

# Appunti sul Modello Relazionale dei Dati

Basato sulle slide del Prof. Danilo Montesi

17 maggio 2025

## Indice

<b>1</b>	<b>Introduzione ai Modelli Logici</b>	<b>2</b>
<b>2</b>	<b>Il Modello Relazionale: Fondamenti</b>	<b>2</b>
<b>3</b>	<b>Relazioni Logiche vs. Tabelle</b>	<b>2</b>
3.1	Relazione Logica (Matematica) . . . . .	2
<b>4</b>	<b>Strutture Dati Non Posizionali</b>	<b>2</b>
<b>5</b>	<b>Il Modello "Value-Based" (Basato su Valori)</b>	<b>3</b>
<b>6</b>	<b>Definizioni Chiave: Schema, Tupla, Istanza</b>	<b>3</b>
<b>7</b>	<b>Gestione di Strutture Dati Annidate</b>	<b>4</b>
<b>8</b>	<b>Informazioni Parziali e Valori NULL</b>	<b>4</b>
<b>9</b>	<b>Vincoli di Integrità</b>	<b>4</b>
9.1	Tipi di Vincoli . . . . .	4
9.2	Vincoli di Tupla (e di Dominio) . . . . .	5
9.3	Chiavi (Superchiavi, Chiavi Candidate, Chiave Primaria) . . . . .	5
9.4	Integrità Referenziale (Chiavi Esterne e Azioni Compensative) . . . . .	5

# 1 Introduzione ai Modelli Logici

I database utilizzano diversi approcci per organizzare logicamente i dati.

- **Modelli Tradizionali:**
  - **Gerarchico:** Struttura ad albero (es. file system). Ogni "figlio" ha un solo "genitore". Navigazione rigida.
  - **Di Rete (Network):** Evoluzione del gerarchico, permette a un "figlio" di avere più "genitori". Più flessibile ma complesso.
  - **Relazionale:** Il modello dominante. Dati organizzati in tabelle.
- **Modelli Più Recenti:**
  - **Object Oriented:** Dati visti come oggetti con proprietà e metodi. Poco comune per DBMS generici.
  - **XML:** Per dati semi-strutturati, spesso complementare al relazionale (es. salvare configurazioni complesse in una cella).

**Caratteristica distintiva:**

- I modelli Gerarchico e Network usano **riferimenti espliciti (puntatori)** tra record.
- Il modello **Relazionale è "value-based"**: i collegamenti avvengono tramite valori condivisi (es. `userID` in `Posts` che corrisponde a `id` in `Users`).

## 2 Il Modello Relazionale: Fondamenti

- **Definito da E. F. Codd nel 1970:** Obiettivo principale era l'**indipendenza dei dati**, separando la rappresentazione logica dalla memorizzazione fisica.
- *Prisma/ORM Insight:* Un ORM astrae ulteriormente, ma il DBMS relazionale sottostante già opera questa separazione.
- **Implementato nei DBMS reali dal 1981.**
- **Basato sulla definizione logica di "relazione"** (dalla teoria degli insiemi), con differenze pratiche.
- **Le relazioni sono rappresentate tramite tabelle.**

**Terminologia Importante:**

- **Relazione Logica (Teoria degli Insiemi):** Un sottoinsieme del prodotto cartesiano di due o più insiemi (domini).
- **Relazione (Modello Relazionale):** Una tabella con righe e colonne.
- **Relationship (Modello Entità-Relazione - ER):** Descrive un legame specifico tra entità (es. "uno studente *si iscrive a* un corso").

## 3 Relazioni Logiche vs. Tabelle

### 3.1 Relazione Logica (Matematica)

- Dati due insiemi (domini)  $D_1 = \{a, b\}$  e  $D_2 = \{x, y, z\}$ .
- Il **prodotto cartesiano**  $D_1 \times D_2$  è l'insieme di tutte le coppie ordinate possibili:  $\{(a, x), (a, y), (a, z), (b, x), (b, y), (b, z)\}$ .
- Una **relazione**  $r$  è un *sottoinsieme* di questo prodotto cartesiano, es:  $r = \{(a, x), (a, z), (b, y)\}$ .
- Questo si estende a  $n$  domini  $D_1, \dots, D_n$ . Una tupla è  $(d_1, \dots, d_n)$ .
- **Proprietà di una relazione logica (come insieme):**
  1. **Nessun ordine tra le tuple:** L'ordine delle righe non ha significato.
  2. **Le tuple sono tutte distinte:** Non ci possono essere righe duplicate.
  3. **Ogni n-upla è ordinata:** L'ordine dei valori *all'interno* di una tupla (cioè, l'ordine delle colonne) è significativo.

