

# Basi di Dati

# O. Introduzione

Per comprendere i contenuti del corso è bene fornire delle definizioni di base:

**Def** Un dato è una rappresentazione originaria, non interpretata di un evento o di un fenomeno.

La rappresentazione viene effettuata attraverso uno codice.

**ES** 17 5 44 21 15

**Def** L'informazione è un insieme di dati che vengono interpretati e resi significativi dal destinatario

ES	°C (precipitazione)	Km/h (umidità)	ora
17	5	44	21 15

**Def** Una base di dati rappresenta un aspetto del mondo reale che è di interesse per uno specifico scopo.

È un insieme di dati coerenti e con un significato preciso per un particolare insieme di utenti.

**oss** Non esiste, quindi, una base di dati che raccolga tutte le informazioni. Deve sempre fare riferimento ad uno specifico problema del mondo reale e deve avere uno scopo ben preciso (per quanto complesso).

Questo viene definito mini-mondo di interesse.

**ES** Una base di dati che contiene misurazioni atmosferiche non può contenere dati di transazioni bancarie.

**Def** Il software che gestisce una base di dati si chiama DBMS, DataBase Management Software.

Un DBMS è un insieme di programmi che permettono la creazione, le letture, le modifiche e la condivisione delle basi di dati.

**oss** Un DBMS ha anche l'importante compito di controllare le ridondanze (ovvero la duplicazione dello stesso dato in più punti del database). Questo può ridurre il dispendio di spazio e ridurre le inconsistenze nei dati.

## 0.1 Contenuti

Il corso tratta le progettazione di una base di dati nella sua interezza:

↳ livello concettuale: rappresenta i requisiti del database in modo astratto. (noi utilizzeremo il modello Entità-Associazione)

↳ livello logico: tiene conto delle strutture implementative rimanendo però indipendente dai dati. (noi utilizzeremo il modello relazionale)

↳ livello fisico: Utilizzeremo il linguaggio SQL per realizzare la base di dati.

## 1. Progettazione Concettuale - Il Modello Entità-Associazione

Il modello Entità-Associazione (o Entity-Relationship o ER) è stato ideato da P.P. Chen nel 1976 ed è un modello concettuale spesso utilizzato nelle progettazione concettuale.

**oss** Gli schemi scritti in questo modo sono comprensibili anche ad un committente non tecnico

## 1.1 Entità

**Def** L'entità rappresenta una classe di oggetti del mini-mondo. Questi oggetti possono essere fisici (giocatore di una squadra) oppure concettuali (la transazione di un acquisto).

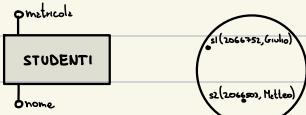
Entity

Vengono rappresentati con un rettangolo all'interno del quale viene scritto il nome della classe.

### 1.1.1 Attributi

**Def** Gli attributi di un'entità sono delle proprietà che la contraddistinguono.

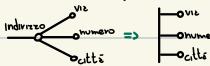
Vengono rappresentati da un segmento ed un pallino.



**Def** Un attributo si dice semplice se hanno un dominio di dati e possono assumere uno o più valori.

**Def** Un attributo si dice composto se è costituito da più attributi semplici.

**oss** Gli attributi complessi devono essere tradotti in attributi semplici.



**Def** Gli attributi possono essere multivalue, si specifica le cardinalità minima e massima con una tuple (min, max)

Si possono utilizzare n e m per indicare dei valori generici (**oss** (n,m))

**Def** Gli attributi possono essere opzionali, in tal caso min=0. Se min>1 l'attributo è obbligatorio.

**Def** Un attributo che può avere solo un numero finito di valori si dice variabile categorica.

**oss** Per rappresentare una variabile categorica si può usare l'enumerazione per limitare gli input.

Non però convertiremo questo tipo di attributi in entità per maggiore flessibilità all'aggiunta di elementi.

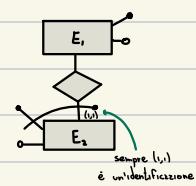
**Def** Diciamo identificatore di un'entità il sottoinsieme (minimo) di attributi che identificano gli elementi.

Vengono rappresentati da un segmento ed un pallino pieno.



**Def** Un'entità si dice debole se è identificata da un'altra entità (detta forte).

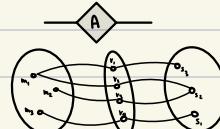
L'entità debole è costata all'implementazione quindi sono da evitare.



## 1.2 Associazione

**Def** Il costrutto associazione permette di fare riferimento ad un'altra entità in un'attributo.

Dal punto di vista insiemistico è un sottoinsieme del prodotto cartesiano  $A \subseteq E_1 \times E_2$ .



Viene rappresentato con un rombo all'interno del quale viene scritto il nome dell'associazione

Oss Una relazione, come un'entità non può avere elementi uguali.

