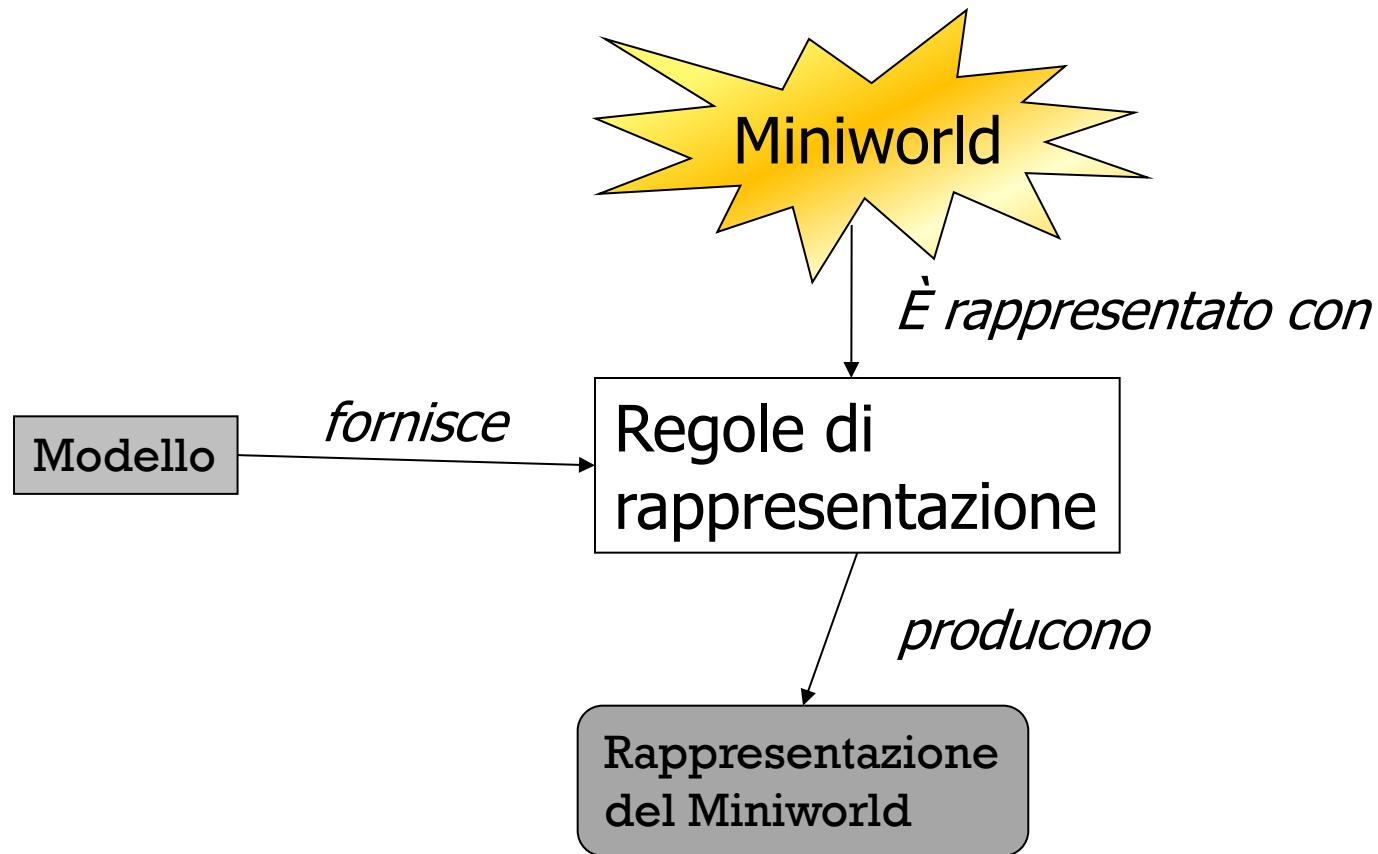
COSA DEVE FARE UN MODELLO

Un modello deve:

- **Rappresentare una certa realtà:**
 - **Es:** Una mappa rappresenta una porzione di territorio.
È quindi un modello del territorio, che rappresenta alcune caratteristiche, nascondendone altre.
- **Fornire un insieme di strutture simboliche per descrivere la rappresentazione della realtà:**
 - **Es:** Una mappa ha una serie di simboli grafici per rappresentare delle entità reali.

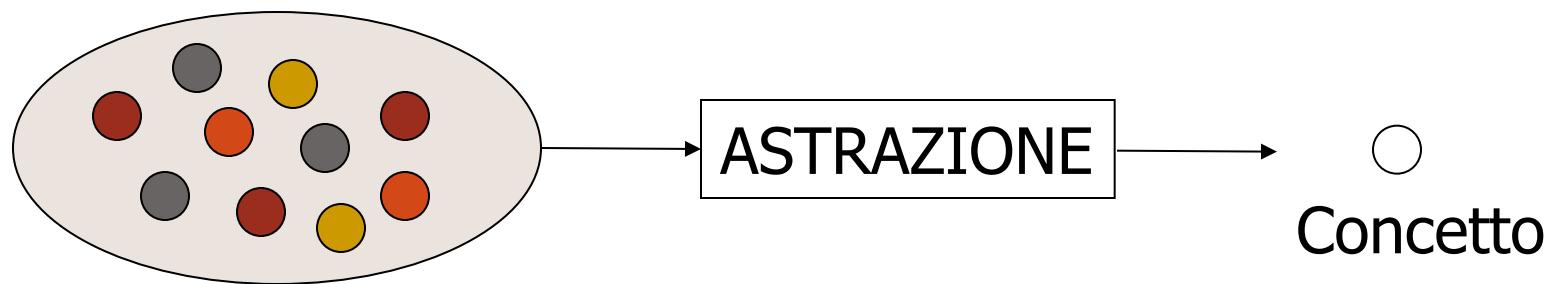


COS'È UN MODELLO (SCHEMA)