**Esempio Posizionale:** `Matches`  $\subseteq$  `string`  $\times$  `string`  $\times$  `int`  $\times$  `int`

Una tupla: (Barca, Bayern, 3, 1) Qui il significato dipende dalla *posizione*: (SquadraCasa, SquadraOspite, GolCasa, GolOspite).

## 4 Strutture Dati Non Posizionali

Nelle tabelle reali, non ci affidiamo solo alla posizione delle colonne.

- **Ogni colonna ha un nome univoco (attributo)** associato a un dominio (tipo di dato). L'attributo definisce il "ruolo" del dominio.
- Esempio: Home (string), Away (string), GoalsH (int), GoalsA (int).
- **La struttura dati diventa non posizionale:** L'ordine specifico delle colonne nella definizione della tabella è irrilevante per la logica.
- *SQL Insight:* SELECT Home, Away FROM Matches e SELECT Away, Home FROM Matches accedono agli stessi dati; cambia solo la presentazione.

**Una tabella rappresenta una relazione se:**

1. Ogni riga può assumere qualsiasi posizione.
2. Ogni colonna può assumere qualsiasi posizione (identificate dal nome).
3. Tutte le righe sono differenti.
4. Tutti i nomi delle colonne (intestazioni) sono differenti.
5. I valori all'interno di una colonna sono omogenei (stesso tipo di dato).

## 5 Il Modello "Value-Based" (Basato su Valori)

Questo è un concetto chiave.

- I riferimenti (collegamenti) tra dati in relazioni (tabelle) diverse sono rappresentati tramite **valori** nelle tuple (righe).
- **Esempio Pratico:**
  - Tabella STUDENT (Number, Surname, Name, ...)
  - Tabella EXAM (Student\_ID, Lecture\_ID, Grade, ...)
  - Per collegare un esame a uno studente, EXAM.Student\_ID conterrà un valore che corrisponde a un valore in STUDENT.Number.
- *SQL Insight:* Questo è come funzionano le FOREIGN KEY e le clausole JOIN ... ON table1.column = table2.column.

**Vantaggi della struttura "value-based":**

1. Indipendenza dalla struttura fisica dei dati.
2. Memorizzazione solo dei dati rilevanti.
3. Utente e programmatore vedono gli stessi dati.
4. Dati facilmente condivisibili tra ambienti diversi.
5. I collegamenti basati su valori possono essere "navigati" in entrambe le direzioni.

## 6 Definizioni Chiave: Schema, Tupla, Istanza

- **Schema di una Relazione (Tabella):**
  - Nome della relazione seguito dall'elenco dei suoi attributi (colonne).
  - Notazione:  $R(A_1, A_2, \dots, A_n)$
  - Esempio: STUDENTS (Number, Surname, Name, YearOfBirth)
  - *SQL Insight:* Corrisponde a CREATE TABLE STUDENTS (...).
- **Schema di un Database:**
  - Insieme degli schemi di tutte le relazioni nel database.
  - Esempio:  $R = \{\text{STUDENTS}(\dots), \text{EXAMS}(\dots), \text{LECTURES}(\dots)\}$
  - *Prisma/ORM Insight:* Il tuo file schema.prisma definisce lo schema.
- **Tupla (Riga):**
  - Una tupla  $t$  su un insieme di attributi  $X$  è una mappatura che associa a ogni attributo  $A \in X$  un valore dal dominio di  $A$ .
  - $t[A]$  esprime il valore della tupla  $t$  per l'attributo  $A$ .
  - Esempio: Se  $t = (6554, \text{Rossi, Mario}, 1978/12/05)$ , allora  $t[\text{Name}] = \text{Mario}$ .
- **Istanza di una Relazione (Contenuto di una Tabella):**
  - Insieme *finito* di tuple che soddisfano lo schema. Dati attuali in un dato momento.
- **Istanza di un Database Relazionale:**
  - Insieme di istanze di relazione, una per ogni schema. Tutti i dati in tutte le tabelle.

