Introduzione ai Database Management Systems (DBMS)

Prof. Francesco Gobbi

I.I.S.S. Galileo Galilei - Ostiglia (MN)
Materia: Informatica

23 settembre 2024

Cos'è un DBMS?

- **DBMS**: Database Management System
- È un software che permette la gestione dei dati, garantendo accesso, manipolazione e sicurezza degli stessi.
- Esempi di DBMS: MySQL, PostgreSQL, Oracle, SQL Server
- Questi hanno dei vantaggi rispetto ai sistemi file-based

DDL: Cos'è?

- **DDL**: Data Definition Language (Linguaggio di Definizione dei Dati)
- Utilizzato per definire la struttura del database, come la creazione o la modifica di tabelle e indici.
- Comandi principali:
  - `CREATE`: Crea tabelle, viste, indici
  - `ALTER`: Modifica tabelle esistenti
  - `DROP`: Elimina tabelle o altre strutture

Esempio di DDL: Creazione di una tabella

```
CREATE TABLE Studenti (
 id INT PRIMARY KEY,
 nome VARCHAR(50),
 cognome VARCHAR(50),
 data_nascita DATE
);
```

- Crea una tabella chiamata `Studenti` con quattro colonne: `id`, `nome`, `cognome`, `data_nascita`.

DML: Cos'è?

- **DML**: Data Manipulation Language (Linguaggio di Manipolazione dei Dati)
- Utilizzato per inserire, aggiornare e cancellare dati all'interno delle tabelle.
- Comandi principali:
  - `INSERT`: Inserisce nuovi dati
  - `UPDATE`: Aggiorna i dati esistenti
  - `DELETE`: Elimina dati esistenti

Esempio di DML: Inserimento di dati

```
 INSERT INTO Studenti (id, nome, cognome,
data_nascita)
VALUES (1, 'Mario', 'Rossi', '2003-05-10');
```

- Inserisce un nuovo studente con `id` 1, `nome` 'Mario', `cognome` 'Rossi' e `data_nascita` '2003-05-10'.

QL: Cos'è?

- **QL**: Query Language (Linguaggio di Interrogazione)
- Utilizzato per interrogare il database e ottenere informazioni.
- Comando principale:
- `SELECT`: Recupera i dati da una o più tabelle

Esempio di QL: Interrogazione di dati

```
 SELECT nome, cognome
FROM Studenti
WHERE data_nascita > '2003-01-01';
```

- Recupera i nomi e i cognomi degli studenti nati dopo il 1º gennaio 2003.

Vantaggi del DBMS

Il DBMS deve risolvere i problemi del modello file-based e deve avere una **serie di proprietà fondamentali** che ne garantisce il funionamento e le migliorie.

Proprietà 1: Eliminazione della ridondanza e della inconsistenza

- **Gli stessi dati non compaiono più volte in archivi diversi**, grazie a un sistema di archivi integrati di dati.
- Il database **non può presentare campi uguali con valori diversi in archivi differenti**.
- **Esempio pratico**: In un sistema scolastico, l'informazione "indirizzo dello studente" è memorizzata una sola volta nel database e non ripetuta in più archivi, riducendo il rischio di inconsistenze.

Proprietà 2: Facilità di accesso ai dati

- Il **ritrovamento dei dati è facilitato e trasparente al programmatore**.
- Anche nel caso di database molto grandi, le richieste vengono soddisfatte velocemente, anche da più utenti simultanei.
- **Esempio pratico**: In un sistema di e-commerce, i clienti possono cercare e visualizzare rapidamente i loro ordini, anche se migliaia di utenti accedono contemporaneamente al database.

Proprietà 3: Interrogazioni non predefinite

- È **possibile interrogare i dati con linguaggi semplici e standard come SQL**. *Noi utilizzeremo MySQL*.
- Ogni nuova interrogazione può essere costruita con facilità, senza dover modificare la struttura del database.
- **Esempio pratico**: Un'azienda può effettuare una query per trovare tutti i clienti che hanno fatto un acquisto superiore a una certa soglia senza averlo pianificato inizialmente.

