Modelli di Database e Progettazione Logica

Dal modello concettuale al modello relazionale

Prof. Fedeli Massimo - Tutti i diritti riservati

ITS 4.0 - Fabbrica digitale

16 dicembre 2025

Indice

Progettazione Logica

Definizione

Una volta approntato il modello concettuale (E-R), si procede alla definizione del **modello logico**. Esso consiste in uno schema realizzato in funzione delle caratteristiche del DBMS che si intende utilizzare.

- È più vicino alla rappresentazione informatica dei dati.
- Si ottiene traducendo lo schema concettuale tramite regole definite.
- Deve essere indipendente dalle strutture fisiche.
- Deve essere utilizzabile dai programmi applicativi.

Evoluzione dei modelli logici

Nel tempo si sono succeduti diversi tipi di modelli:

- **Gerarchico** (anni '60): rappresentabile tramite albero.
- **Reticolare** (anni '60): rappresentabile tramite grafo.
- **Relazionale** (anni '70): il più diffuso, basato su tabelle.
- **A oggetti** (anni '80): estensione del paradigma Object-Oriented.
- **XML** (anni '90): per l'esportazione e scambio dati.

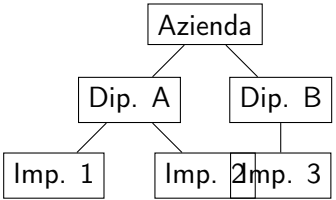
Modello Gerarchico

- Primo modello ad affermarsi sul mercato.
- Dati organizzati in strutture ad **albero**.
- Esiste una **radice** (record principale) da cui partono sottoalberi.
- Relazioni di tipo **Padre-Figlio** (1:N).

Struttura del Modello Gerarchico

Caratteristiche:

- Ogni nodo ha un solo genitore (tranne la radice).
- Un genitore può avere più figli.
- Accesso ai dati sequenziale partendo dalla radice.



Vantaggi e Svantaggi del Modello Gerarchico

Vantaggi:

- Velocità nelle query con percorsi prevedibili.
- Integrità dei dati (relazioni fisse).

Svantaggi:

- Rigidità: difficile modificare la struttura.
- Ridondanza: dati replicati se presenti in più rami.
- Accesso limitato: impossibile collegare nodi di rami diversi direttamente.

Modello Reticolare

- Basato su strutture a **grafo**.
- Estensione del modello gerarchico: non esiste una radice unica.
- Ogni nodo può essere punto di partenza.
- Implementa relazioni **N:N** (molti a molti).
- Utilizza puntatori fisici per connettere i record.

Esempio Reticolare

Scenario: Docenti e Classi

- Un docente insegna in più classi.
- Una classe ha più docenti.
- Struttura complessa di puntatori intrecciati.

Svantaggi principali:

- Spreco di spazio per i puntatori.
- Rigidità nelle modifiche.
- Complessità se i dati non sono direttamente connessi.

Modello Relazionale

Origine

Definito da **Edgar F. Codd** nel 1970.

- Obiettivo: indipendenza dei dati e non duplicazione.
- Basato sul concetto matematico di **relazione tra insiemi**.
- Struttura fondamentale: la **Tabella** (Relazione).
- Esempi software: MySQL, Oracle, SQL Server, Access.

Indipendenza nel Modello Relazionale

Il modello garantisce:

1. **Indipendenza fisica:** modifiche all'hardware o alle strutture di accesso non impattano i programmi.
2. **Indipendenza logica:** modifiche allo schema logico (es. nuove tabelle) non devono bloccare le applicazioni esistenti.

Altri Modelli: A Oggetti e XML

Modello a Oggetti (OODBMS):

- Estende il paradigma Object-Oriented ai DB.
- Gestisce dati complessi (multimedia, CAD).
- Memorizza dati e metodi (comportamenti).

Modello XML:

- Non è un vero modello di DB, ma un formato di scambio.
- Metalinguaggio gerarchico basato su tag.
- Compatibilità universale (file di testo).

Concetti Matematici: Prodotto Cartesiano

Dati due insiemi A_1 e A_2:

Prodotto Cartesiano $A_1 \times A_2$

È l'insieme di tutte le coppie ordinate (x, y) dove $x \in A_1$ e $y \in A_2$.

Esempio:

- $A_1 = \{4, 9, 16\}$
- $A_2 = \{2, 3\}$
- $A_1 \times A_2 = \{(4, 2), (4, 3), (9, 2), (9, 3), (16, 2), (16, 3)\}$

Concetto di Relazione Matematica

Relazione

Una relazione matematica è un **sottoinsieme** del prodotto cartesiano che soddisfa una determinata proprietà.

