

# Database

## Riassunto corso di base di dati unipr del professore Flavio Bertini, a cura di Edoardo D'Errico.

Le basi di dati fanno parte del sistema informatico.

è opportuno distinguere il dato dall'informazione: il **dato** da solo non ha alcun significato, ma una volta interpretato e correlato opportunamente, diventa **informazione**, che permette di arricchire la nostra conoscenza del mondo.

Un **Dbms** è un software in grado di gestire collezioni di dati che siano grandi, condivise, persistenti, assicurando la loro affidabilità e privacy. Un **Database** è una collezione di dati gestita da un dbms.

Alla mente del genio, sorge spontanea una domanda: "Concettualmente, come posso descrivere questi dati e quale struttura devono avere per essere elaborati da una macchina?" Abbiamo diversi modi per descrivere i dati, ognuno con livello di astrazione differente. Il livello concettuale è la rappresentazione più vicina alla logica umana ed è indipendente da ogni sistema, per la progettazione concettuale il modello più utilizzato è quello *E – R*.

Ragionando in maniera meno astratta, la maniera in cui rappresentare i dati in maniera comprensibile ad una macchina, in maniera indipendente dalla rappresentazione fisica, è descritto da il **Modello logico dei dati**. Tra i più comuni abbiamo:

1. Modello relazionale.
2. Modello gerarchico(basato su alberi, che sarebbero comunque grafi).
3. Modello reticolare(basato su grafi).
4. Modello a oggetti.
5. Modello XML.
6. Modello semi strutturato e flessibile(e.g NoSql).

I dbms seguono l'architettura a tre livelli ANSI-SPARC, i dati sono descritti secondo tre differenti livelli di astrazione, per ciascun livello esiste un'opportuno schema.

1. Livello esterno: esso si occupa della interlocuzione con gli applicativi, e quindi con gli utenti finali, infatti rappresenta la visione del database da parte dell'utente e quindi descrive quella parte del database che è importante per il singolo utente o per un gruppo di utenti.
2. Livello logico: Precedentemente descritto.
3. Livello fisico come vengono effettivamente memorizzati i dati dalla macchina(e.g. strutture dati utilizzate).

Questa architettura permette di modificare ognuno di questi livelli senza dover modificare gli altri 2.

Per comunicare con un dbms abbiamo 2 tipi di linguaggi:

1. DDL(data definition language) utilizzati per definire schemi logici, esterni, fisici e le autorizzazioni.

esempio:

```
CREATE TABLE hours (  
    course CHAR(20),  
    teacher CHAR(20),  
    room CHAR(4),  
    hour CHAR(5)  
)
```

1. DML(data manipulation language) utilizzati per le query e aggiornamento di istanze di basi di dati.

esempio:

```
SELECT * from T;
```

## Modello relazionale

In ogni base di dati esistono dati che rimangono invariati nel tempo, detti **schemi** e altri che variano detti **istanze**. Nel caso di un modello Relazionale, lo schema è costituito dalla sua intestazione, ovvero nome della relazione  $R$  seguito dai suoi attributi, che corrispondono alle colonne.  $R(A_1, A_2, \dots, A_n)$

L'**istanza** invece è costituita dall'insieme, variante nel tempo delle sue righe, ognuna di esse fa riferimento allo schema e solo attraverso di esso i dati possono essere interpretati.

Il modello relazionale si basa su il concetto di relazione matematica e tabella. In maniera intuitiva, possiamo dire che tra queste 2, esiste un isomorfismo, la correlazione di questi concetti, uno formale e l'altro intuitivo è uno dei motivi che ha permesso il successo di questo modello.

Una relazione è un sottoinsieme del prodotto cartesiano di  $n$  insiemi, chiamati domini della relazione.  $S \subseteq D_1 \times D_2 \times \dots \times D_n$

Esempio di relazione logica: Matches  $\subseteq \text{string} \times \text{string} \times \text{int} \times \text{int}$