ASTRAZIONE

- L'astrazione è un procedimento mentale che sostituisce con un concetto un insieme di oggetti in base ad alcune loro proprietà:



Esempio:

l'astrazione consente di definire il concetto di “automobile”. Grazie ad esso è possibile descrivere e riconoscere tutte le automobili della realtà.



ASTRAZIONE E MODELLI

- L'astrazione è importante perché permette di comunicare ed elaborare una rappresentazione del miniworld.
- Per comunicare ed elaborare tale rappresentazione si utilizzano dei modelli concettuali e logici.
- Un **modello concettuale** fornisce i simbolismi per rappresentare concetti astratti in modo indipendente da qualsiasi elaboratore.
- Un **modello logico** traduce le strutture concettuali in strutture logiche processabili da un DBMS.



CATEGORIE DI DATA MODEL

È possibile categorizzare i vari data model proposti secondo i concetti utilizzati per descrivere la struttura del database.

- I data model di alto livello o concettuali forniscono concetti che sono vicini al modo di percepire i dati degli utenti.
 - Anche noti come **entity-based** o **object-based** data models
- I data model di basso livello o fisici forniscono concetti che descrivono dettagli su come i dati sono memorizzati.
- I data model rappresentazionali o di implementazione forniscono concetti comprensibili agli utenti finali, ma che non sono troppo lontani dal modo in cui i dati sono fisicamente organizzati.
 - Essi nascondono alcuni dettagli della memorizzazione dei dati ma possono essere implementati in maniera diretta.
- I data model auto-descrittivi combinano la descrizione dei dati con i valori dei dati stessi
 - Ad esempio, come si fa con l'XML



DATA MODEL DI ALTO LIVELLO

- I data model di alto livello usano concetti quali entità, attributi e relazioni:
 - Un'entità rappresenta un oggetto o concetto del mondo reale.
 - **Es:** Un impiegato o un progetto.
 - Un attributo rappresenta qualche proprietà importante che descrive ulteriormente un'entità.
 - **Es:** Il nome o lo stipendio di un impiegato.
 - Una relazione tra due o più entità rappresenta un'interazione tra le entità.
 - **Es:** Una relazione “lavora su” tra un impiegato e un progetto.
- Il più comune data model di alto livello è il modello entità-relazione.

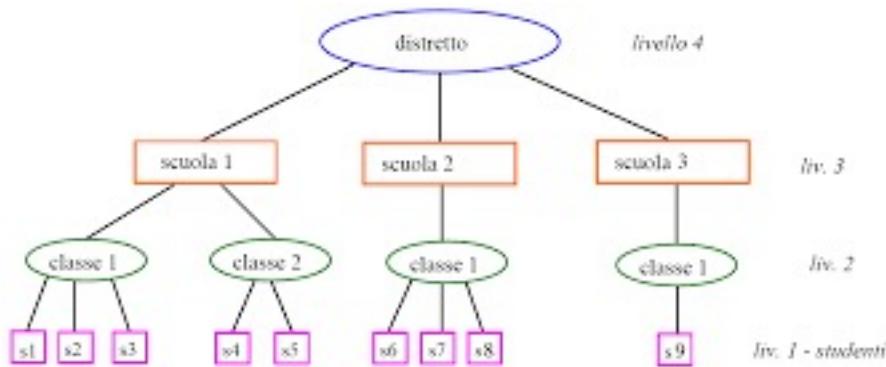


DATA MODEL RAPPRESENTAZIONALI

- I data model rappresentazionali, per la facilità con cui possono essere implementati, comprendono i tre data model più usati dai DBMS commerciali:
 - Il modello **relazionale** (i dati sono incapsulati in record a struttura fissa. La relazione è rappresentata da una tabella)
 - Il modello **reticolare** (si basa sull'utilizzo di grafi e si basano sui concetti di record e set. Utilizzato per limitare la ridondanza)
 - Il modello **gerarchico** (utilizza strutture ad albero con i record come nodi e le associazioni come archi. Ha problemi per quanto riguarda la ridondanza dei dati)
- Tali modelli rappresentano i dati usando strutture di record. Per tale motivo sono chiamati anche modelli record-based.
- I modelli **object-oriented**, grazie alla forte astrazione dei dati, costituiscono una nuova famiglia di data model, posti a metà strada tra il rappresentazionale e il concettuale.