Le associazioni possono collegare 2 o più entità:

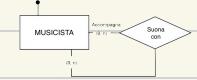
- Associazioni ricorsive



- Associazioni binarie (le solite, più usate)

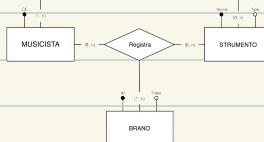
- Associazioni n-arie (3/4 hanno applicazioni anche nel corso, da 6 in poi c'è un errore nello schema)

Oss Costruendo un'associazione ricorsiva è importante pensare bene alle cardinalità. Può essere utile dare un verso di lettura dando un nome ad ogni rombo.



Oss Un'associazione n-aria porta un significato diverso delle associazioni binarie dei suoi elementi.

L'esempio a destra è perfettamente corretto in certi casi.



Oss Quando una delle entità di un'associazione n-aria partecipa all'associazione con cardinalità (1,1) si può trasformare l'associazione in una binaria e una (n-1)-aria.

### 1.2.1 Attributi

**Def** Ogni associazione può avere attributi, proprio come quelli di un'entità. Questi sono attributi che non appartengono a nessuna entità, ma alla loro relazione.

es. l'anno in cui il musicista ha iniziato a suonare un certo strumento.

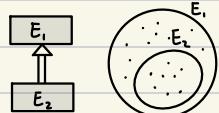
### 1.2.2 Cardinalità

Trovare le cardinalità di associazioni n-arie può risultare complesso, una procedura utile è la seguente:

1. Scegliere un'entità e sceglierne uno specifico elemento
2. Chiedersi quante (n-1)-tuple di elementi delle altre entità possono trovarsi nell'attributo.
3. Ripetere per tutte le entità dell'associazione

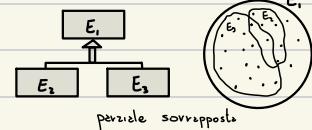
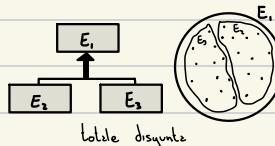
### 1.3 Enhanced Entity-Relationship Model

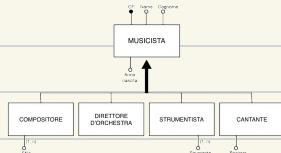
**Def** Diciamo specializzazioni o generalizzazioni il rapporto tra due o più entità assimilabile a quello di sottoclasse e superclasse.



Una specializzazione può essere:

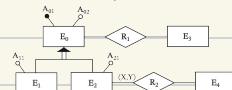
- totale (freccia piena) o parziale (freccia vuota)
- disgiunta o sovrapposta



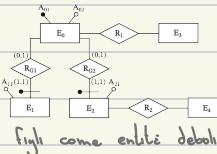
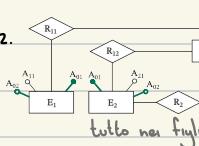
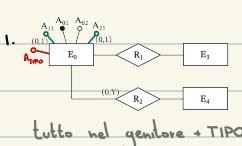


### 1.3.1 Ristrutturazione delle generalizzazioni

Dato che i comuni DBMS non consentono di rappresentare le generalizzazioni, è spesso necessario trasformare il costrutto in entità e relazioni. In particolare, dato il costrutto:



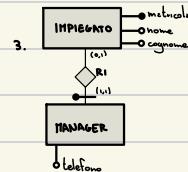
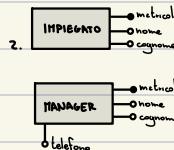
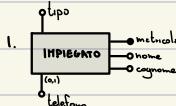
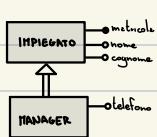
Posso trasformarlo in:



1. occupa più memoria per la presenza dei NULL e attributo TIPO.

2. solo se generalizzazione totale

3. solo se generalizzazione parziale



### 1.4 Scelte di Rindondanza

Spesso l'aggiunta di alcuni dati già contenuti (rindondanza) migliora la velocità delle query.

La domanda è quando questo è vantaggioso, ricordando che i dati rindondanti vengono calcolati, salvati e aggiornati (regole di derivazione)

Per fare questo usiamo le tabelle delle operazioni e dei volumi.

#### ESEMPIO



Scelgo ① o ②?

compila le tabelle delle operazioni e dei volumi.

### Tabelle delle operazioni

OP1: inserire una fattura

INPUT (ID, netto, IVA)

Freq. media 10/giorno

OP2: leggere una fattura

INPUT (ID)

Freq. media 100/giorno

### Tabelle dei volumi

NAME	TIPO	VOLUME
fattura	E	1'000

confronto:

①			②		
OP1	Tipo	Quantità	OP1	Tipo	Quantità
S		1 (x2)	S		1 (x2)
			L		1
			S		1 (x2)
					5 OP × 10
2 OP × 10					
OP2	Tipo	Quantità	OP2	Tipo	Quantità
L		1	L		1
1 OP × 100			1 OP × 100		
120 op/giorno			150 op/giorno		

In verde dei moltiplicatori  
perché la scrittura è più difficile

In questo caso la duplicazione è sconveniente. (Si fanno più operazioni)

## 1.5 Costruzione dello Schema ER

- Costruire un glossario che per ogni termine contenga: descrizione, sinonimi, collegamenti.