Esempio Relazione Q ("è quadrato di"):

- Considerando l'esempio precedente.
- $Q = \{(4, 2), (9, 3)\}$

Dalla Matematica alle Tabelle

Insiemi → **Domini**
Prodotto Cartesiano → Combinazioni possibili
Relazione Significativa → **Tabella** Dati

A1	A2
4	2
9	3

Tabella: Tabella "QuadratoDi"

Esempio: Automobili

Domini:

- Modello = {Panda, C3, C4}
- Costruttore = {Fiat, Citroen}

La relazione **ProdottoDa** (sottoinsieme significativo) sarà:

Modello	**Costruttore**
Panda	Fiat
C3	Citroen
C4	Citroen

Terminologia Relazionale

- **Grado:** Numero di colonne (attributi) della relazione.
- **Attributo:** Nome identificativo di una colonna.
- **Dominio:** Insieme dei valori assumibili da un attributo.
- **Cardinalità:** Numero di righe (tuple o n-uple) della tabella.

La Relazione come Tabella

- Ogni riga è una **tupla** (o record).
- Ogni colonna è un **attributo** (o campo).
- I valori in una colonna sono omogenei (stesso dominio).
- L'ordine delle righe e delle colonne è irrilevante.

Chiave Primaria (PK)

Un attributo (o insieme minimo di attributi) che identifica **univocamente** ogni riga della tabella.

Notazione dello Schema

Lo schema di una tabella si rappresenta indicando il nome della relazione seguito dai nomi degli attributi tra parentesi. Le chiavi sono sottolineate.

Automobili(Modello, Costruttore, Segmento, Porte, Posti)

Database Vendite Prodotti (Esempio)

Consideriamo tre tabelle:

1. **Reparti** (CodReparto, NomeReparto)
2. **Prodotti** (CodProdotto, Descrizione, Prezzo, *CodReparto*)
3. **Vendite** (Numero, Data, Quantità, *CodProdotto*)

Le relazioni tra tabelle sono gestite tramite i valori dei dati, non puntatori fisici.

Chiavi Esterne (Foreign Keys)

Definizione

Una **Chiave Esterna (FK)** è un attributo (o insieme di attributi) che fa riferimento alla Chiave Primaria di un'altra tabella.

Esempio:

- *CodReparto* in Prodotti è FK verso Reparti.
- *CodProdotto* in Vendite è FK verso Prodotti.

Serve a garantire l'integrità referenziale.

Requisiti fondamentali di una tabella

1. Tutte le righe hanno lo stesso numero di colonne.
2. Gli attributi sono **atomici** (elementari, non scomponibili).
3. I valori di una colonna sono omogenei.
4. Ogni riga è diversa dalle altre (univocità garantita dalla PK).
5. Ordine delle righe non significativo.

Regole di Derivazione

Il passaggio dal diagramma E-R alle tabelle segue regole precise:

1. Ogni **Entità** diventa una **Relazione** (Tabella).
2. Ogni **Attributo** dell'entità diventa colonna della tabella.
3. L'**Identificatore** diventa **Chiave Primaria**.

Associazione 1:1

Caso Standard: L'associazione 1:1 diventa spesso un'unica relazione che unisce gli attributi di entrambe le entità.

Esempio: Cittadino - CodiceSSN

- **Anagrafe**(CodiceFiscale, Cognome, Nome, ..., CodiceSanitario, Regione)

Associazione 1:1 con Partecipazione Facoltativa

Se la partecipazione è facoltativa (molti valori nulli possibili), conviene mantenere **due tabelle separate**.

Esempio: Dipendente (1) — (1) AutoAziendale (0,1)

- Si aggiunge la chiave primaria dell'entità "forte" come chiave esterna nell'entità "debole".
- **Dipendenti**(Matricola, Cognome, ...)
- **AutoAziendali**(Targa, Modello, *Matricola*)

Associazione 1:N (Uno a Molti)

Regola Fondamentale

Si aggiunge agli attributi dell'entità che sta dal lato "Molti" (N) l'identificatore univoco dell'entità che sta dal lato "Uno" (1). Questo diventa **Chiave Esterna**.

Esempio 1:N

Contratto (1) — (N) Dipendente

- **Contratti**(Codice, Descrizione, StipendioBase)
- **Dipendenti**(Matricola, Cognome, Nome, *CodiceContratto*)

Gli eventuali attributi dell'associazione vanno nella tabella "a molti".

Associazione N:N (Molti a Molti)

Regola

L'associazione N:N diventa una **nuova relazione** (tabella aggiuntiva).

La nuova tabella contiene:

- Le chiavi primarie delle due entità originali (che insieme formano la chiave primaria composta o sono chiavi esterne).
- Gli eventuali attributi dell'associazione.

Esempio N:N

Docente (N) — *Insegnare* — *(N) Classe* Si ottengono tre tabelle:

1. **Docenti**(CodDocente, Cognome, Nome)
2. **Classi**(SiglaClasse, Aula)
3. **Insegnare**(*CodDocente*, *SiglaClasse*, Materia, Ore)

Chiave Primaria nella tabella di associazione N:N

Attenzione alla definizione della chiave primaria della nuova tabella.

