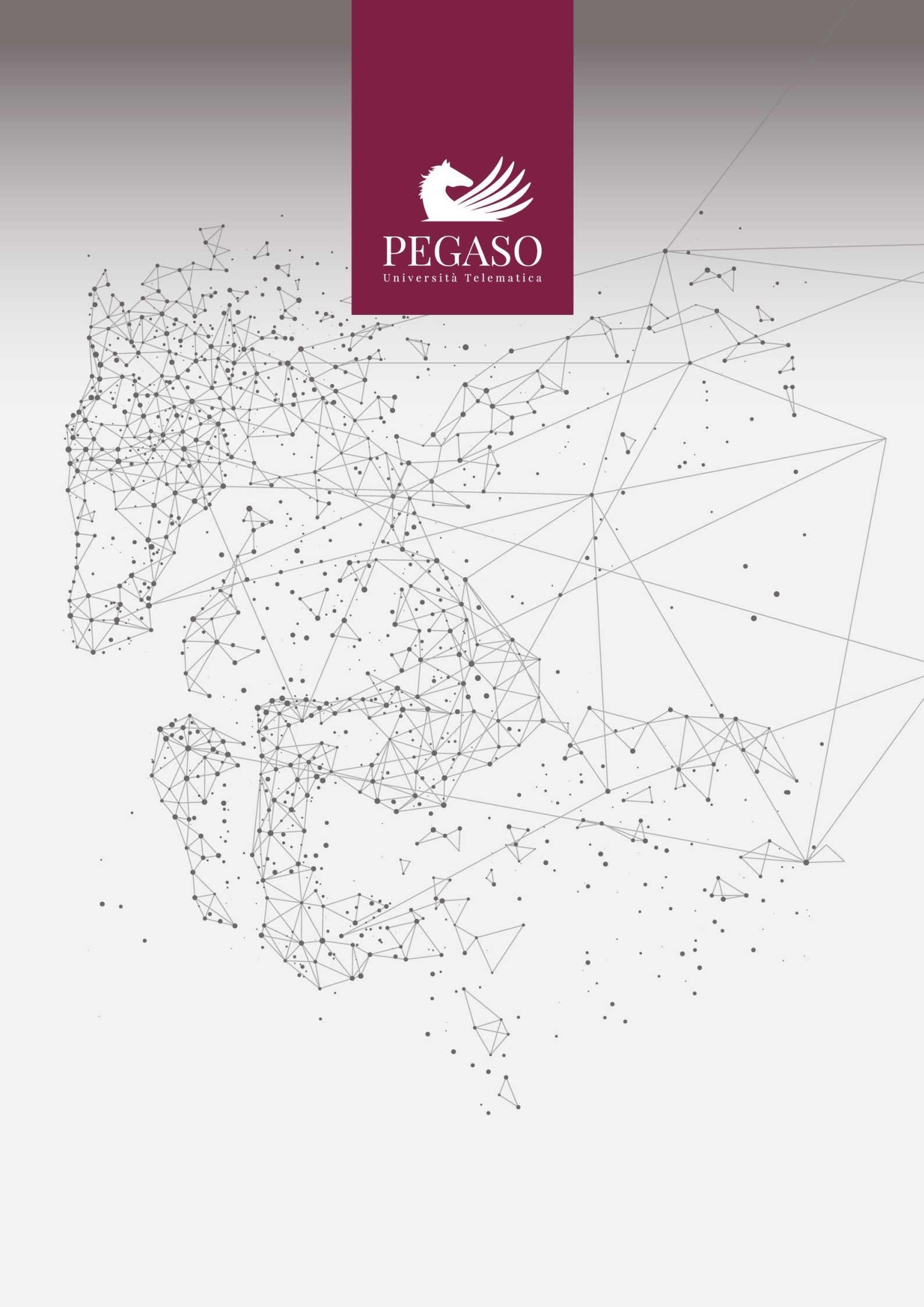
PEGASO
Università Telematica

Indice

1. Introduzione

In questa unità didattica vengono introdotti i principali modelli di dati, compreso il modello Entity-Relationships che verrà utilizzato durante il corso, le architetture dei moderni DBMS ed infine vengono discussi due argomenti fondamentali: l'indipendenza logica e fisica dei dati.