↳ Sostituire i sinonimi nel testo dei requisiti

↳ Dindicare il testo per ogni termine

- Costruire le entità, ogni concetto che descrive una classe di oggetti e/o ha proprietà significative con esistenza autonoma dovrebbe essere trasformato in entità

- Costruire le relazioni, se due o più entità hanno un concetto che le associa.

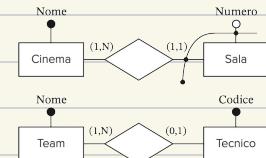
### 1.5.1 Design Pattern

Possiamo usare dei pattern di design per modellare le informazioni sulle entità.

#### PART OF

Sono tipicamente relazioni one-to-many e vogliono rappresentare il fatto che un'entità è parte di un'altra.

Nel primo caso una sala non può esistere senza un cinema.



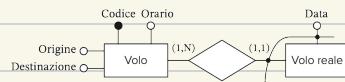
Nel secondo caso un tecnico può esistere anche senza un team.



#### INSTANCE OF

Rappresentano istanze di un'entità più generica, in questo modo

non dobbiamo ripetere più volte le stesse informazioni.



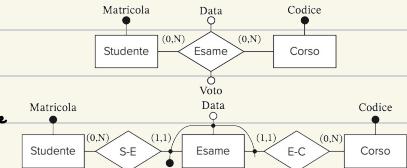
#### REIEZIONE

La relazione è il pattern di conversione di elementi in entità.

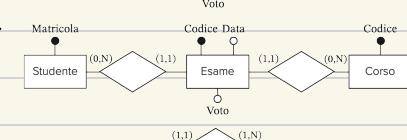
La relazione di attributo consiste nel convertire attributi in entità.

La relazione di relazione binaria consiste nel convertire una relazione in entità.

In particolare nell'esempio è utile farlo per permettere allo studente di fare l'esame più di una volta.

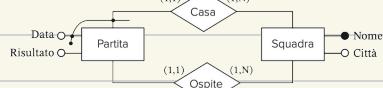


Si trovano così due soluzioni equivalenti, soprattutto se viste nella progettazione concettuale.

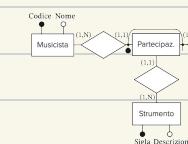


La relazione di relazione ricorsiva converte la relazione ricorsiva partita

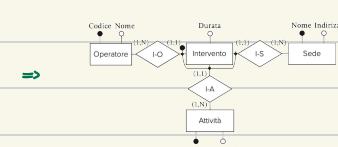
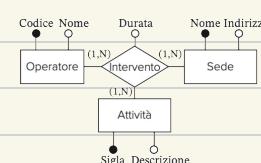
in un'entità, permettendo così più partite tra le stesse squadre.



#### Attributo di relazione



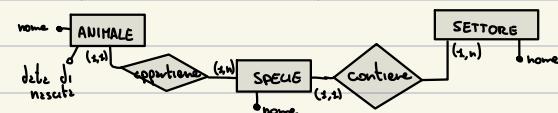
#### Relazione ternaria



Le relazioni sono da applicare quando ci rendiamo conto che un attributo dovrebbe essere un'entità.

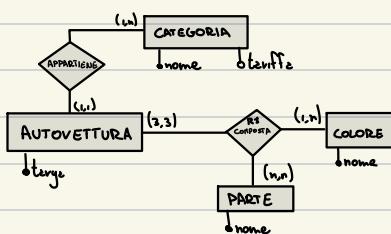
ES

In un giardino zoologico ci sono degli animali appartenenti a una specie e eventi una carta età; ogni specie è localizzata in un settore (avente un nome dello zoo)



ES

Una agenzia di noleggio di autovetture ha un parco macchine, ognuna delle quali ha una targa, un colore per ogni parte (tettuccio, cofano, portiera, ...) e fa parte di un categoria; per ogni categoria c'è una tariffa di noleggio.



- 1) L'entità parte contiene solo 3 elementi (tetto, laterale, posteriore)
- 2) Gli elementi coinvolti nell'associazione R1 della parte dell'auto devono essere valori distinti per le parti coinvolte

## 2. Il Modello Relazionale

Il modello relazionale viene proposto da E.F. Codd nel 1970 e si fonda sul concetto matematico di relazione rappresentato da una tabella.

Il modello risponde al requisito di indipendenza fisica dei dati, ovvero c'è una netta separazione tra il modello e la sua implementazione fisica.