Barca	Bayern	3	1
Bayern	Barca	2	0
Barca	Psg	0	2
Psg	Real	0	1

In quanto il prodotto cartesiano è un insieme, inferiamo facilmente che

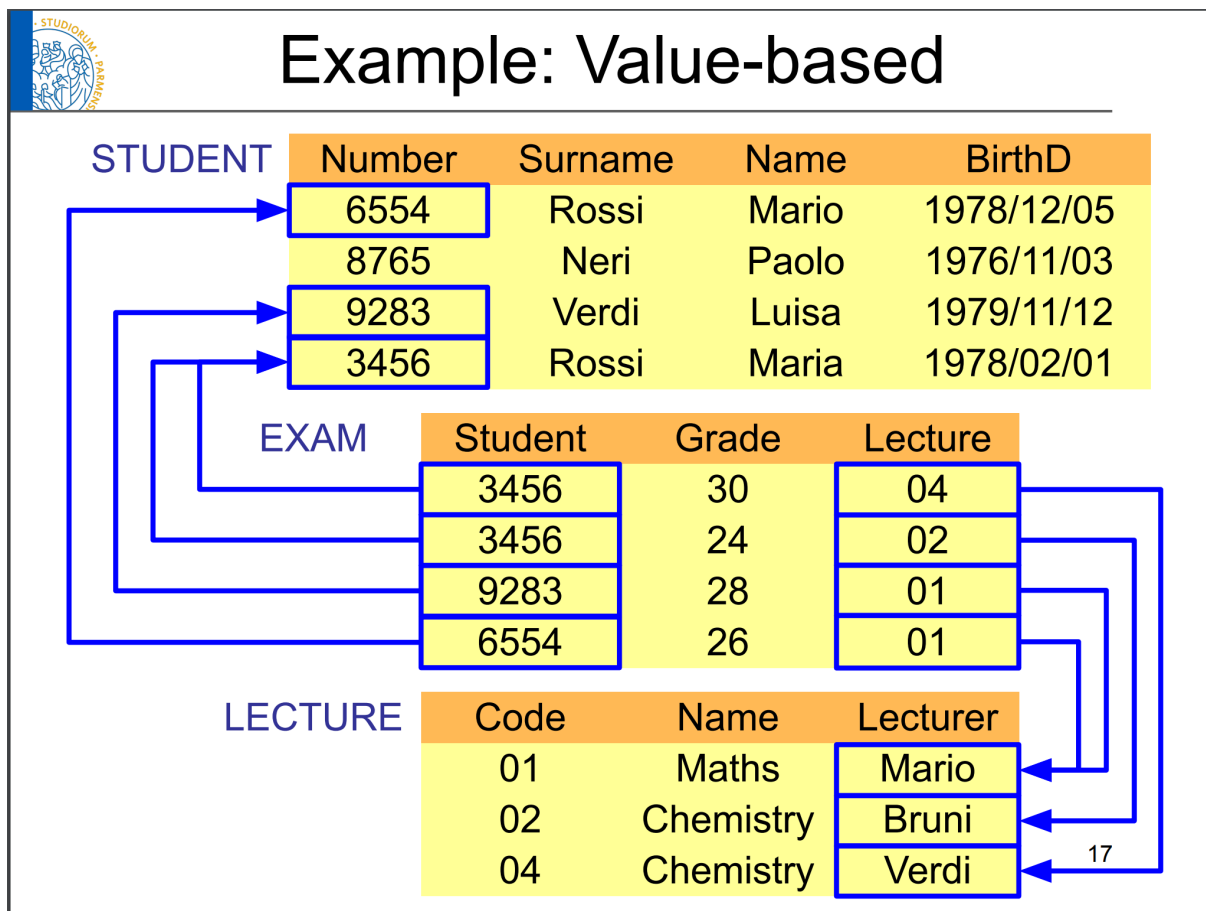
1. Non è definito un ordinamento tra le  $n$ -uple.
2. Le  $n$ -uple sono tutte distinte.

Allo stesso tempo però, è definito un ordinamento fra i domini, senza il quale non riusciremmo ad interpretare correttamente il significato della relazione, ad esempio in questo caso il primo dominio corrisponde alle squadre in casa ed il secondo a quelle in trasferta. Noi vogliamo una struttura che non sia posizionale. Ciò che vogliamo noi è poter far riferimento ad un valore in maniera simbolica, dunque indipendente dalla posizione.

Per ovviare a questo problema, andiamo ad associare ad ogni attributo un dominio. esempio:

SquadraDiCasa	SquadraOspitata	RetiCasa	RetiOspitata
Barca	Bayern	3	1
Bayern	Barca	2	0
Barca	Psg	0	2
Psg	Real	0	1

Spesso viene utile associare più relazioni, nel caso del modello relazionale ciò avviene tramite valori, il che rispetto all'utilizzo di puntatori (altro approccio) risulta più astratto e più indipendente dal contesto.



Il modello relazionale permette di modellare anche realtà complicate, nonostante spesso avvenga in modo indiretto e rigido.

Restaurant Da Filippo Via Roma 2, Roma		
Receipt 1235 at 2002/10/12		
3	Cover Charge	3,00
2	Appetizer	6,20
3	Soup	12,00
2	Meat Loaf	18,00
Total Sum		39,20

Restaurant Da Filippo Via Roma 2, Roma		
Receipt 1240 at 2002/10/13		
2	Cover Charge	2,00
2	Appetizer	7,00
2	Soup	8,00
2	Fish	20,00
2	Coffee	2,00
Total Sum		39,00

In questo caso abbiamo bisogno di due relazioni, in quanto il numero di ordinazioni non è fisso.

RECEIPT	Number	Data	Total
	1235	12/10/2002	39,20
	1240	13/10/2002	39,00

COURSE	Receipt	Row	Qty	Description	Price
	1235	1	3	Cover Charge	3,00
	1235	2	2	Appetizer	6,20
	1235	3	3	Soup	12,00
	1235	4	2	Meat Loaf	18,00
	1240	1	2	Cover Charge	2,00
	...	...	...	...	...

Ad esempio con un modello NoSql avremmo potuto descrivere questo caso più agevolmente con una sola relazione.

