

LE BASI DI DATI RELAZIONALI – PROGETTAZIONE LOGICA

Come abbiamo già detto, la progettazione logica è la fase di progettazione che, a partire dal progetto concettuale (diagramma E/R più eventuali altri documenti che lo spieghino) porta al progetto logico. Come detto, il progetto logico consente di modellizzare la realtà in una forma più “vicina” ai sistemi informatici rispetto al diagramma E/R, e quindi più facilmente implementabile nella successiva progettazione fisica. Sostanzialmente la progettazione logica consente di ottenere delle tabelle.

In particolare, la progettazione logica di nostro interesse è quella che utilizza il **modello logico relazionale** delle basi di dati.

Il modello logico relazionale è stato ideato dal ricercatore della IBM Edgard Codd (1923 -2003) (per la precisione **Edgard Frank “Tedd” Codd**) che ha esposto tale modello in un articolo pubblicato nel 1970 dal titolo “**Modello relazionale per basi di dati condivise**”. In questo modello la realtà viene modellizzata utilizzando il concetto matematico di **relazione**, rappresentabile con delle tabelle.

Per fare un po’ di chiarezza è quindi necessario spiegare **cosa è una relazione** (dal punto di vista matematico) e perché si può rappresentare con una tabella. Poi vedremo che, con semplici regole di derivazione, sarà possibile passare dal diagramma E/R alle tabelle.

Cosa è una relazione? (Il modello logico relazionale)

Definizione di prodotto cartesiano.

Per capirlo bisogna definire cosa è il **prodotto cartesiano** fra **n** insiemi.

Ipotizziamo di avere n insiemi chiamati D_1, D_2, \dots, D_n .

Prodotto cartesiano fra n insiemi: insieme di tutte le possibili **ennuple** di elementi **ordinati** (d_1, d_2, \dots, d_n) tali che d_1 appartiene a D_1 , d_2 appartiene a D_2, \dots, d_n appartiene a D_n . **In altre parole il primo elemento della n-upla deve appartenere al primo insieme, il secondo elemento al secondo insieme ecc....**

(ennuple sta per coppie nel caso di $n=2$, sta per terne nel caso di $n=3$, quadruple per $n=4...$)

ESEMPIO: con $n=2$ il prodotto cartesiano è l'insieme di tutte le possibili **coppie** di elementi tali che il primo elemento appartiene all'insieme1 e il secondo elemento appartiene all'insieme2

insieme 1 → MODELLO {Panda, Multipla, C2, C3 }

insieme 2 → COSTRUTTORE { Fiat, Citroen }

il prodotto cartesiano MODELLO X COSTRUTTORE (si legge: *modello **cartesiano** costruttore*) è il seguente insieme:

MODELLO X COSTRUTTORE = { (Panda, Fiat), (Panda, Citroen), (Multipla, Fiat), (Multipla, Citroen), (C2, Fiat), (C2, Citroen), (C3, Fiat), (C3, Citroen) }

dove ogni parentesi tonda è una ennupla (in questo caso una coppia) del prodotto cartesiano.

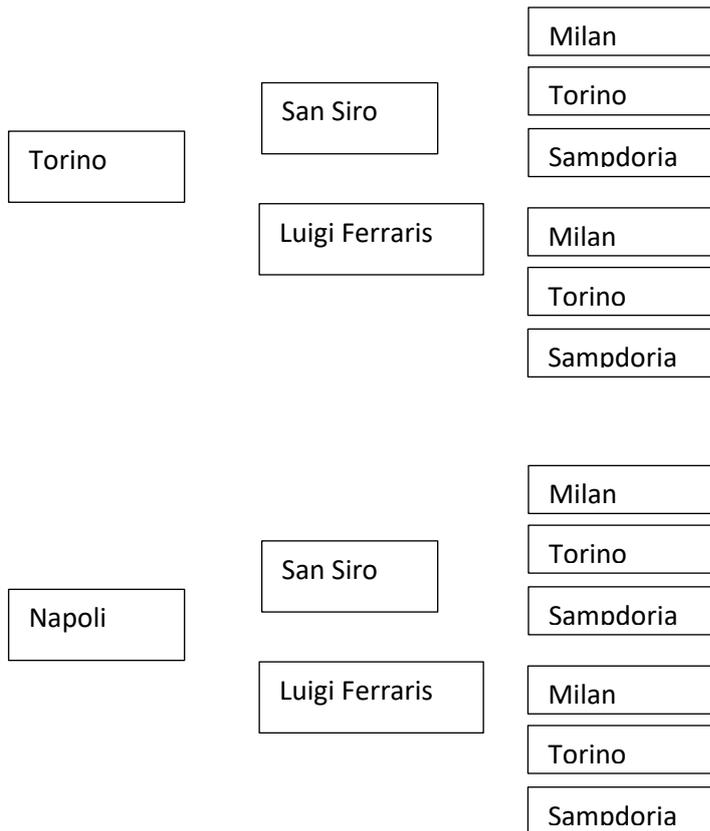
Esercizio: determinare il prodotto cartesiano di questi 3 insiemi:

insieme 1 → CITTA' {Milano, Torino, Napoli}

insieme 2 → STADIO { San Siro, Luigi Ferraris }

insieme 3 → SQUADRA {Milan, Torino, Sampdoria}





Esercizio. Scrivere l'insieme

Definizione di relazione

Una relazione fra n insiemi è un sottoinsieme del prodotto cartesiano fra gli insiemi stessi, ossia un insieme che contiene alcune (al limite tutte) le n-uple del prodotto cartesiano.

Nel nostro esempio, infatti, se consideriamo il seguente sottoinsieme di MODELLO X COSTRUTTORE:

$$\{(Panda, Fiat), (Multipla, Fiat), (C2, Citroen), (C3, Citroen)\}$$

Si può notare come esso rappresenti una **relazione** fra gli elementi dei due insiemi di partenza che può essere descritta con la frase “*ProdottoDa*”.

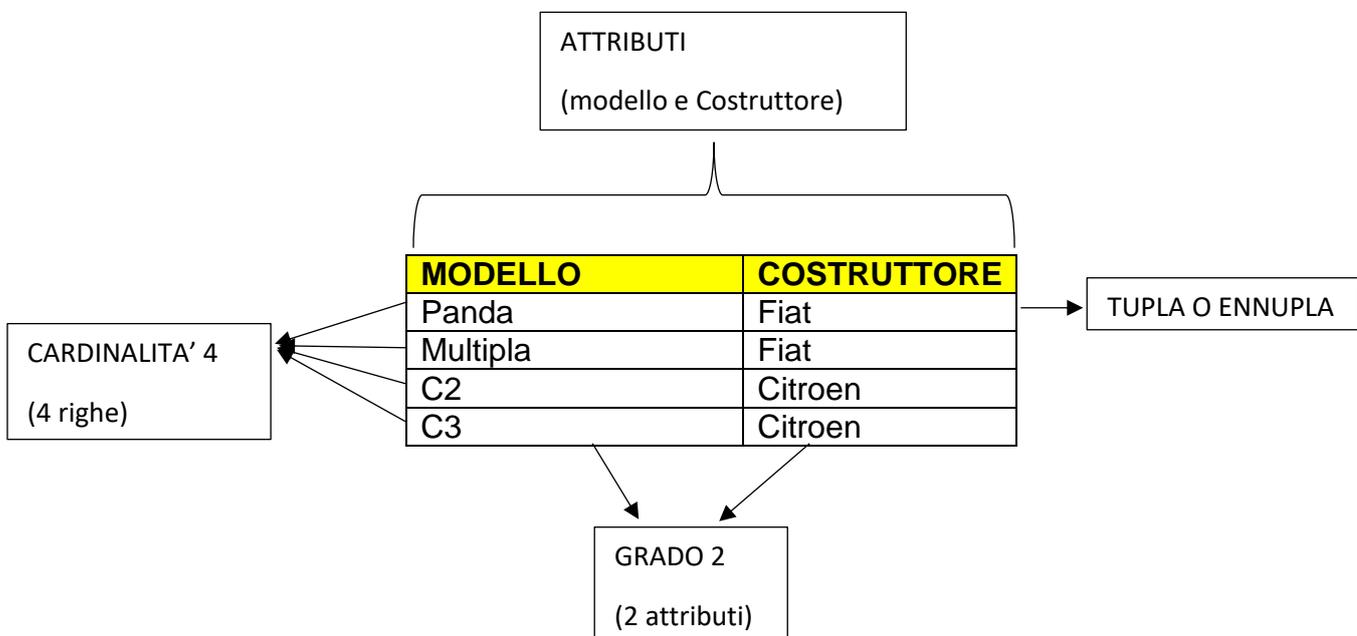
Tornando all'esempio, la relazione "ProdottoDa" si può rappresentare nel seguente modo:

MODELLO	COSTRUTTORE
Panda	Fiat
Multipla	Fiat
C2	Citroen
C3	Citroen

- Una **relazione** può quindi essere rappresentata con una **tabella**.
- Il **grado** della relazione (n) è individuato dal numero di **colonne** della tabella
- Il **nome di ciascuna colonna** rappresenta un **attributo** della relazione
- L'insieme dei valori che può assumere un attributo è detto **dominio** dell'attributo, l'insieme di tutti i domini è chiamato dominio della relazione.
- Il numero di **n-uple** (chiamate anche **tuple**) che compongono la tabella, ossia delle righe, è chiamato **cardinalità** della relazione.