### 2.1 Relazione

#### 2.1.1 Relazione Matematica

**Def** Diciamo tupla o n-uple una sequenza ordinata di valori di attributi.

**Def** Dati gli insiemi  $D_1, \dots, D_n$ , il prodotto cartesiano è l'insieme di tutte le possibili n-uple:

$$D_1 \times \dots \times D_n = \{(d_1, \dots, d_n) | d_i \in D_1, \dots, d_n \in D_n\}$$

**Def** Una relazione è un sottoinsieme del prodotto cartesiano:  $R \subseteq D_1 \times \dots \times D_n$

dove  $D_1, \dots, D_n$  si dicono domini della relazione e  $n$  è il suo grado.

es.  $R \subseteq \text{numero} \times \text{stringa} \rightarrow$

200	Pippo
201	Pluto
202	Topolino

**Def**  $|R|$  è la cardinalità della relazione

**Oss** Una relazione è un insieme quindi le n-uple sono uniche e non ordinate

### 2.1.2 Relazione nel Modello Relazionale

La relazione nel modello relazionale è simile alla relazione matematica con alcune differenze:

- I componenti della relazione sono detti attributi, ognuno caratterizzato da un nome e da un set di valori (il dominio)
- Una relazione può essere rappresentata da una tabella con gli attributi come colonne, il loro ordine è irrilevante

**Def** Sia  $X = \{A_1, \dots, A_n\}$  l'insieme degli attributi,  $D = \{D_1, \dots, D_n\}$  l'insieme dei domini.

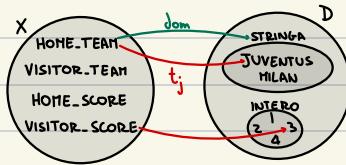
la funzione  $\text{dom}: X \rightarrow D$  mappa ogni attributo al suo dominio  
 $A_i \rightarrow D_i$

**Def** La funzione  $t_j: X \rightarrow D$  mappa ogni attributo ad un valore del suo dominio  
 $A_i \rightarrow v_{ij} \in D_i$

**Def** Uno schema di relazione  $R(A_1, \dots, A_n)$  consiste di un nome e un insieme ordinato di attributi.

**Def** Un'istanza di relazione dello schema di relazione  $R(A_1, \dots, A_n)$  è un insieme di tuple

$r = \{t_1, \dots, t_m\}$  dove ogni tupla è una lista di valori  $t_j = \langle v_{1j}, \dots, v_{nj} \rangle$



R	A <sub>1</sub>	...	A <sub>n</sub>
t <sub>1</sub>	v <sub>11</sub>	...	v <sub>1n</sub>
⋮	⋮	⋮	⋮
t <sub>m</sub>	v <sub>m1</sub>	...	v <sub>mn</sub>

**NULL**

Ogni attributo può assumere uno dei valori del suo dominio oppure NULL.

Può essere utilizzato se il valore non è conosciuto oppure per indicare che l'attributo non si applica alla tupla.

Non sono mai da utilizzare i valori inutilizzati (es. lo 0) al posto del NULL.

### 2.2 Vincoli

In generale un database è composto di più schemi di relazione

**Def** Uno schema di database è un insieme di schemi di relazione  $DB = \{R_1(X_1), \dots, R_n(X_n)\}$

**Def** Una istanza di un database su uno schema DB è l'insieme delle istanze di relazione  $db = \{r_1(R_1), \dots, r_n(R_n)\}$

Un vincolo di integrità è una condizione espresso nello schema di database e deve essere soddisfatto dalle istanze di database.

Ogni vincolo è implementato da un predicato booleano che ritorna true se l'istanza di database rispetta il vincolo, false altrimenti.

I vincoli possono essere raggruppati in:

- Vincoli inerenti al modello relazionale (SOLAMENTE "non ci possono essere due tuple uguali")
- Vincoli intra-relazioni
- Vincoli inter-relazioni

### 2.2.1 Vincoli Intra-Relazionali

Sono vincoli che vengono verificati su una singola istanza di uno schema di relazione.

- **Vincolo di dominio** (specifica il dominio dell'attributo A; es. l'attributo "voto" deve essere tra 18 e 30)
- **Vincolo di tuple** (un vincolo sui valori di una tupla di valori es. Lode può essere true solo se Voto è 30)
- **Vincolo di chiave** (un certo insieme di attributi deve essere una chiave)
- **Vincolo di chiave primaria** (un vincolo di chiave che non permette i valori NULL)

**Def** Sia  $R(T)$  uno schema di relazione,  $r(R)$  una sua istanza e  $SK$  un sottosinsieme di attributi.  $SK$  è una superchiave di  $R$  se

$$\forall t_1, t_2 \in r, t_1 \neq t_2 \Rightarrow t_1[SK] \neq t_2[SK]$$

**Def** Una chiave  $K$  è una superchiave minima, non è possibile togliere alcun attributo senza perdere l'univocità.

**Oss** Le superchiavi (o chiavi) dovrebbero essere sempre significative nel mini-mondo di interesse.

**Def** La chiave primaria è una chiave i cui attributi non possono essere NULL, le altre chiavi sono delle candidate.

**Def** Un attributo si dice primo se è membro di una chiave candidata.



### 2.2.2 Vincoli Inter-Relazionali

Esiste un vincolo di grande importanza che va verificato tra due relazioni, il vincolo di integrità referenziale.