Proprietà 4: Integrità dei dati

- **I vincoli di integrità, ovvero delle regole per impedire l'inserimento di dati non corretti da parte dell'utente**, sono memorizzati e controllati dal DBMS.
- Sono previsti **controlli per evitare anomalie causate da programmi o utenti**.
- **Esempio pratico**: Un sistema di gestione ordini impedisce l'inserimento di un prodotto in un ordine se quel prodotto non esiste nel database.

Proprietà 5: Indipendenza dalla struttura logica e fisica dei dati

- **I programmi applicativi sono indipendenti dalla struttura logica con cui i dati sono organizzati**.
- È possibile modificare la struttura del database senza influenzare le applicazioni.
- **Esempio pratico**: Una banca può cambiare la struttura fisica di come i conti bancari sono memorizzati (es. da dischi rigidi a cloud storage) senza dover modificare l'applicazione utilizzata dai dipendenti per visualizzare i dati dei clienti.

Proprietà 6: Utilizzo da più utenti e controllo della concorrenza

- I **dati del database possono essere usati da più utenti contemporaneamente**, con visioni parziali per ciascuno.
- **Il DBMS garantisce che le operazioni svolte in concorrenza** da più utenti non interferiscano tra loro.
- **Esempio pratico**: In un sistema di prenotazione online, più utenti possono prenotare lo stesso volo senza sovrapporsi grazie al controllo della concorrenza, che garantisce la corretta gestione delle prenotazioni.

Proprietà 7: Sicurezza dei dati

- Sono previste **procedure di controllo per impedire accessi non autorizzati ai dati**, quindi per esempio in sezioni del database che sono determinati utent possono inserire o modificare i dati.
- **Protezione da guasti accidentali o intrusioni esterne**.
- **Esempio pratico**: In un sistema bancario, solo i dipendenti autorizzati possono accedere ai conti dei clienti, mentre i clienti possono visualizzare solo i propri dati tramite login sicuro.

Il dizionario dei dati

All'interno di ogni DBMS c'è un **dizionario dei dati o catalogo dei dati**, quindi dei **metadati** del database che servono per spiegare i dati contenuti nello stesso.

- Il dizionario dei dati garantisce l'indipendenza logica del database rispetto ai programmi applicativi utilizzati.
- Il DBMS utilizza il dizionario dei dati per:
  - Realizzare i controlli per garantire l'integrità dei dati.
  - Autorizzare gli utenti in base alle politiche definite per l'accesso ai dati.
  - Fornire servizi per il controllo della consistenza.
- **Esempio pratico**: In un sistema di gestione di un'università, il dizionario dei dati contiene informazioni sulle tabelle degli studenti, i corsi e i vincoli tra essi, come i requisiti per l'iscrizione a un corso.

Modello Entità/Associazioni : Modello(Diagramma) E/R

- Il modello Entità/Associazioni è uno strumento fondamentale per **comunicare e rappresentare** chiaramente la struttura e le relazioni tra i dati di un sistema.
- La lettura di un modello E/R **permette di comprendere il problema analizzato** e di esprimere chiaramente le relazioni in linguaggio naturale.
- Il modello E/R è un modello concettuale per la costruzione del database e per la sua rappresentazione.

Definizioni del Modello E/R : Modello Concettuale

- **Entità:**
  - Un'entità è un oggetto o **concetto del mondo reale che può essere distinto da altri oggetti**. Un'entità rappresenta una "cosa" nel modello E/R.
  - **Esempio:** Nel contesto di una banca, un Cliente, un Conto o un Movimento sono tutte entità.
- **Attributo:**
  - Un attributo è una **proprietà o caratteristica di un'entità che la descrive o la qualifica**. **Ogni entità è descritta da un insieme di attributi**.
  - **Esempio:** Per l'entità Cliente, esempi di attributi sono il Codice Fiscale (usato anche come chiave primaria), il Nome, e il Cognome (usato anche come chiave primaria).
- **Relazione/Associazione:**
  - Una relazione (o associazione) **definisce come due o più entità sono connesse tra loro**. Le relazioni rappresentano le **interazioni tra le entità**.
  - **Esempio:** La relazione "Possedere" può connettere un Cliente con un Conto, indicando che un Cliente possiede uno o più Conti.