Nell'esempio:

La relazione *ProdottoDa* è una relazione di grado 2 e cardinalità 4. I suoi attributi sono: Modello e Costruttore. Il dominio degli attributi è formato dagli elementi dei due insiemi di partenza: (Panda, Multipla, C3, C4) e (Fiat, Citroen).



ALTRO ESEMPIO DI RELAZIONE: relazione Automobili

Automobili

MODELLO	COSTRUTTORE	CATEGORIA	POSTI
Focus	Ford	C	5
Panda	Fiat	B	4
147	Alfa Romeo	C	5
Multipia	Fiat	C	6
C2	Citroen	B	4
C3	Citroen	C	5
Punto	Fiat	B	5

Data la somiglianza con la struttura di un file di dati, spesso si utilizzano i seguenti termini come **sinonimi**:

Tupla → Record

Attributo → Campo

Oltre a: Relazione → Tabella

Nella tabella si osservi che i nomi delle colonne forniscono l'aspetto **intensionale** dei dati rappresentati, mentre i valori contenuti nelle righe forniscono l'aspetto **estensionale**.

Un altro modo per rappresentare l'aspetto **intensionale** dei dati è quello di indicare l'insieme degli attributi della relazione. Questo insieme è chiamato **schema** della relazione. Nel nostro esempio lo **schema** della relazione automobili è il seguente:

Automobili: (Modello, Costruttore, Categoria, Posti)

L'attributo sottolineato indica la Chiave Primaria.

Chiave Primaria: insieme minimale di attributi che identifica univocamente una n-upla (tupla) della relazione.

Una chiave primaria è quindi un campo, o un insieme di campi, che ha/hanno un valore univoco per ogni riga di una tabella (es. il codice fiscale per una tabella *persona*, il numero di matricola per una tabella *studenti*...)

Requisiti fondamentali che caratterizzano il modello relazionale:

- 1) Tutte le righe contengono lo stesso numero di colonne
- 2) Gli attributi rappresentano informazioni elementari (**atomiche**) non scomponibili ulteriormente
- 3) I valori appartenenti ad un campo sono tutti dello stesso tipo (**omogenei**) poiché appartengono al dominio di quel campo
- 4) Non ci possono essere due o più righe che hanno gli stessi valori in tutti campi, questo implica che debba sempre esserci una **chiave primaria** che identifica univocamente una tupla, inoltre la chiave primaria non deve mai avere valore nullo.
- 5) Le tuple appaiono nella tabella secondo un ordine non prefissato ed irrilevante

Derivazione delle relazioni dal modello E/R

Il passaggio dal modello concettuale (diagramma E/R) al modello logico relazionale (tabelle) avviene secondo le seguenti regole di derivazione:

1. Ogni *entità* diventa una *relazione*.
2. Gli *attributi* dell'entità diventano gli *attributi* della relazione.
3. Ogni attributo della relazione eredita le caratteristiche dagli attributi dell'entità.
4. L'*identificatore univoco* dell'entità diventa la *chiave primaria* della relazione.
5. L'*associazione uno a uno* viene rappresentata aggiungendo ad una delle due relazioni (a scelta) la chiave primaria dell'altra relazione e gli eventuali attributi della associazione. Tale attributo (insieme di attributi) aggiunto prende il nome di **chiave esterna (FK, foreign key)**.
6. L'*associazione uno a molti* viene rappresentata aggiungendo alla relazione che rappresenta l'entità che svolge il **ruolo "a molti"** la chiave primaria dell'entità che svolge il ruolo "a uno". Tale attributo (insieme di attributi) prende il nome di **chiave esterna (FK, foreign key)**. Inoltre si aggiungono alla relazione che rappresenta l'entità che svolge il ruolo "**a molti**" gli eventuali attributi della associazione.
7. L'*associazione molti a molti* viene rappresentata mediante una **ulteriore relazione** che contiene gli identificatori univoci delle due entità e gli eventuali attributi dell'associazione. La chiave primaria della nuova relazione è composta dalle chiavi primarie delle due entità oltre agli eventuali attributi dell'associazione necessari a rendere univoche le tuple della relazione ottenuta.