Per indicare l'assenza di un valore si può utilizzare il valore null.(Ci sono diversi contesti che meriterebbero tipi più specifici,esempio unknown,inexistant,uninformative)

## Definizioni

- Sia  $X$  l'insieme degli attributi e  $D$  l'insieme dei domini, specifichiamo la corrispondenza fra attributi e domini per mezzo di una funzione  $\text{dom} : X \rightarrow D$  che  $\forall A \in X$  associa un valore del dominio  $\text{dom}(A) \in D$ .
- Una **tupla** su un insieme di attributi  $X$  è una funzione  $t$  che  $\forall A \in X$  associa un valore del dominio  $\text{dom}(A)$ .

La nostra definizione di relazione verrà rivisitata per renderla non posizionale.

- Una relazione su  $X$  è un insieme di tuple su  $X$ .

Notazione:  $T[A]$  dove  $A \in X$  indicherà il valore di  $A$  su  $t$ .

esempio: se  $t$  è la tupla rappresentata dalla prima riga della tabella,  $t[\text{SquadraOspitata}] = \text{Lazio}$ .

- Uno **schema di Relazione** è costituito da un simbolo  $R$  detto nome della relazione e da un insieme di attributi  $X = \{A_1, A_2, \dots, A_n\}$ , lo schema viene indicato come  $R(X)$ . Ovviamente ad ogni attributo viene associato un dominio come precedentemente descritto.
- Uno **schema di database** è un insieme di schemi di relazione  $R = \{R_1(X_1), R_2(X_2), \dots, R_n(X_n)\}$ .

Esempio: nell'immagine precedente possiamo dire che

$R = \{\text{Student}\{\text{Number}, \text{Surname}, \text{Name}, \text{BirthD}\},$

$\text{Exam}\{\text{Student}, \text{Grade}, \text{Lecture}\},$

$\text{Lecture}\{\text{Code}, \text{Name}, \text{Lecturer}\}\}$

- Un'**istanza di relazione** o **relazione su uno schema**  $R(X)$  è un insieme  $r$  di tuple su  $X$ .
- Un'**istanza di base di dati** o **base di dati** su uno schema  $R = \{R_1(X_1), R_2(X_2), \dots, R_n(X_n)\}$  è un insieme di relazioni  $r = \{r_1, r_2, \dots, r_n\}$  dove ogni  $r_i$  per  $1 \leq i \leq n$ , è una relazione sullo schema  $R_i(X_i)$ .

## Vincoli di integrità

I vincoli di integrità sono delle proprietà (che possono essere interpretate come un predicato(v-f)) che devono essere verificate per rappresentare le informazioni in maniera accurata, corretta e di qualità. Nel caso il dbms non supporti tutti i vincoli il programmatore può programmare il constraint fuori dal dbms.

Abbiamo diversi tipi di vincoli:

- Vincolo intrarelazionale, ovvero il suo soddisfacimento è definito rispetto ad una singola relazione. esempio vincolo su una tupla (una tupla deve soddisfare una proprietà) o su un valore (detto anche vincolo di dominio).
- Vincolo interrelazionale, ovvero coinvolge più relazioni.

Abbiamo anche i vincoli di integrità referenziale, prima di parlare di essi però è necessario definire il concetto di chiave e superchiave. Concettualmente una **chiave** sarebbe un insieme di attributi che identificano univocamente una tupla in una relazione.

- Un insieme  $K$  di attributi è **superchiave** di  $r$  se  $r$  non contiene due tuple distinte  $t_1, t_2 : r[t_1] = r[t_2]$
- $K$  è **chiave** se è una **superchiave minimale** di  $r$ , ovvero non contiene un'altra superchiave.

Notare che ogni relazione contiene una superchiave che è l'insieme di tutti gli attributi e dunque almeno 1 chiave. Le chiavi permettono di correlare tuple su relazioni differenti, ovviamente mi aspetto che queste correlazioni siano coerenti.

Più formalmente, un **vincolo di integrità referenziale**, chiamato anche **foreign key** tra gli attributi  $X$  di una relazione  $R_1$  e una relazione  $R_2$  impone che i valori su  $X$  in  $R_1$  siano chiave primaria di  $R_2$ .

Un attributo  $A$  in  $R_1$  è una chiave esterna che si riferisce ad una relazione  $R_2$  se:

- L'attributo  $A$  ha lo stesso dominio dell'attributo che è chiave primaria di  $R_2$ .
- Il valore dell'attributo FK in una tupla  $t_1$  in  $R_1$  è null oppure è il valore di una tupla  $t_2$  in  $R_2$  per cui  $t_1[\text{FK}] = t_2[\text{PK}]$ .

Nel caso un vincolo referenziale venisse violato, ad esempio una tupla contenente la chiave primaria venisse rimossa, abbiamo diversi approcci:

- Rimozione a cascata (viene rimossa anche tupla contenente FK).
- Introduzione di valore null nella FK oppure valore di default.

Nota, nel caso di constraint su più attributi l'ordine degli attributi è importante.

## **Algebra relazionale**

Vedere slide, inutile riassumere.