ESEMPI DI DATA MODEL RAPPRESENTAZIONALI



gerarchico

reticolare

CLIENTI

Bianchi	Mazzini	Bergamo	24127
---------	---------	---------	-------

Neri	Garibaldi	Napoli	80120
------	-----------	--------	-------

Rossi	Rosmini	Milano	20125
-------	---------	--------	-------

Gialli	Cavour	Bari	70122
--------	--------	------	-------

Verdi	Dante	Roma	00128
-------	-------	------	-------

Rosa	Crispi	Torino	10137
------	--------	--------	-------

Moro	Colombo	Milano	20143
------	---------	--------	-------

CONTI

9000	120345
------	--------

7000	500
------	-----

4500	3500
------	------

5100	58500
------	-------

8000	6320
------	------

4310	37
------	----

MOVIMENTI

23/12/2009	Vers	2.500,00
------------	------	----------

20/04/2010	Prel	1.250,00
------------	------	----------

27/05/2010	Vers	550,00
------------	------	--------

..
----	----	----

21/08/2010	Prel	350,00
------------	------	--------

21/08/2010	Prel	536,00
------------	------	--------

Activity Code	Activity Name
23	Patching
24	Overlay
25	Crack Sealing

Activity Code	Date	Route No.
24	01/12/01	I-95
24	02/08/01	I-66

Date	Activity Code	Route No.
01/12/01	24	I-95
01/15/01	23	I-495
02/08/01	24	I-66

relazionale



DATA MODEL FISICI

- I data model fisici descrivono come sono memorizzati i dati nel calcolatore rappresentando informazioni quali formati di record, ordinamenti di record, percorsi di accesso.
 - Un percorso di accesso è una struttura che rende efficiente la ricerca di particolari record di database.



SCHEMI E ISTANZE DI DATABASE

- In qualsiasi data model è necessario distinguere tra la descrizione del database ed il database stesso.
- La differenza è simile a quella tra tipi e variabili nei linguaggi di programmazione.
- Lo schema è la struttura logica del database.
- Un'istanza è il contenuto del database in un particolare istante di tempo.



SCHEMI DI DATABASE

- La descrizione del database è detta schema (o metadati).
- Lo schema del database è specificato in fase di progetto e non ci si aspetta che cambi frequentemente.
 - Per rappresentare uno schema si crea un diagramma di schema, secondo opportune convenzioni definite dal data model.
 - Un diagramma di schema visualizza la struttura dei record ma non le reali istanze dei record.
- Il DBMS memorizza lo schema nel catalogo per poterne fare riferimento ogni qualvolta gli occorre.



SCHEMI DI DATABASE:

ESEMPIO DB UNIVERSITÀ

STUDENTE		
NOME	MATRICOLA	ANNO
CORSO		
DENOMIN.	SEMESTRE	TITOLARE
PREREQUISITI		
DENOMIN.	PROPEDEUTICITA'	
VOTAZIONE		
NOME	DENOMIN.	VOTO

- Ogni oggetto nello schema è detto costrutto di schema.
 - *Es:* STUDENTE o CORSO sono ciascuno un costrutto di schema.



ISTANZE DI DATABASE

- I dati reali in un db possono cambiare frequentemente.
 - *Es:* Il db UNIVERSITA' cambia ogni volta che si inserisce un nuovo studente o si registra un nuovo esame per uno studente.
- Uno stato del database (detto anche istanza o insieme di occorrenze del db) è l'insieme dei dati presenti nel database in un particolare istante di tempo.



ISTANZE DI DATABASE (2)

- Dato un stato del database, ogni costrutto di schema ha un proprio insieme corrente di istanze
 - *Es:* Il costrutto STUDENTE conterrà l'insieme di entità (o record) studenti individuali come sue istanze.
- Per un dato schema di database possono essere costruiti diversi stati di database.
- Ogni volta che si inserisce o cancella un record o si modifica il valore di un data item, si cambia lo stato del database.



L'IMPORTANZA DELLO SCHEMA DI DATABASE

- Quando si crea un nuovo database, si specifica al DBMS lo schema del database:
 - in tale fase il db è nello “*stato vuoto*”, senza dati. Si passa nello “*stato iniziale*”, quando si inseriscono i dati per la prima volta.
- Il DBMS è parzialmente responsabile nel garantire che ogni stato del DB sia valido, cioè che soddisfi la struttura ed i vincoli specificati nello schema.
- Quindi:
 - specificare uno schema corretto è estremamente importante; lo schema deve essere progettato con la massima cura.



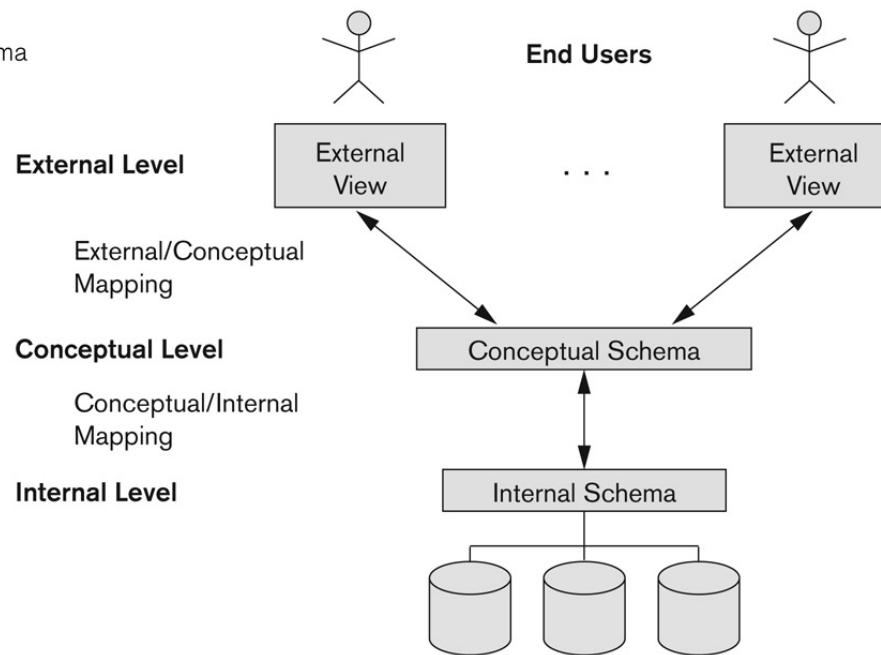
ARCHITETTURA DEI R-DBMS



L'ARCHITETTURA THREE-SCHEMA (2)