- Spesso è la combinazione delle due chiavi esterne.
- A volte serve aggiungere un attributo temporale (es. Data) se l'associazione può ripetersi nel tempo.

Esempio: Studente - Valutazione - Materia
Chiave: (Matricola, CodMateria, Data)

Associazioni Ricorsive

Un'entità è associata a se stessa. *Esempio: Dipendente che supervisiona altri Dipendenti (1:N)*

- Nella tabella **Dipendenti** si aggiunge un campo *Supervisore* che è FK verso la Matricola della stessa tabella.
- **Dipendenti**(Matricola, Nome, *Supervisore*)

Operatori Relazionali

Gli operatori agiscono su una o più relazioni per ottenerne una nuova. Sono la base del linguaggio SQL.

- **Selezione** (σ)
- **Proiezione** (π)
- **Congiunzione o Join** ($\bowtie$)
- **Operazioni insiemistiche** (Unione, Intersezione, Differenza)

Selezione (σ)

Definizione

Estrae le **righe** (tuple) che soddisfano una condizione P.

Caratteristiche:

- Grado: uguale alla tabella originale.
- Cardinalità: $\leq$ tabella originale.
- Notazione: $\sigma_P(T)$

Proiezione (π)

Definizione

Estrae solo alcune **colonne** (attributi) specificate in una lista L.

Caratteristiche:

- Grado: uguale al numero di colonne in L.
- Cardinalità: $\leq$ tabella originale (rimuove i duplicati se l'operazione è puramente insiemistica).
- Notazione: $\pi_L(T)$

Congiunzione (Join - $\bowtie$)

Combina due tabelle R e S collegando le righe che hanno valori corrispondenti negli attributi comuni.

- **Equi-Join:** mantiene le colonne duplicate di confronto.
- **Natural Join:** elimina le colonne duplicate (ridondanza).

Tipi di Join

Join Interno (Inner Join)

Mostra solo le righe che hanno corrispondenza in entrambe le tabelle.

Problema di Assenza

Con l'inner join si perdono le righe che non hanno corrispondenza (es. Clienti senza ordini, Agenti senza clienti).

Join Esterni (Outer Join)

Servono a mantenere le righe anche se non c'è corrispondenza (completando con NULL).

- **Left Join:** Tutte le righe della tabella di Sinistra + corrispondenze di Destra.
- **Right Join:** Tutte le righe della tabella di Destra + corrispondenze di Sinistra.
- **Full Join:** Tutte le righe di entrambe le tabelle.

Self Join

È una congiunzione di una tabella con se stessa.

- Utile per associazioni ricorsive.
- Richiede l'uso di alias per distinguere le due "copie" della tabella.

Esempio: Trovare il nome del capo di ogni impiegato.

Operazioni Insiemistiche

Richiedono che le due tabelle abbiano lo stesso schema (stessi attributi).

- **Unione ($R \cup S$)**: righe in R oppure in S.
- **Intersezione ($R \cap S$)**: righe comuni a R e S.
- **Differenza ($R - S$)**: righe in R ma non in S.

Perché Normalizzare?

Se le tabelle non sono ben progettate, si generano **ridondanze** (dati ripetuti). La ridondanza causa **Anomalie**:

- Occupazione inutile di spazio.
- Inconsistenza dei dati.
- Problemi nelle operazioni di modifica.

Le Anomalie

Esempio Tabella *Inventario* con indirizzo magazzino ripetuto.

1. **Anomalia di Aggiornamento:** Se cambia l'indirizzo, devo aggiornarlo in tutte le righe. Rischio inconsisitenza.
2. **Anomalia di Cancellazione:** Se cancello l'ultimo prodotto di un magazzino, perdo anche l'indirizzo del magazzino.
3. **Anomalia di Inserimento:** Non posso inserire un nuovo magazzino se non ho ancora prodotti da associargli.

Il Processo di Normalizzazione

La normalizzazione è il processo di scomposizione delle tabelle per eliminare le ridondanze e le anomalie.

Concetto chiave: Dipendenza Funzionale

Si ha dipendenza funzionale $A \rightarrow B$ quando il valore di un attributo A determina univocamente il valore di B.

Soluzione Esempio Magazzino

Scomporre la tabella **Inventario** in due tabelle:

1. **Inventario**(Prodotto, Magazzino, Quantità)
2. **Negozi**(Magazzino, Indirizzo)

In questo modo, l'indirizzo è memorizzato una sola volta per ogni magazzino.

Conclusioni

- Il **Modello Relazionale** è lo standard attuale per la maggior parte dei sistemi.
- La corretta **Progettazione Logica** (derivazione E-R e normalizzazione) è cruciale per l'efficienza e l'integrità dei dati.
- L'**Algebra Relazionale** fornisce le basi teoriche per interrogare i dati (SQL).

Grazie per l'attenzione!