Questo vincolo viene verificato se sono verificati i vincoli di chiave esterna.

**Def** Un insieme di attributi  $FK = (A_1, \dots, A_n)$  di uno schema  $R$ , è una chiave esterna se esiste uno schema  $R_2$  con chiave primaria  $PK = (B_1, \dots, B_n)$  e se  $\text{dom}(A_i) = \text{dom}(B_i) \quad \forall i=1, \dots, n$ ,  $\forall t \in r(R), t_i[FK] = \text{NULL} \vee \exists t_2 \in r_2 | t_i[FK] = t_2[PK]$

### 2.3 Traduzione di un Modello ER in un Modello Relazionale

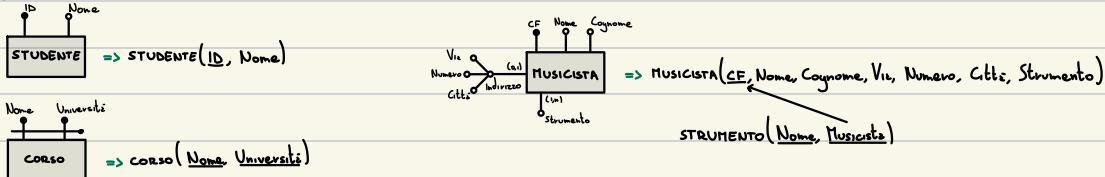
Il processo di traduzione è automatico, la maggior parte delle scelte sono già state fatte, è sufficiente applicare le regole.

Si noti che queste regole portano già ad una forma ottima, non sarà necessario utilizzare il processo di normalizzazione che vedremo in seguito.

#### 2.3.1 Traduzione di un'Entità Forte

- Si converte l'entità E in una relazione  $R_E$  dello stesso nome
- Gli attributi di  $R_E$  sono tutti gli attributi di E
- l'attributo che identifica E diventa chiave primaria in  $R_E$

ES



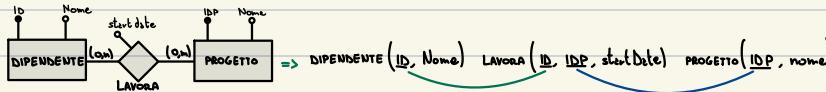
## 2.3.2 Traduzione di Associazioni Binarie

La soluzione generica  $(x_1, n) - (y_1, m)$  richiede di:

- Convertire l'associazione Q in una relazione R<sub>0</sub>
- Gli attributi di R<sub>0</sub> sono tutti gli attributi di Q, più tutte le chiavi primarie delle associazioni di Q.
- La chiave primaria è la chiave di una delle relazioni con cardinalità massima 1.

Se non ne esiste nessuna allora tutte le chiavi primarie delle entità formano la chiave primaria di R<sub>0</sub>

ES



Nel caso più specifico  $(x_1, 1) - (y_1, m)$  oppure  $(x_1, 1) - (y_1, 1)$  è più efficiente non utilizzare la relazione R<sub>0</sub>.

Si integrano nell'entità con cardinalità  $(x_1, 1)$  oppure libera scelta tra  $(x_1, 1)$  e  $(y_1, 1)$  gli attributi di Q e la chiave primaria dell'altra entità.

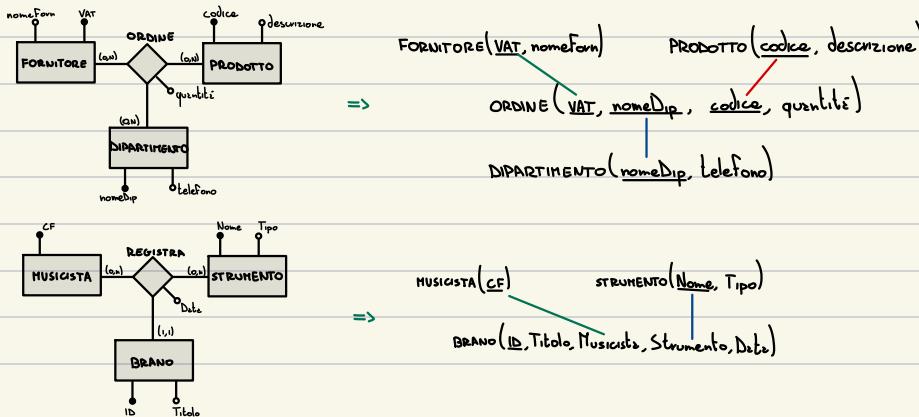
ES



## 2.3.3 Traduzione di Associazioni Ternarie

Sono del tutto simili alle associazioni binarie

ES



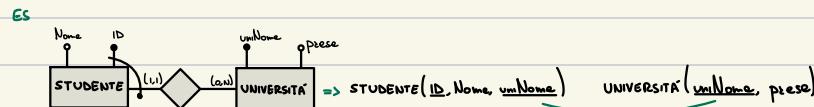
### 2.3.3 Traduzione di Associazioni Ricorsive

Anche in questo caso la traduzione risulta simile alle associazioni binarie.