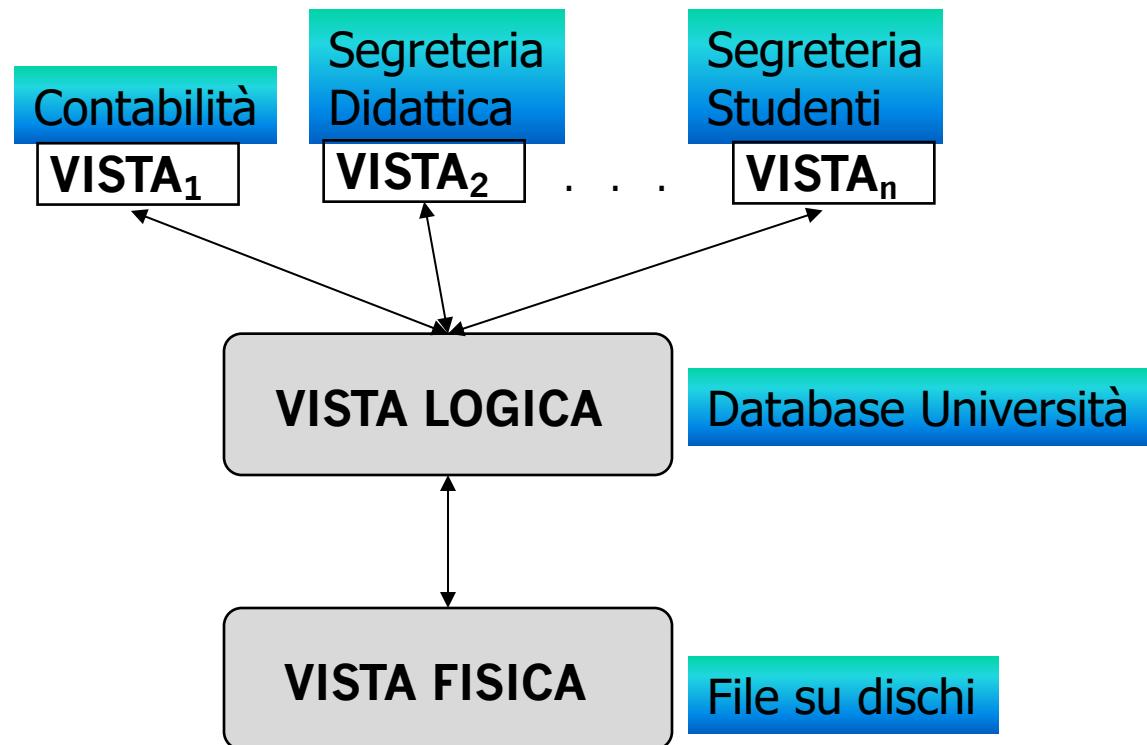
Figure 2.2

The three-schema architecture.



- Ricordiamo le caratteristiche principali dell'approccio di database:
 - supporto di viste multiple per gli utenti;
 - uso di un catalogo per memorizzare la descrizione del DB (lo schema);
 - condivisione dei dati ed elaborazione delle transazioni in modalità multiutente.
- L'architettura per database system chiamata **three-schema** aiuta ad ottenere tali caratteristiche, definendo il database in tre livelli e separando quindi le applicazioni utente dal database fisico.

L'ARCHITETTURA THREE-SCHEMA: UN'ESEMPIO



Gli schemi possono essere specificati a tre livelli:

- **Interno (Internal Schema)**
 - Per descrivere le strutture di storing fisiche (*physical data model*)
- **Concettuale (Conceptual Schema)**
 - Descrive la struttura e i vincoli del database
- **Esterno (External Schema)**
 - Descrive le varie viste per gli utenti



LIVELLO INTERNO

Livello interno come schema interno:

- Descrive la struttura di memorizzazione fisica del DB.
- Lo schema interno usa un data model fisico e descrive i dettagli completi del data storage e gli access paths del DB.
- **Esempio:**
Dividi i record del file studenti in tre partizioni sui dischi 5, 6 e 7.



LIVELLO CONCETTUALE

Livello concettuale come schema concettuale:

- Descrive la struttura del DB per una comunità di utenti.
- Lo schema concettuale nasconde i dettagli delle strutture di storage fisico e si concentra sulla descrizione delle entità, tipi di dati, relazioni, operazioni utente e vincoli.
- Può usare un data model ad alto livello o uno di implementazione.
- **Esempio:**
type Studente= **record**
 nome : **string**;
 matricola : **string**;
 anno : **string**;
end;



LIVELLO ESTERNO

Livello esterno come schema esterno:

- Definisce un sottoinsieme del DB per una particolare applicazione.
 - Include più schemi esterni o viste utente.
 - Usa un data model di alto livello o di implementazione.
 - Ogni schema esterno descrive la parte del DB cui è interessato un particolare utente 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.