Diagramma E/R: Esempio

- La figura mostra un **diagramma E/R di una possibile base dati per un sistema bancario**.
- *Ogni cliente può possedere più conti e ogni conto può variare in base ai movimenti*.

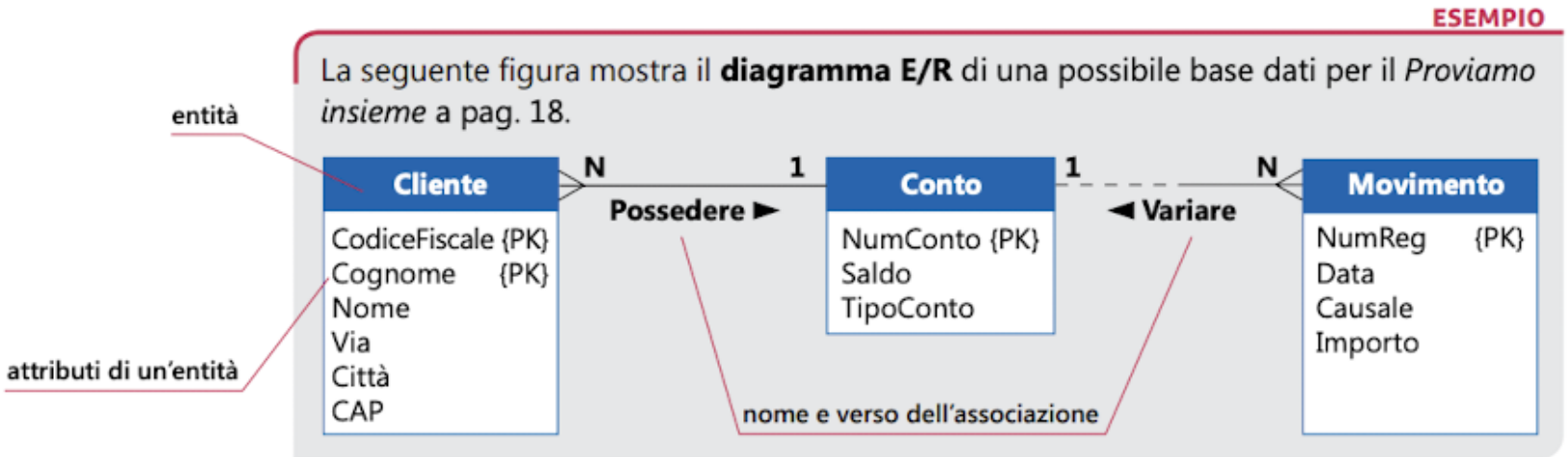

Figura: Diagramma E/R di esempio: Cliente, Conto, Movimento

Esempi di Associazioni: Possedere e Variare

- **Possedere:**
  - Ogni cliente deve possedere uno e un solo conto.
  - Ogni conto deve essere posseduto da uno o più clienti.
- **Variare:**
  - Ogni conto può essere variato da uno o più movimenti e ci possono essere conti non movimentati.
  - Ogni movimento deve variare uno e un solo conto.

Il modello logico: Definizione

- Dopo lo schema concettuale Entità/Associazioni, un database può essere progettato e realizzato tramite il modello logico.
- Il modello logico organizza i dati in modo da consentire operazioni di manipolazione e interrogazione.

Tipi di modelli logici

- Nei primi anni dello sviluppo della teoria dei database, sono emersi tre modelli principali:
  - **Modello gerarchico:** Le associazioni tra entità sono rappresentate con una struttura ad albero, dove ogni record ha un singolo record padre.
  - **Modello reticolare:** Le associazioni sono rappresentate tramite puntatori che collegano i record correlati, formando una rete.
  - **Modello relazionale:** Rappresenta i dati come un insieme di tabelle, oggi il modello più diffuso.

Il modello relazionale : Modello Logico

- Il **modello relazionale è diventato predominante** grazie alla sua semplicità ed efficienza.
- **Rappresenta il database come un insieme di tabelle**, permettendo un'organizzazione naturale e flessibile dei dati.