2. I modelli

Un modello di dati può essere definito come un modello astratto che condivide gli elementi dei dati e standardizza il modo in cui si relazionano tra loro. Ad esempio, un modello di dati può specificare il colore e le dimensioni dell'auto definendone anche il proprietario. Il termine *modello di dati* viene utilizzato in due sensi distinti ma strettamente correlati. A volte si riferisce ad una formalizzazione astratta degli oggetti e delle relazioni trovate in un'organizzazione manifatturiera. Altre volte fa riferimento a un insieme di concetti utilizzati nella definizione di tali formalizzazioni: ad esempio concetti come entità, attributi, relazioni o tabelle. Quindi il "modello dati" di un'applicazione bancaria può essere definito usando il "modello dati" relazione-entità. Un modello di dati determina esplicitamente la struttura dei dati: i modelli di dati sono specificati in una notazione di modellazione dei dati, che è spesso in forma grafica (McCaleb,1999). Gestire grandi quantità di dati strutturati e non strutturati è una funzione primaria dei sistemi di informazione. I modelli di dati descrivono la struttura, la manipolazione e l'integrità dei dati memorizzati nei sistemi di gestione dei dati come i database relazionali. In genere non descrivono dati non strutturati, come documenti di elaborazione testi, messaggi e-mail, immagini, audio digitale e video. Lo scopo principale dei modelli di dati quindi, è supportare lo sviluppo di sistemi informativi, fornendo la definizione e il formato dei dati. Secondo West e Fowler (Fowler,1999) "se questo viene fatto in modo coerente tra i sistemi, è possibile ottenere la compatibilità dei dati: se le stesse strutture di dati vengono utilizzate per archiviare e accedere ai dati, diverse applicazioni possono condividere i dati. Tuttavia, i sistemi e le interfacce spesso costano più di quanto dovrebbero, per costruire, gestire e mantenere, ma possono anche limitare l'azienda piuttosto che supportarla. Una delle cause principali è la scarsa qualità dei modelli di dati implementati nei sistemi e nelle interfacce ".

2.1 Modello logico

Il modello logico o schema logico è un modello di dati di un dominio specifico espresso indipendentemente da un particolare prodotto di gestione del database o tecnologia di archiviazione (modello di dati fisici) ma in termini di strutture di dati come tabelle relazionali e colonne, classi orientate agli oggetti o Tag XML. Il modello logico dei dati è adottato nei DBMS esistenti per l'organizzazione dei dati.

Sono utilizzati dai programmi ed indipendenti dalle strutture fisiche. Tra gli esempi di modelli ligici abbiamo:

- Modello relazionale
- Modello reticolare
- Modello gerarchico
- Modello a oggetti
- Modello basato su XM

2.2 Modello concettuale

Il modello concettuale descrive la semantica di un'organizzazione senza riferimento alla tecnologia sottostante. Tale astrazione permette di rappresentare i dati in modo indipendente da ogni sistema. Il modello concettuale cerca di descrivere i concetti del mondo reale e viene utilizzato nelle fasi preliminari di progettazione di una base di dati. Il più diffuso è il modello *Entity-Relationship.*In pratica, il modello Entità-Relazione (ER), è una rappresentazione grafica delle entità e delle loro relazioni reciproche, tipicamente utilizzate nel calcolo in relazione all'organizzazione di dati all'interno di database o sistemi di informazione. Di tale modello ce ne occuperemo in dettaglio nelle prossime lezioni. In Figura 1 è illustrato un semplice esempio di diagramma E-R (tratto da beginnersbook.com).