I MAPPING TRA I LIVELLI DELL'ARCHITETTURA

- I tre schemi sono solo delle descrizioni di dati: gli unici dati che realmente esistono sono a livello fisico.
- In un DBMS basato sull'architettura three-schema, ogni gruppo di utenti utilizza una propria vista esterna.
- 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 concettuale e poi...
 - ...in una richiesta sullo schema interno per procedere sul database memorizzato.



INDIPENDENZA LOGICA E FISICA DEI DATI



INDIPENDENZA DEI DATI

L'architettura three-schema 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 concettuale 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 concettuali (o esterni).



INDIPENDENZA LOGICA

- Indica la capacità di cambiare lo schema concettuale senza dover cambiare lo schema esterno e gli applicativi correlati
- Lo schema concettuale può essere cambiato per espandere oppure per 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 solo ai dati restanti non devono essere alterati.
 - Se uno schema ad un livello più basso viene modificato, soltanto il **mapping tra questo schema e quelli di livello più alto necessitano di essere cambiati**;
 - i programmi applicativi che fanno riferimento agli schemi esterni sono indifferenti alle modifiche, siccome si riferiscono solo agli schemi esterni.



INDIPENDENZA LOGICA: ESEMPIO

Lo schema esterno

ESAMI SUPERATI	Nome	Matricola	ESAMI	
			DENOMIN.	VOTO
	Rossi	N86000484	Basi di Dati	24
			Object Orientation	28
	Verdi	N86000323	Algoritmi	30
			Sistemi Biometrici	27

non dovrebbe essere alterato cambiando il file votazione da

VOTAZIONE		
NOME	DENOMIN.	VOTO
Rossi	Basi di Dati	24
Rossi	Object Orientation	28
Verdi	Algoritmi	30
...

a

VOTAZIONE			
NOME	MATRICOLA	DENOMIN.	VOTO
Rossi	N86000484	Basi di Dati	24
Rossi	N86000484	Object Orientation	28
Verdi	N86000323	Algoritmi	30
...



INDIPENDENZA FISICA

- Indica la capacità di cambiare lo schema interno senza dover cambiare lo schema concettuale
- Un cambiamento dello schema interno può essere dovuto alla riorganizzazione di qualche file fisico (**es:** creando ulteriori strutture di accesso, o modificando degli indici), per migliorare l'esecuzione del ritrovamento o dell'aggiornamento.
 - Se i dati non subiscono alterazioni, non è necessario cambiare lo schema concettuale.
- 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 **Denominaz.** e **Semestre** non dovrebbe richiedere variazioni a una query come

“ritrova tutti i corsi tenuti nel secondo semestre”

sebbene la query possa essere eseguita in maniera più efficiente.



L'ARCHITETTURA THREE-SCHEMA: VANTAGGI E SVANTAGGI

VANTAGGI

- L'architettura three-schema consente facilmente di ottenere una reale indipendenza dei dati.
- Il database risulta più flessibile e scalabile.

SVANTAGGI

- Il catalogo del DBMS multilivello 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.



DBMS LANGUAGES



LINGUAGGI DBMS

- *Un DBMS deve fornire ad ogni categoria di utenti delle interfacce e dei linguaggi adeguati.*
- Alla fase di progetto del DB segue quella di specifica degli schemi concettuale ed interno e di qualsiasi mapping tra i due.
- Nei DBMS dove non c'è una netta separazione tra livelli, progettisti e DBA usano un **data definition language (DDL)** per definire entrambi gli schemi.
- Il compilatore del DDL (contenuto nel DBMS) ha la funzione di:
 - Trattare gli statement DDL per identificare le descrizioni dei costrutti dello schema.
 - Memorizzare la descrizione dello schema nel catalogo del DBMS.



LINGUAGGI DBMS (2)

- Dove c'è una chiara distinzione tra livello concettuale ed interno si usa:
 - uno **storage definition language (SDL)** per specificare lo schema interno,
 - il **DDL** per specificare soltanto lo schema concettuale.
- I mapping tra i due schemi possono essere specificati in uno qualsiasi dei due linguaggi.
- Nelle architetture three-schema viene usato un **view definition language (VDL)** per specificare le viste utente e i loro mapping sullo schema concettuale.



LINGUAGGI DBMS (3)

- Una volta che gli schemi sono compilati ed il DB è riempito di dati:
 - il DBMS fornisce un **data manipulation language (DML)** per consentire agli utenti di manipolare (cioè recuperare, inserire, cancellare e aggiornare) dati.
- 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 della memorizzazione.
- ***Esempio:***
 - Il linguaggio di database relazionali **structured query language (SQL)** rappresenta una combinazione di DDL, VDL, DML e SDL.



I DML DI ALTO LIVELLO

- Consentono da soli di specificare operazioni di database complesse in maniera concisa.
- In molti DBMS gli statement di DML di alto livello
 - Possono essere specificati interattivamente da terminale.
 - Possono essere inglobati (*embedded*) in un linguaggio di programmazione general-purpose (in questo caso gli statement DML sono identificati all'interno del programma in modo che il DBMS possa trattarli (*precompilazione*)).
 - 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 DI BASSO LIVELLO

- Deve essere inglobato in un linguaggio general-purpose;
- È detto **record-at-a-time** perché tipicamente ritrova record individuali nei DB e tratta ciascun record separatamente;
 - ha quindi bisogno di usare costrutti di programmazione come l'iterazione per ritrovare e trattare ogni record da un insieme di record.