Vantaggi del modello relazionale

- Il modello relazionale è attualmente considerato il più semplice ed efficace, adattandosi alla classificazione e alla strutturazione dei dati.
- Offre una capacità elevata di manutenzione, ristrutturazione e esportazione dei dati.
- Nato nel 1970, è stato proposto da **Edgar F. Codd**, ricercatore IBM, che ha rivoluzionato l'approccio alla gestione dei dati.

Concetti fondamentali del modello relazionale

- Il modello relazionale si basa su concetti matematici rigorosi, enfatizzando l'uso di un linguaggio matematico universale.
- Due obiettivi principali:
  - Utilizzare un linguaggio conosciuto a livello universale, quale è il linguaggio matematico.
  - Eliminare i problemi di ambiguità nella terminologia e nella simbologia.

- Il modello relazionale è un **modello basato sui valori**:
  - Le associazioni tra entità sono descritte tramite i valori assunti dai campi nelle righe delle tabelle.
  - Non viene fatto uso di puntatori per collegare entità, semplificando la gestione e l'interrogazione dei dati.

Esempio di utilizzo dei modelli gerarchico e reticolare

- Nei modelli gerarchico e reticolare, un esempio pratico è il collegamento tra Cliente e Conto bancario.
- Nel record che descrive l'anagrafica del cliente, è inserito un puntatore che indica la posizione del record relativo al conto del cliente.
- Tuttavia, la gestione di questi modelli richiedeva una capacità elaborativa elevata e complicava operazioni di manutenzione, portando al loro superamento dal modello relazionale.

Tabelle del database: Clienti

Cognome	**Via**	**Città**	**CAP**	**NumConto**
Bianchi	Mazzini	Bergamo	24127	9000
Neri	Garibaldi	Napoli	80120	7000
Rossi	Rosmini	Milano	20125	4500
Galli	Cavour	Bari	70122	5100
Rosa	Crispi	Torino	10137	8000
Verdi	Dante	Roma	00128	5100
Moro	Colombo	Milano	20143	4310

Tabella: Tabella Clienti

- La tabella **Clienti** contiene le informazioni anagrafiche dei clienti e il numero del conto (**NumConto**) associato a ciascuno di essi.
- **NumConto** è il collegamento tra la tabella Clienti e le altre tabelle del database.

Tabelle del database: Conti

NumConto	**Saldo**
9000	120.345,00 €
7000	500,00 €
4500	3.500,00 €
5100	58.500,00 €
8000	6.320,00 €
4310	37,00 €

Tabella: Tabella Conti

- La tabella **Conti** mostra i saldi associati a ciascun conto, identificati dal **NumConto**.
- I saldi forniscono una visione chiara delle finanze dei clienti.

Tabelle del database: Movimenti

NumConto	NumReg	Data	Causale	Importo
9000	1	23/12	Vers	2.500,00 €
7000	2	20/04	Prel	1.250,00 €
4500	3	27/05	Vers	550,00 €
4310	150	21/08	Prel	350,00 €
4310	151	21/08	Prel	536,00 €

Tabella: Tabella Movimenti

- La tabella **Movimenti** registra tutte le operazioni effettuate sui conti, con dettagli su data, causale e importo.
- **NumConto** qui associa ogni movimento al conto specifico.

Concetti Fondamentali del Modello Relazionale

- Nel modello relazionale, le associazioni tra le tabelle sono basate sui valori condivisi nei campi comuni, come **NumConto**.
- **NumConto** in **Clienti**, **Conti** e **Movimenti** permette di collegare i dati tra queste tabelle senza l'uso di puntatori.
- **Esempio:** Il cliente Bianchi possiede il conto con NumConto 9000, che ha un saldo di 120.345,00 € e il 23 dicembre ha effettuato un versamento di 2.500,00 €.