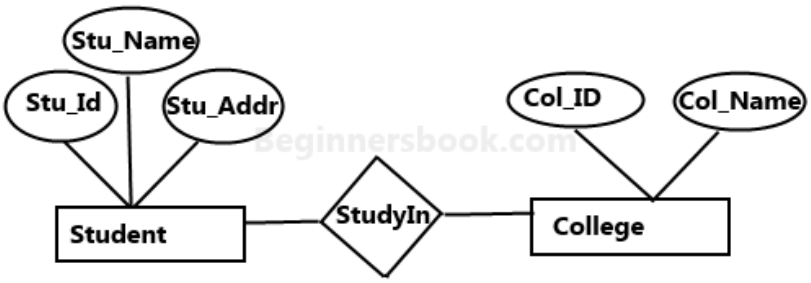

Figura 1: *Esempio di modello concettuale.*

3. Architetture di un DBMS

Il design di un DBMS dipende dalla sua architettura. Può essere centralizzata o decentralizzata o gerarchica. L'architettura di un DBMS può essere vista come singola o multilivello. Un'architettura a più livelli divide l'intero sistema in n moduli correlati ma indipendenti, che possono essere modificati, o sostituiti in modo indipendente.

Nell'architettura a **1 livello**, il DBMS è l'unica entità in cui l'utente si trova direttamente sul DBMS e lo utilizza. Qualsiasi cambiamento fatto qui sarà fatto direttamente sul DBMS stesso. Non fornisce strumenti pratici per gli utenti finali. Normalmente i progettisti di database e i programmatori preferiscono utilizzare l'architettura a livello singolo.

- Se l'architettura di DBMS è a **2 livelli**, si deve disporre di un'applicazione attraverso la quale risulti possibile accedere al DBMS. I programmatori utilizzano un'architettura a due livelli in cui accedono al DBMS tramite un'applicazione. Qui il livello dell'applicazione è completamente indipendente dal database in termini di funzionamento, progettazione e programmazione. I due livelli sono quelli interno e quello logico, come illustrato in Figura 2. Lo schema logico rappresenta la descrizione della base di dati nel modello logico (ad esempio, la struttura della tabella) mentre lo schema interno (o fisico) rappresenta lo schema logico per mezzo di strutture memorizzazione (file; ad esempio, record con puntatori, ordinati in un certo modo).

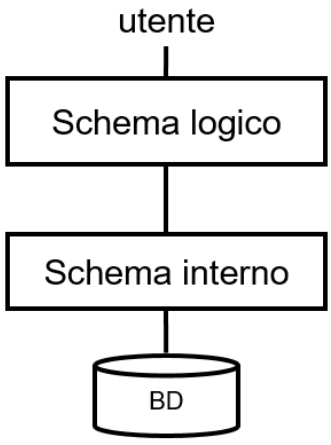

Figura 2: *Architettura semplificata a 2 livelli.*

3.1 Architettura a 3 livelli

Un'architettura a 3 livelli separa i livelli l'uno dall'altro in base alla complessità degli utenti e al modo in cui utilizzano i dati presenti nel database. È l'architettura più utilizzata per progettare un DBMS.

- **Livello database** (interno): a questo livello risiede il database insieme ai linguaggi di elaborazione delle query. Abbiamo anche le relazioni che definiscono i dati e i loro vincoli a questo livello;
- **Livello applicazione** (logico): a questo livello risiedono il server applicativi e i programmi che accedono al database. Ad un utente, questo livello applicazione presenta una vista astratta del database. Gli utenti finali non sono a conoscenza dell'esistenza del database oltre l'applicazione. All'altro estremo, il livello del database non è a conoscenza di nessun altro utente oltre il livello dell'applicazione. Quindi, il livello dell'applicazione si trova nel mezzo e funge da mediatore tra l'utente finale e il database;
- **Livello utente** (esterno): gli utenti finali operano su questo livello e non conoscono alcuna esistenza del database oltre questo livello. A questo livello, l'applicazione può fornire più viste del database. Tutte le viste sono generate da applicazioni che risiedono nel livello applicazione.