SQL, UN LINGUAGGIO INTERATTIVO

- “*Trovare i corsi tenuti in aule a piano terra*”

Corsi

Corso	Docente	Aula
Basi di dati	Rossi	DS3
Sistemi	Neri	N3
Reti	Bruni	N3
Controlli	Bruni	G

Aule

Nome	Edificio	Piano
DS1	OMI	Terra
N3	OMI	Terra
G	Pincherle	Primo



SQL, UN LINGUAGGIO INTERATTIVO (2)

SELECT Corso, Aula, Piano

FROM Aule, Corsi

WHERE Nome = Aula **AND** Piano = “*Terra*”

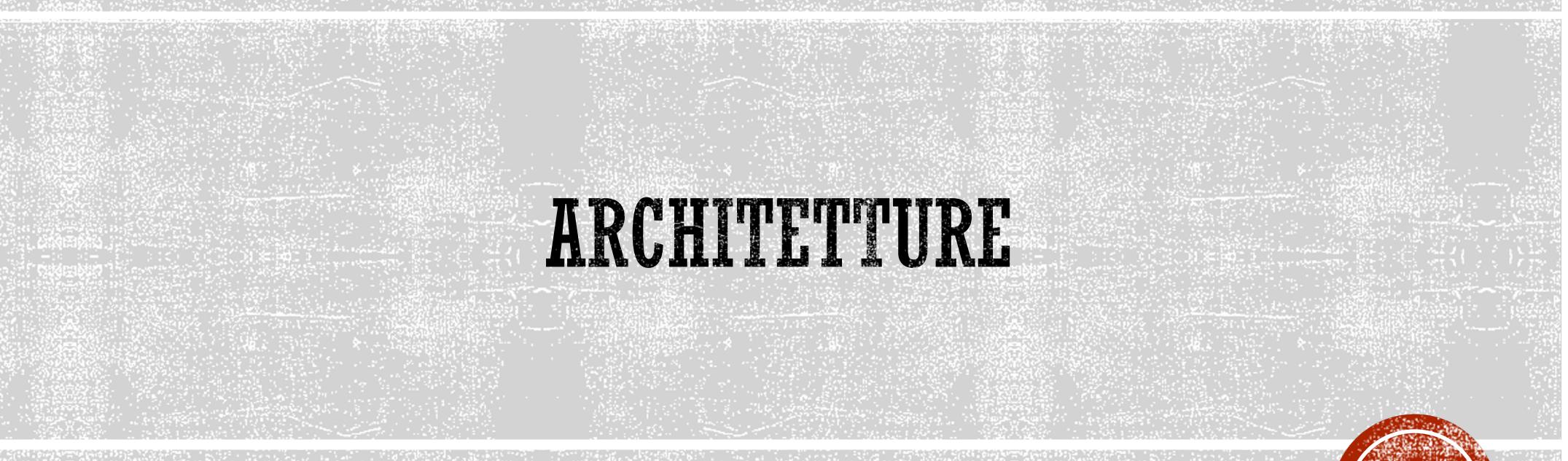
Corso	Aula	Piano
Sistemi	N3	Terra
Reti	N3	Terra



SQL IMMERSO (EMBEDDED) IN LINGUAGGIO OSPITE

```
write('nome della citta?'); readln(citta);
EXEC SQL DECLARE P CURSOR FOR
  SELECT NOME, REDDITO
  FROM PERSONE
  WHERE CITTA = :citta ;
EXEC SQL OPEN P ;
EXEC SQL FETCH P INTO :nome, :reddito ;
while SQLCODE = 0 do
  begin
    write('nome della persona:', nome, 'aumento?'); readln(aumento);
    EXEC SQL UPDATE PERSONE
      SET REDDITO = REDDITO + :aumento
      WHERE CURRENT OF P ;
    EXEC SQL FETCH P INTO :nome, :reddito ;
  end;
EXEC SQL CLOSE CURSOR P ;
```