Introduzione al Modello Fisico

- **Il modello fisico dei dati descrive il modo in cui un dato modello logico è realizzato concretamente sulle memorie di massa del computer**.
- Nel caso del modello relazionale, il modello fisico precisa:
  - Come sono realizzate le tabelle.
  - Come implementare i vincoli sui dati.
  - Come rappresentare le associazioni tra tabelle.
  - Come costruire gli indici sui campi di una tabella.
- Il modello fisico si occupa di problemi trattati dai progettisti di uno specifico DBMS.
- Esempi di prodotti DBMS che gestiscono i modelli fisici dei dati includono **Oracle, DB2, MySQL, SQL Server, Access**.

L'architettura ANSI-SPARC

- L'architettura ANSI-SPARC è uno standard per la progettazione e la gestione dei database che definisce tre livelli di astrazione:
  - Livello esterno o concettuale
  - Livello logico
  - Livello interno o fisico
- Questa separazione permette di gestire la complessità del database e di mantenere l'indipendenza tra i dati e le applicazioni.

Livello Esterno

- **Il livello esterno è il più vicino all'utente finale**. Possiamo dire che è il nostro modello concettuale, ovvero il Modello/Diagramma E/R.
- Definisce le viste o le sottoinsiemi dei dati che ciascun utente o gruppo di utenti può vedere.
- Ogni utente ha una vista personalizzata del database che può differire dalle viste di altri utenti.
- **Compiti principali:**
  - Creare e gestire le viste per gli utenti.
  - Garantire la sicurezza e la privacy dei dati mostrando solo le informazioni rilevanti.

Livello Logico

- **Il livello logico rappresenta la struttura logica dell'intero database, di come sono create le tabelle**.
- Fornisce una descrizione globale dei dati e delle relazioni tra essi, indipendentemente da come sono fisicamente memorizzati.
- Questo livello è indipendente dalle applicazioni e dalle modalità di accesso al database.
- **Compiti principali:**
  - Definire le entità, gli attributi e le relazioni del modello dati.
  - Mantenere l'integrità e la coerenza dei dati.
  - Stabilire vincoli e regole per la gestione dei dati.

Livello Fisico

- **Il livello fisico descrive come i dati sono effettivamente memorizzati nel sistema di archiviazione**.
- Gestisce le strutture di memorizzazione e gli accessi fisici ai dati.
- È il livello più vicino al sistema di memorizzazione, il File System (parte del Sistema Operativo), e si occupa delle performance e dell'efficienza del database.
- **Compiti principali:**
  - Definire le strutture fisiche di archiviazione dei dati.
  - Ottimizzare le prestazioni delle operazioni di accesso ai dati.
  - Implementare indici e gestire lo spazio di memorizzazione.

Indipendenza dei Dati

- L'architettura a tre livelli del database realizza meccanismi di astrazione dei dati e assicura la cosiddetta **indipendenza dei dati**.
- Ciò significa che **i livelli superiori non sono influenzati dai cambiamenti che avvengono nei livelli inferiori**.
- Si distinguono **due tipi di indipendenza dei dati:**
  - **Indipendenza logica:** Lo schema esterno non è influenzato dai cambiamenti apportati allo schema logico.
  - **Indipendenza fisica:** Lo schema logico non è influenzato dai cambiamenti apportati allo schema interno dei dati.

Esempi di Indipendenza dei Dati: Logica e Fisica

Esempio di Indipendenza Logica:

- L'aggiunta di un campo, quindi una nuova colonna, nella tabella.
- Le applicazioni che utilizzano questa vista non necessitano di modifiche.

Esempio di Indipendenza Fisica:

- Lo spostamento di una tabella da un dispositivo di memorizzazione a un altro non influenza lo schema logico.
- Gli utenti e le applicazioni non avvertono il cambiamento.

Esempio Pratico: Indipendenza Logica e Fisica

ID Studente	Nome	Cognome
1	Mario	Rossi
2	Anna	Verdi
3	Luca	Bianchi

Tabella: Tabella Studenti

- **Indipendenza Logica:** Se aggiungi un campo **Indirizzo** alla tabella Studenti, le viste esistenti non cambiano, a meno che non lo specifichi.
- **Indipendenza Fisica:** Se la tabella Studenti viene spostata su un altro server, la struttura logica e le applicazioni rimangono invariate.