### 2.3.4 Traduzione di un'Entità Debole

La traduzione di un'entità debole è simile alla traduzione di un'associazione con cardinalità (1,1)



## 2.4 Ottimizzazione del Modello Relazionale

Guideline 1: Lo schema relazionale dovrebbe essere facilmente comprensibile, non vanno combinati attributi di diverse entità o relazioni.

Guideline 2: Non ci dovrebbero essere anomie nell'inserimento, aggiornamento e rimozione dei dati del database.

Se non possono essere evitate vanno innestate e i programmi devono tenerne conto nelle query.

Guideline 3: I valori NULL vanno tenuti al minimo, bisogna evitare di inserire attributi che sono frequentemente NULL.

Guideline 4: Faccendo una operazione di JOIN che utilizza chiavi primarie o chiavi esterne non si dovrebbero generare tuple incorrecte (spur).

### 2.5.1 Normalizzazione

La normalizzazione è il processo di analisi dello schema relazionale per minimizzare la duplicazione, le anomalie di aggiornamento e le tuple incorrecte.



1 NF. Il dominio degli attributi deve includere solo elementi atomici (semplici, indivisibili) e il valore di ogni attributo è un singolo valore.

Si può risolvere separando le informazioni o su più attributi o su una nuova entità.

**Def** Date una relazione R e due insiemi di attributi X e Y, una dipendenza funzionale  $X \rightarrow Y$  è un vincolo:

$$\forall t_1, t_2 \in r | t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$$

**Prop**  $Y \subseteq X \Rightarrow X \rightarrow Y \quad X \rightarrow Y \Rightarrow |X, Z| \rightarrow |Y, Z| \quad X \rightarrow Y, Y \rightarrow Z \Rightarrow X \rightarrow Z$

**Es** matricola  $\rightarrow$  nome

**Def** Una dipendenza funzionale si dice completa se la rimozione di un attributo A da X rende invalida la dipendenza.

Altrimenti si dice dipendenza parziale.

**2 NF**: Ogni attributo non primo (non membro di una chiave candidata) A in R è dipendenza funzionale completa

d'ogni chiave di R



**Def** Una dipendenza funzionale  $X \rightarrow Y$  si dice dipendenza transitiva se esiste un insieme di attributi Z che non è una chiave candidata o un sottoinsieme di una chiave primaria e  $X \rightarrow Z$  e  $Z \rightarrow Y$  sono veri.

**Es** matricola  $\xrightarrow{X}$  Dipartimento  $\xrightarrow{Z}$  NomeResponsabile  $\xrightarrow{Y}$

**3NF**: Se nessun attributo non primo è una dipendenza transitiva di qualsiasi chiave di R.

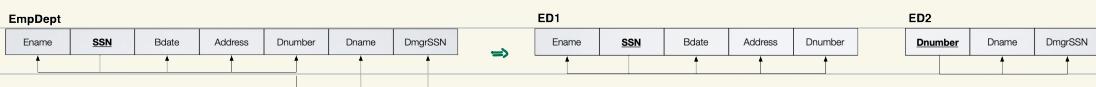
**ES** STUDENTE (matricola, Dipartimento, NomeResponsabile)  $\Rightarrow$  STUDENTE (matricola, Dipartimento)  $\xrightarrow{\text{DIPARTIMENTO}}$  (Nome, NomeResponsabile)

**Def** Una dipendenza funzionale  $X \rightarrow Y$  si dice triviale se  $Y \subseteq X$ , altrimenti è non triviale

**3NF-Alternativa**: Per ogni dipendenza funzionale non triviale  $X \rightarrow Y$  verificate una delle due condizioni è verificata:

A. X è superchiave di R

B. Y è un attributo primo (membro di una chiave candidata) di R



**Boyce-Codd Normal Form (BCNF)**: Per ogni dipendenza funzionale non triviale  $X \rightarrow A$  allora X è superchiave di R.

**Oss** BCNF è più stretta di 3NF, manca la condizione B.

### 3. Algebra Relazionale

L'algebra relazionale è un linguaggio procedurale (diverso dai più comuni linguaggi dichiarativi dove si specifica il risultato ignorando le procedure utilizzate per raggiungerlo).

Il linguaggio offre una serie di operatori il cui risultato può essere utilizzato come input di un'altra operazione.

#### 3.1 Operazioni di Base

##### 3.1.1 Selezione

Selezione delle righe di una tabella in base ad un criterio.