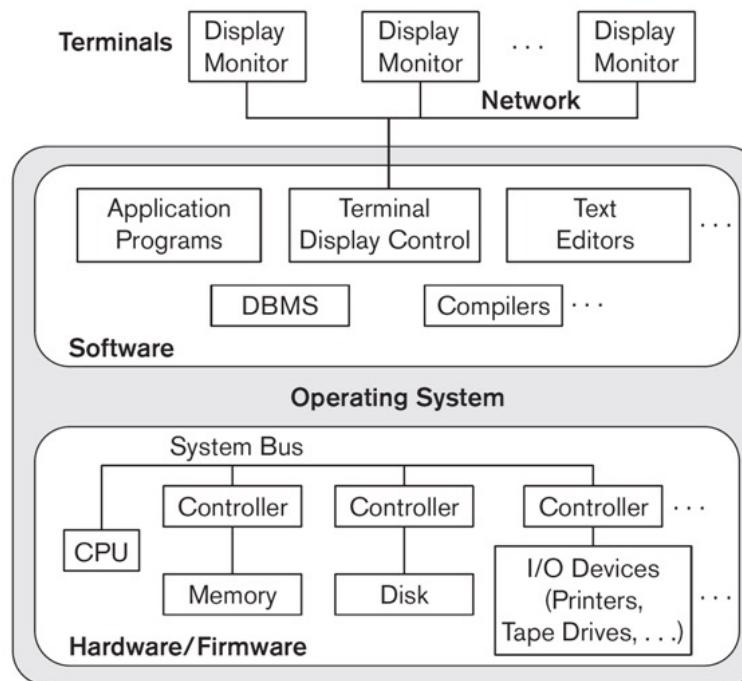


ARCHITETTURE



ARCHITETTURA DEL DBMS CENTRALIZZATO

- Tutte le funzionalità del DBMS, l'esecuzione del programma applicativo, e l'elaborazione dell'interfaccia utente sono eseguite su una sola macchina.



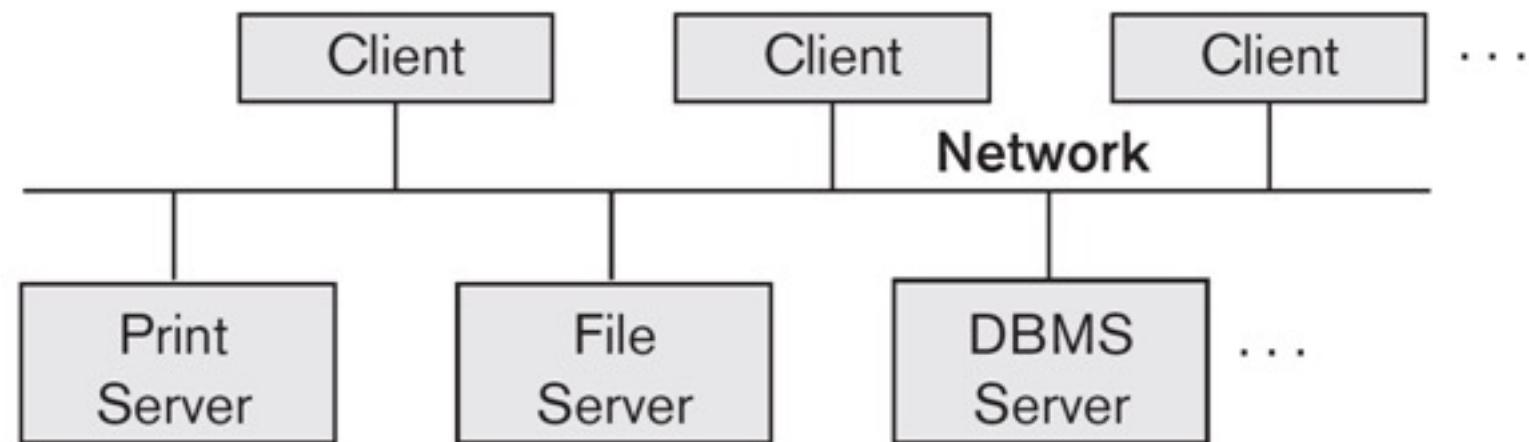
ARCHITETTURA CLIENT/SERVER BASE

- Sono **Server** con specifiche funzionalità:
 - **File server**
 - Mantiene e gestisce i file delle macchine client.
 - **Printer server**
 - È connesso a diverse stampanti,
 - Tutte le richieste di stampa dalle macchine client sono trasmesse a questa macchina.
 - **Web server o e-mail server**
- **Le macchine Client:**
 - Forniscono l'utente di un interfaccia appropriata per utilizzare i server.
 - Hanno capacità operazionali locali per eseguire applicazioni in locale.