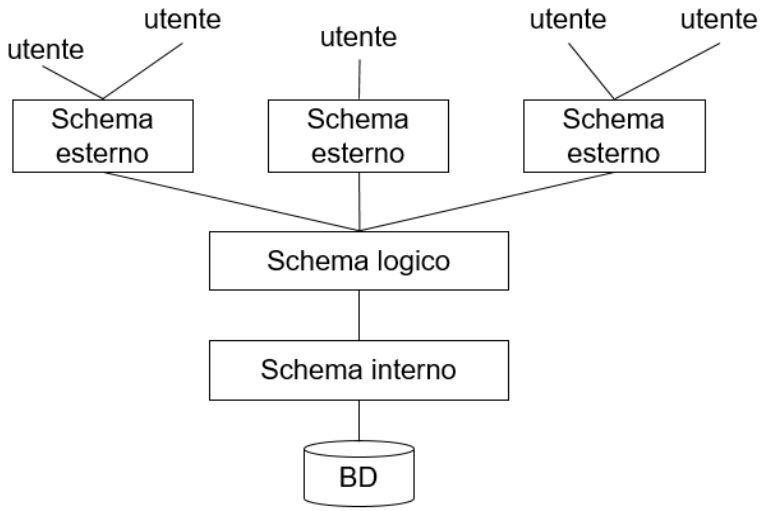

Figura 3: *schema a 3 livelli di un dbms.*

L'architettura di database a più livelli è altamente modificabile poiché quasi tutti i suoi componenti sono indipendenti e possono essere modificati indipendentemente, come illustrato in Figura 3.

L'architettura a 3 livelli consente visualizzazioni utente personalizzate e indipendenti: ogni utente è in grado di accedere agli stessi dati, ma con una diversa visualizzazione personalizzata dei dati. Questi

dovrebbero essere indipendenti: le modifiche a una vista non dovrebbero influire sugli altri. Nasconde i dettagli della memoria fisica dagli utenti: gli utenti non devono avere a che fare con i dettagli fisici della memoria del database. L'amministratore del database dovrebbe essere in grado di modificare le strutture di archiviazione del database senza influire sulle visualizzazioni degli utenti. La struttura interna del database non deve essere modificata dalle modifiche agli aspetti fisici dello spazio di archiviazione: ad esempio, un passaggio a un nuovo disco.

3.2 Architettura ANSI/SPARC

L'architettura ANSI-SPARC, dove ANSI-SPARC sta per *American National Standards Institute, Standards Planning and Requirements Committee*, è uno standard di disegno astratto per un Database Management System (DBMS), proposto per la prima volta nel 1975. Il modello ANSI-SPARC tuttavia non è mai diventato uno standard formale. Nessun sistema di DBMS si basa interamente su di esso (tendono a non mostrare piena indipendenza fisica o a impedire l'accesso diretto dell'utente al livello concettuale), ma l'idea di indipendenza logica dei dati è ampiamente adottata.

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

Corsi Sedi

Corso	Aula	Edificio	Piano
Sistemi	N3	OMI	Terra
Reti	N3	OMI	Terra
Controlli	G	Pincherle	Primo

Figura 4: *un esempio di diverse viste sugli stessi dati.*

In Figura 4 un esempio con diverse viste sullo stesso database.

4. Indipendenza dei dati

L'indipendenza dei dati è il tipo di trasparenza dei dati che conta per un DBMS centralizzato. Fa riferimento all'immunità delle applicazioni utente alle modifiche apportate alla definizione e all'organizzazione dei dati. Idealmente, i programmi applicativi non dovrebbero essere esposti ai dettagli della rappresentazione e dell'archiviazione dei dati. Il DBMS fornisce una visione astratta dei dati che nasconde tali dettagli. Esistono due tipi di indipendenza dei dati: indipendenza *fisica* e *logica*.