Indicate con  $\sigma_F(r(X))$  dove  $r(X)$  è un'istanza di uno schema relazionale,  $X$  il suo insieme di attributi

$F$  è la formula proposizionale (definisce come filtrare le righe)

ES

	Nome	Cognome	CF
t <sub>1</sub>	Giulio	Rossi	0
t <sub>2</sub>	Giovanni	Bianchi	1
t <sub>3</sub>	Matteo	Vichi	2

$$\sigma_{Nome = Matteo}(\text{PERSONE}) \rightarrow$$

	Nome	Cognome	CF
t <sub>3</sub>	Matteo	Vichi	2

##### 3.1.2 Proiezione

Selezione delle colonne di una tabella.

Indicate con  $\pi_Y(r(X))$  dove  $Y \subseteq X$

ES

	Nome	Cognome	CF
t <sub>1</sub>	Giulio	Rossi	0
t <sub>2</sub>	Giovanni	Bianchi	1
t <sub>3</sub>	Matteo	Vichi	2

$$\pi_{CF}(\text{PERSONE}) \rightarrow$$

	CF
t <sub>1</sub>	0
t <sub>2</sub>	1
t <sub>3</sub>	2

##### 3.1.3 Ridenominazione

Rinominare uno schema o i suoi attributi può essere utile in alcune situazioni, in particolare per dare un nome all'output di un'operazione.

Si indica con  $\rho_{R_1(B_1, \dots, B_n)}(R_1(A_1, \dots, A_n))$

ES

$$\rho_{\text{PEOPLE}(CF, Nome)}(\text{PERSONE}(CF, Nome))$$

$$\rho_{\text{PEOPLE}}(\text{PERSONE}(CF, Nome))$$

$$\rho(CF, Nome)(\text{PERSONE}(CF, Nome))$$

#### 3.2 Operazioni Insiemistiche

Le operazioni insiemistiche sono definite su insiemi di tuple che devono essere tutte dello stesso grado (con lo stesso numero di attributi) e gli attributi devono essere dello stesso tipo.

Formalmente, date due relazioni  $R(A_1, A_2, \dots)$  e  $S(B_1, B_2, \dots)$  è possibile definire le operazioni insiemistiche

se  $R$  e  $S$  sono compatibili ovvero se:

- se il grado delle due relazioni è lo stesso
- se  $\text{dom}(A_i) = \text{dom}(B_i)$ ,  $i=1, \dots, n$

### 3.2.1 Unione

$$r_1(X) \cup r_2(X) = \{t \mid t \in r_1(X) \text{ OR } t \in r_2(X)\}$$

### 3.2.2 Differenza

$$r_1(X) \setminus r_2(X) = \{t \mid t \in r_1(X) \text{ AND } t \notin r_2(X)\}$$

### 3.2.3 Intersezione

$$r_1(X) \cap r_2(X) = \{t \mid t \in r_1(X) \text{ AND } t \in r_2(X)\}$$

## 3.3 Operazioni di Correlazione

Le operazioni più importanti dell'algebra relazionale sono quelle di correlazioni, ovvero le JOIN.

### 3.3.1 Prodotto Cartesiano

$$r_1(X) \times r_2(Y) = \{t_S \mid t \in r_1(X) \text{ AND } t \in r_2(Y)\}$$

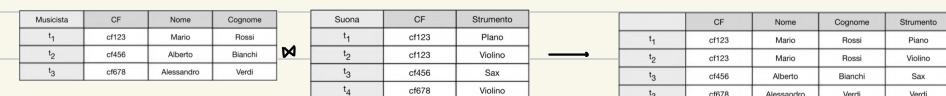
dove  $X$  e  $Y$  sono disjunti ( $X \cap Y = \emptyset$ ) altrimenti non sarebbe possibile distinguere  $A \in X$  da  $A \in Y$  (dove  $A \in X \cap Y$ )

La relazione risultante ha:

- grado pari alla somma dei gradi delle tabelle ( $|X| + |Y|$ )
- cardinalità pari al prodotto delle cardinalità ( $|r_1(X)| \cdot |r_2(Y)|$ )

### 3.3.2 Join Naturale

Il join naturale ( $\bowtie$ ) cerca attributi comuni alle due tabelle e unisce le tuple che hanno tali attributi con valori uguali



$$r_1(X) \bowtie r_2(Y) = \{t \mid t \in r_1(X) \text{ AND } t \in r_2(Y)\}$$

La relazione risultante ha:

- come grado  $|X \cup Y|$  ( $|X| + |Y| - |X \cap Y|$ )
- cardinalità che varia a seconda dei casi  $\leq |r_1(X) \bowtie r_2(Y)| \leq |r_1(X)| \cdot |r_2(Y)|$

$\leq$  se non ci sono tuple con gli attributi comuni uguali  
 $\leq$  se  $X \cap Y$  è chiave primaria di  $R_1(X)$  e chiave esterna di  $R_2(Y)$

OSS

- Se  $X \bowtie Y = \emptyset$  l'operazione di join naturale è equivalente al prodotto cartesiano.
- Se  $X = Y$  l'operazione di join naturale è equivalente ad un'operazione di intersezione.