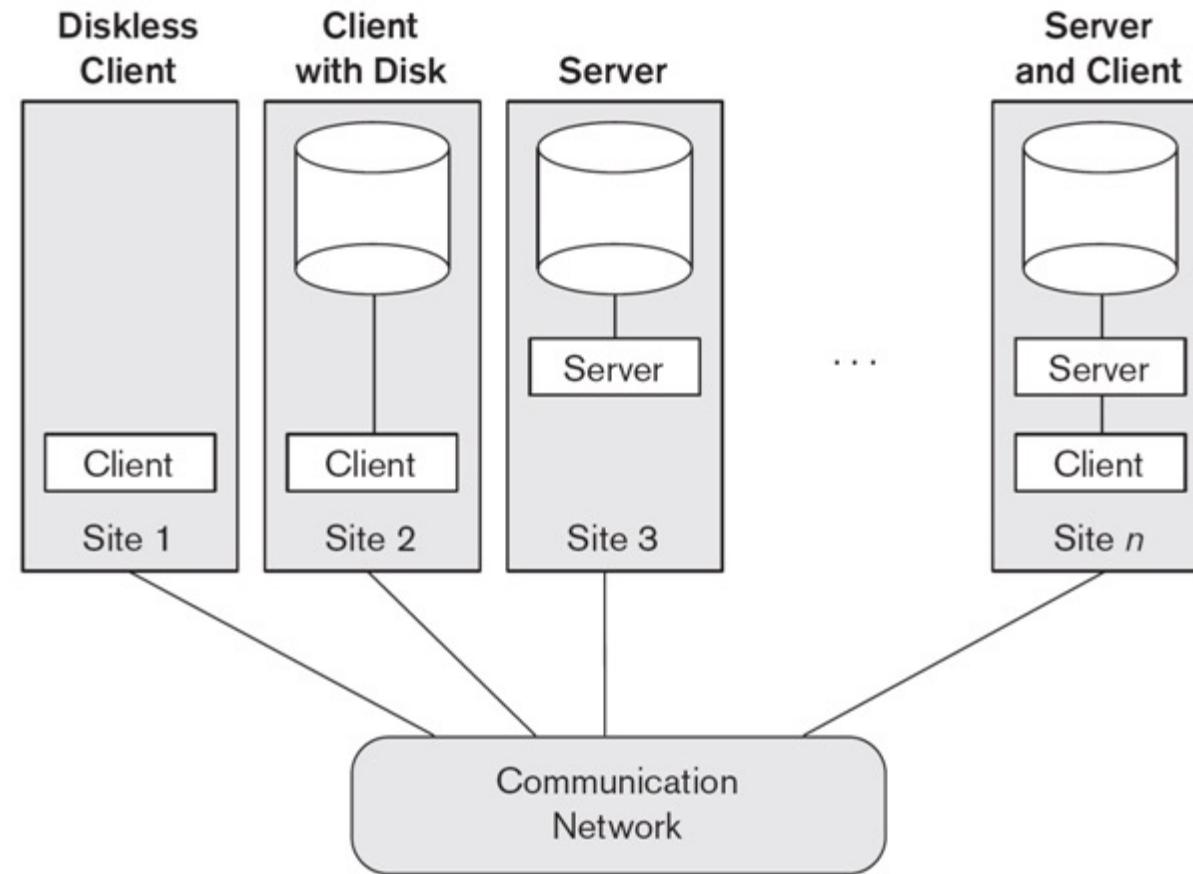
ARCHITETTURA CLIENT/SERVER TWO-TIER

- Architettura logica:



ARCHITETTURA CLIENT/SERVER TWO-TIER (2)

- Architettura fisica:



ARCHITETTURA CLIENT/SERVER TWO-TIER (3)

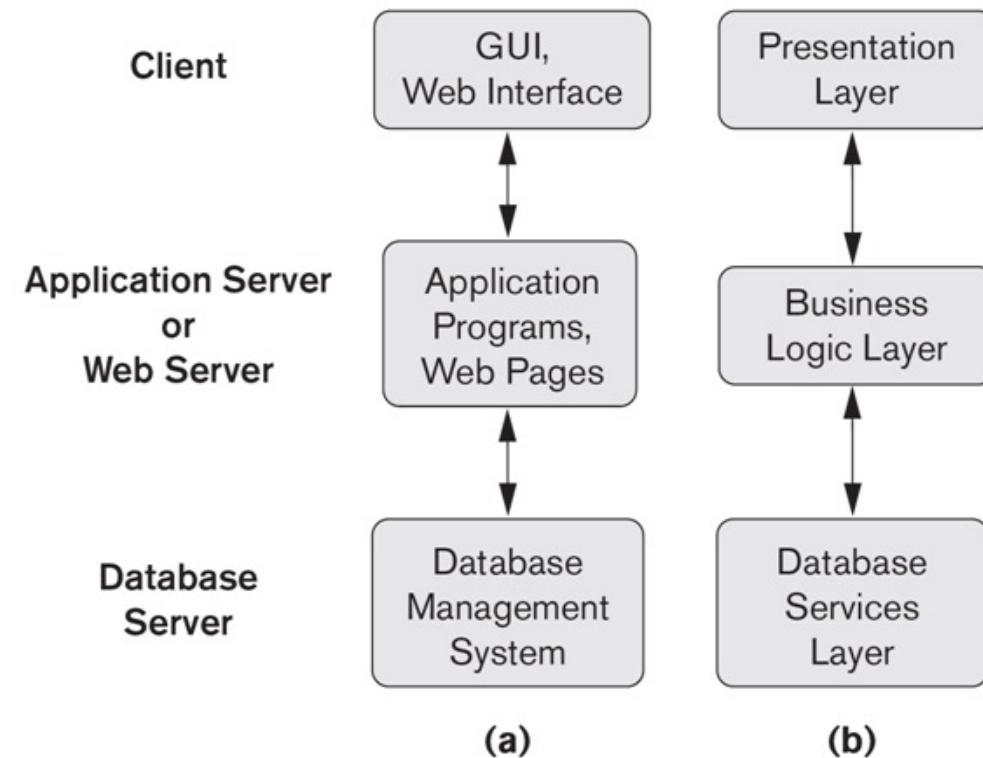
- Open Database Connectivity (ODBC)
 - Fornisce le Application Programming Interface (API).
 - Permette ad un programma in esecuzione sulla macchina client di comunicare con il DBMS
 - Entrambi le macchine client e server devono avere installato il software necessario.
- Java DataBase Connectivity (JDBC)
 - Permette ai programmi client scritti in **Java** di accedere ad uno o più DBMS attraverso un interfaccia standard.



ARCHITETTURA CLIENT/SERVER THREE-TIER

- **Application server o Web server**

- Aggiunge uno strato intermedio tra il client ed il database server.
- Esegue i programmi applicativi e memorizza le regole di business.



(a)

(b)



FINE

Per eventuali domande: (in ordine di preferenza personale)

- Ora.
- Chat di Teams
- Mail: silvio.barra@unina.it