4.1 Indipendenza logica

L'indipendenza logica dei dati si riferisce alla possibilità di modificare lo schema concettuale senza dover modificare lo schema esterno. L'indipendenza dei dati logici viene utilizzata per separare il livello esterno dalla vista concettuale. Se apportiamo modifiche alla vista concettuale dei dati, la visualizzazione dell'utente dei dati non verrà modificata. L'indipendenza dei dati logici avviene a livello di interfaccia utente.

In pratica, Rende indipendente lo schema esterno da quello logico, consentendo di inserire nuove viste senza alterarlo, o di alterarlo mantenendo inalterate le viste definite in precedenza.

4.2 Indipendenza fisica

L'indipendenza fisica dei dati può essere definita come la capacità di modificare lo schema interno senza dover modificare lo schema concettuale. Se apportiamo modifiche alle dimensioni di archiviazione del server di sistema del database, la struttura concettuale del database non verrà modificata.

L'indipendenza fisica dei dati viene utilizzata per separare i livelli concettuali dai livelli interni ed avviene a livello di interfaccia logica. In pratica, consente di mantenere inalterata la struttura logica dei dati al variare dalla realizzazione fisica del sistema. Consente l'utilizzo di basi di dati su piattaforme diverse, o la distribuzione di una base di dati su più macchine. In Figura 5 è illustrato uno schema che chiarisce la collocazione dei suddetti concetti.

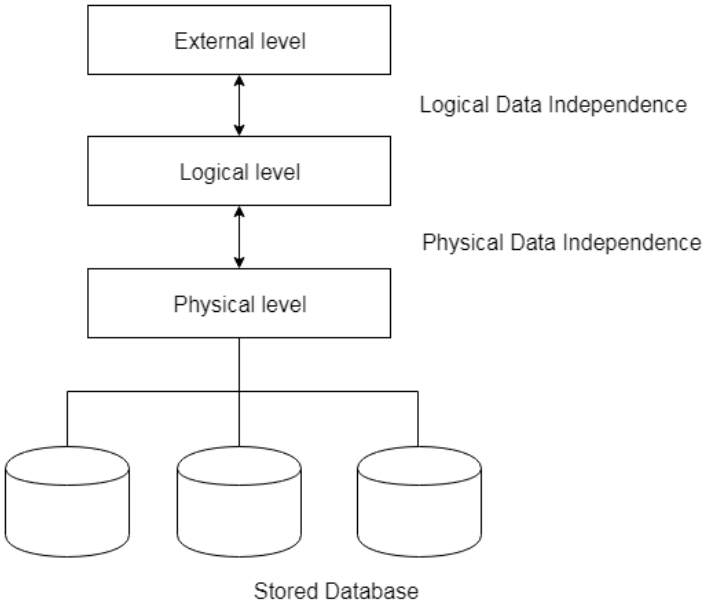

Figura 5: *indipendenza logica e fisica dei dati in un dbms.*

Bibliografia

- Curtin, P., D., Foley, K., Sen, K., Morin, C. (2016). Informatica di base. McGraw-Hill Education.
- Mezzalama, M. and Piccolo, E. (2010). Capire l'informatica. CittàStudi Edizioni.
- Atzeni, P., Ceri, S., Fraternali. P., Paraboschi, S., Torlone, R. (2018). Basi di Dati. McGraw-Hill Education.
- Batini, C., Lenzerini, M. (1988). Basi di Dati. In Cioffi, G. and Falzone, V. (Eds). Calderini. Seconda Edizione.
- ANSI/X3/SPARC Study Group on Data Base Management Systems: (1975), Interim Report. FDT, ACM SIGMOD bulletin. Volume 7, No. 2.
- Michael R. McCaleb (1999). "A Conceptual Data Model of Datum Systems". National Institute of Standards and Technology. August 1999.
- Matthew West and Julian Fowler (1999). Developing High Quality Data Models. The European Process Industries STEP Technical Liaison Executive (EPISTLE).