**Schema vs. Istanza:** Lo schema è la "definizione" (statico), l'istanza sono i "dati reali" (dinamica).

## 7 Gestione di Strutture Dati Annidate

Il modello relazionale classico (Prima Forma Normale - 1NF) richiede valori **atomici**.

- **Esempio:** Una ricevuta con una *lista* di prodotti.

---

```
Ricevuta 1235, Data 2002/10/12, Totale 39.20
Items:
- 3 x Coperto @ 3.00
- 2 x Antipasto @ 6.20
- ...
```

---

- **Rappresentazione relazionale (unnesting):** Tabelle separate collegate da chiavi.
  1. Tabella RECEIPT (Number, Date, Total)
  2. Tabella COURSE\_ITEM (ReceiptNumber, Qty, Description, Price)ReceiptNumber in COURSE\_ITEM è una chiave esterna.
- **Considerazioni sull' "unnesting":**
  - **Ordine delle righe:** Aggiungere colonna LineNumber o ItemOrder.
  - **Righe ripetute:** LineNumber diventa essenziale per distinguerle.

## 8 Informazioni Parziali e Valori NULL

Spesso i dati sono incompleti.

- **Soluzioni errate per dati mancanti:** Usare valori specifici (0, "", "99").
  - Problemi: Valore "non usato" potrebbe non esistere, o diventare utile; complessità applicativa.
- **Soluzione del Modello Relazionale: Valore NULL**
  - NULL indica l'**assenza di un valore**. Non è 0, non è stringa vuota.
  - NULL **non appartiene al dominio** dell'attributo.
  - Per ogni attributo  $A$ ,  $t[A]$  può mappare a un valore in  $\text{dom}(A)$  o a NULL.
  - Si possono definire vincoli per non ammettere NULL (es. NOT NULL).
- **Diversi significati di NULL (concettuali):**
  - Valore sconosciuto.
  - Valore inesistente/non applicabile.
  - Valore non informativo.
- **SQL Insight:** Si interroga con IS NULL e IS NOT NULL.
- **Troppi NULL:** Possibile segno di progettazione non ottimale.

## 9 Vincoli di Integrità

Regole che i dati devono rispettare per garantire correttezza e consistenza.

- Un vincolo è una **funzione booleana (predicato)**: per ogni istanza, è vero o falso.
- **Perché usare i vincoli?**
  1. Descrizione accurata dello scenario reale.
  2. Supportano la "qualità dei dati".
  3. Utili nella progettazione del Database.
  4. Usati dal DBMS per l'ottimizzazione delle query.
- **Supporto dei DBMS:**
  - Molti tipi supportati nativamente (NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK).
  - Vincoli non supportati devono essere implementati a livello applicativo.

### 9.1 Tipi di Vincoli

1. **Vincoli Intra-relazionali (su una singola tabella):**
  - **Sui valori (o Vincoli di Dominio):** Valori ammissibili per una colonna.
  - Esempio: Grade tra 18 e 30.
  - **SQL Insight:** CHECK (Grade >= 18 AND Grade <= 30).
  - **Sulle tuple:** Valori di più colonne *nella stessa riga*.

- Esempio:  $GrossPay = Deductions + Net$ .
  - *SQL Insight*: CHECK ( $GrossPay = Deductions + Net$ ).
2. **Vincoli Inter-relazionali (tra più tabelle):**
- Integrità referenziale (chiavi esterne).
  - Esempio: IDStudiante in ESAMI deve esistere in STUDENTI.
  - *SQL Insight*: FOREIGN KEY.

## 9.2 Vincoli di Tupla (e di Dominio)

- **Vincoli di Tupla**: Regole sui valori di ogni tupla, indipendentemente dalle altre.
- **Vincoli di Dominio (caso specifico)**: Coinvolgono un singolo attributo.
  - Sintassi: Espressioni booleane che confrontano valori del dominio o espressioni aritmetiche.

## 9.3 Chiavi (Superchiavi, Chiavi Candidate, Chiave Primaria)

Fondamentali per identificare univocamente le tuple e stabilire relazioni.