### 3.3.3 Theta Join

$$r_1(X) \bowtie_{\varphi} r_2(Y) = \sigma_{\varphi}(r_1(X) \times r_2(Y))$$

Questo JOIN ha delle importanti implicazioni prestazionali, fare il prodotto cartesiano è costoso

La relazione risultante ha:

- come grado  $|X| + |Y|$
- cardinalità che varia a seconda dei casi  $0 \leq |r_1(X) \bowtie_{\varphi} r_2(Y)| \leq |r_1(X)| \times |r_2(Y)|$

### 3.3.4 Equi Join

Una theta-join in cui le condizioni sono tutte di uguaglianza.

**Def** Un join si dice completo se tutte le tuple delle due tabelle partecipano al risultato.

**Def** Un join si dice incompleto se non è completo. Le tuple non utilizzate si dicono pendenti (dangling).

Se si vogliono far apparire anche le tuple pendenti è possibile usare le join esterne, ne esistono di 3 tipi:

- left join  $r_1(X) \bowtie r_2(Y)$
- right join  $r_1(X) \bowtie r_2(Y)$
- outer join  $r_1(X) \bowtie r_2(Y)$

	CF	Nome	Figli
t <sub>1</sub>	0	Giulio	7
t <sub>2</sub>	1	Giovanni	2
t <sub>3</sub>	NULL	NULL	3

	CF	Nome	Figli
t <sub>1</sub>	0	Giulio	7
t <sub>2</sub>	1	Giovanni	2
t <sub>3</sub>	2	Mario	NULL

	CF	Nome	Figli
t <sub>1</sub>	0	Giulio	7
t <sub>2</sub>	1	Giovanni	2
t <sub>3</sub>	2	Mario	NULL
t <sub>4</sub>	NULL	NULL	3

## 3.4 Operazioni di Aggregazione e Raggruppamento

Un'estensione del modello relazionale che permette l'applicazione di funzioni di aggregazione e raggruppamento.

### 3.4.1 Proiezione Estesa

CONCERTO	Data	Teatro	Prezzo	Biglietti venduti
t <sub>1</sub>	2015/03/28	Verdi	20	120
t <sub>2</sub>	2015/07/03	Fenice	50	200
t <sub>3</sub>	2015/07/30	Olimpico	45	150

$\Pi_{Data, Teatro, Prezzo, Biglietti.} (MUSICISTA)$

	Data	Teatro	
t <sub>1</sub>	2015/03/28	Verdi	4800
t <sub>2</sub>	2015/07/03	Fenice	10000
t <sub>3</sub>	2015/07/30	Olimpico	6750

### 3.4.2 Operazione di Aggregazione

Permettono di svolgere delle operazioni su un'attributo della relazione: SUM, MAX, MIN, AVERAGE, COUNT

La notazione è la seguente  $f_{F_1(A_1), \dots, F_n(A_n)}(R(x))$

ES

$f_{\text{sum(Ricavo)}, \text{average(Ricavo)}} (MUSICISTA)$

### 3.4.2 Operazione di Raggruppamento

Permette di effettuare le operazioni di aggregazione su sottosinsiemi di tuple.

	Nome	Ricavo
$t_1$	R1	200
$t_2$	R2	1800
$t_3$	R2	2100
$t_4$	R1	1500
$t_5$	R3	4200

(come)  $\sum_{\text{Ricavo}} \text{AVERAGE}(\text{Ricavo})$  (incasso)

	Nome	SUM	AVERAGE
$t_1$	R1	2300	1150
$t_2$	R2	3900	1950
$t_5$	R3	4200	4200

### 3.5 Valori Nulli nell'Algebra Relazionale

Vediamo come la presenza del valore di NULL (già introdotto in precedenza) venga gestita dall'algebra relazionale.

#### 3.5.1 Logica a Tre Valori

In presenza del valore NULL modifica il risultato delle operazioni booleane

AND	T	F	U	OR	T	F	U	NOT	T	F
T	T	F	U	T	T	T	T	F	T	F
F	F	F	U	F	T	F	U	T	F	T
U	V	F	U	U	T	U	U	V	U	U

Una serie di regole da applicare in ordine

$$\begin{aligned} T \text{ AND } T &\rightarrow T \\ F \text{ AND } ... &\rightarrow F \\ U \text{ AND } ... &\rightarrow U \end{aligned}$$

$$\begin{aligned} T \text{ OR } ... &\rightarrow T \\ F \text{ OR } F &\rightarrow F \\ U \text{ OR } ... &\rightarrow U \end{aligned}$$

#### 3.5.2 Variazioni delle Operazioni

Le operazioni di base e le operazioni insiemistiche si comportano come se i valori NULL fossero confrontabili.

Il join naturale e il theta join non combinano due tuple se hanno come valore NULL su uno degli attributi.

Le operazioni di aggregazione restano invariate, tranne le funzione COUNT non conta le tuple con valore NULL