- **Superchiave (Superkey)**:
  - Insieme di attributi  $K$  tali che non esistono due tuple distinte  $t_1, t_2$  con  $t_1[K] = t_2[K]$ .
  - I valori combinati degli attributi in  $K$  sono unici per ogni riga.
  - Esempio: In STUDENTS (Number, ...), {Number} è superchiave. Anche {Number, Surname} lo è. L'insieme di *tutti* gli attributi è sempre una superchiave.
- **Chiave (o Chiave Candidata - Candidate Key)**:
  - Una superchiave **minimale** (rimuovendo un attributo, cessa di essere superchiave).
  - Una relazione può avere più chiavi candidate.
- **Vincoli, Schema e Istanze**:
  - Le chiavi sono proprietà dello **schema**, non dedotte da una particolare **istanza**.
- **Esistenza delle Chiavi**:
  - Ogni relazione DEVE avere almeno una chiave.
- **Importanza delle Chiavi**:
  1. Garantiscono identificazione univoca e accessibilità.
  2. Permettono di correlare tuple tra relazioni (modello "value-based").
- **Chiavi e Valori NULL**:
  - Attributi parte di una chiave candidata dovrebbero essere NOT NULL.
- **Chiave Primaria (Primary Key - PK)**:
  - Una chiave candidata scelta come meccanismo principale di identificazione.
  - **NON PUÒ contenere valori NULL**.
  - Ogni relazione ha al massimo una PK. Spesso sottolineata.
  - *SQL Insight*: PRIMARY KEY (attribute\_list) implica UNIQUE e NOT NULL.

## 9.4 Integrità Referenziale (Chiavi Esterne e Azioni Compensative)

Garantisce coerenza dei collegamenti tra tabelle.

- **Vincolo di Integrità Referenziale (o Chiave Esterna - Foreign Key - FK)**:
  - Un insieme di attributi  $X$  in  $R_1$  (tabella referenziante) è una FK che riferenzia la PK (o una chiave candidata univoca) di  $R_2$  (tabella referenziata) se:
    1. Gli attributi  $X$  in  $R_1$  e la PK di  $R_2$  hanno domini compatibili.
    2. Per ogni tupla in  $R_1$ , i valori di  $X$  devono:
      - \* Essere NULL (se permesso).
      - \* Oppure, corrispondere a un valore esistente nella PK di una tupla in  $R_2$ .
- Esempio:

---

```
-- Tabella POLICEMAN
-- ID (PK), Surname, Name

-- Tabella INFRINGEMENT
-- Code (PK), Date, Policeman_ID (FK -> POLICEMAN.ID), ...
CREATE TABLE INFRINGEMENT (
  Code INT PRIMARY KEY,
```

```

EventDate DATE,
Policeman_ID INT,
-- ... altre colonne ...
FOREIGN KEY (Policeman_ID) REFERENCES POLICEMAN(ID)
);

```

---

- **Chiavi Esterne e NULL:**
  - Una FK può contenere NULL se la relazione è opzionale.
  - Esempio: EMPLOYEE (ID, Name, Project\_Code). Se Project\_Code è FK, un impiegato può avere Project\_Code = NULL.
- **Azioni Compensative (se si viola l'integrità referenziale):**
  - Azioni su DELETE/UPDATE sulla tabella referenziata ( $R_2$ ):
    1. RESTRICT (o NO ACTION - **default**): Operazione rifiutata.
    2. CASCADE:
      - \* ON DELETE CASCADE: Elimina righe referenzianti in  $R_1$ .
      - \* ON UPDATE CASCADE: Aggiorna valori FK in  $R_1$ .
    3. SET NULL:
      - \* ON DELETE SET NULL: Imposta FK in  $R_1$  a NULL.
      - \* ON UPDATE SET NULL: (simile).
    4. SET DEFAULT:
      - \* ON DELETE SET DEFAULT: Imposta FK in  $R_1$  al valore di default.
- *SQL Insight:*

---

```

FOREIGN KEY (Project_Code) REFERENCES PROJECT(Code)
ON DELETE SET NULL
ON UPDATE CASCADE;

```

---

- **Vincoli su Attributi Multipli (Chiavi Composte):**
  - PK o FK possono essere composte da più attributi.
  - L'ordine degli attributi nella definizione della FK deve corrispondere a quello della PK referenziata.