Osservazione.: le relazioni che derivano dalle entità vengono rappresentate convertendo al plurale il nome dell'entità per indicare che tale relazione contiene le istanze dell'entità

Negli esercizi completi descriveremo sempre le relazioni ottenute dal diagramma E/R indicando:

Il nome della relazione

Una breve descrizione della relazione

Lo schema della relazione in forma tabellare dove verranno indicati, per ogni campo:

- **il nome**
- **se è chiave primaria (PK) o chiave esterna (FK)**
- **il tipo di dato**
- **la dimensione del dato (il numero di caratteri nel caso di dati di tipo carattere).**
- **se il dato è obbligatorio (non può assumere valore null)**

In fine, sotto la tabella, verranno indicate tutte le chiavi (primarie ed esterne) con l'indicazione, per le chiavi esterne, delle tabelle e dei campi da cui derivano (con la *dot notation*; tabella.campo)

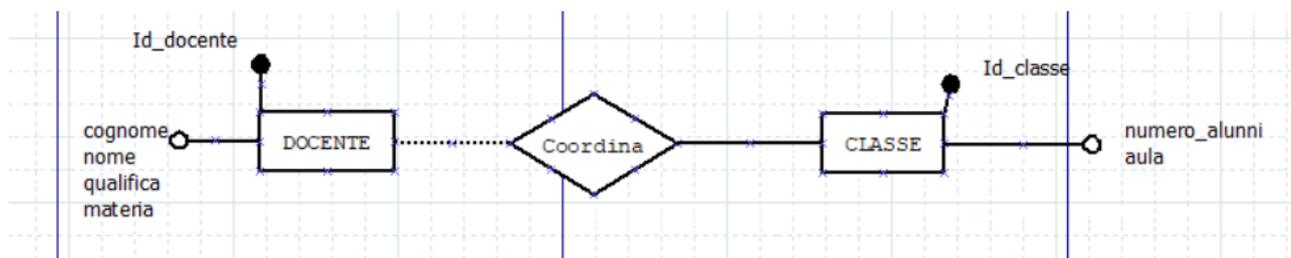
Stabiliamo inoltre di indicare tutti gli attributi con lettere minuscole e di indicare:

- **i nomi composti da più parole utilizzando il carattere underscore (es. data_nascita)**
- **gli identificativi aggiunti da noi e non specificati nel testo con: "id_tal_dei_tali".**

Esempi:

1. associazione 1 a 1.

Ipotizziamo una scuola dove vige la regola che un docente può coordinare al massimo una sola classe



Applicando la regola 5 si può scegliere se aggiungere la chiave esterna alla relazione Docenti oppure la chiave esterna alla relazione Classi. Visto che aggiungendola alla relazione Docenti ci sarebbero molte righe con chiave esterna NULL poiché molti docenti non sono coordinatori, è preferibile aggiungere la chiave esterna alla tabella Classi.

Tenendo presente che in seguito tutte queste tabella verranno realizzate fisicamente su una memoria di massa, la scelta di metter la chiave esterna alla tabella Classi ha dei vantaggi in termini di prestazioni, infatti se avessimo messo la FK *codice_classe_coordinata* alla tabella Docenti:

Ogni volta che il DBMS deve ricercare (o comunque gestire) dei Docenti, avrebbe sempre un campo in più da gestire (la FK *codice_classe_coordinata*) spesso con valore null

In generale, quando uno dei due versi della associazione è facoltativo e uno obbligatorio si preferisce aggiungere la chiave alla relazione che partecipa alla associazione in modo obbligatorio.

DB schema:

Docenti (id_docente, cognome, nome, qualifica, materia)

Classi (id_classe, numero_alunni, aula, *id_coordinatore*) FK = id_coordinatore

Tabelle:

Docenti

Tabella che contiene i dati anagrafici dei docenti

Nome	Chiave	Tipo	Dimensione	Not null
id_docente	PK	numerico		X
cognome		carattere	30	X
nome		carattere	30	X
qualifica		carattere	30	X
materia		carattere	30	X

Chiave primaria: id_docente

Classi

Tabella che contiene le informazioni su ciascuna classe della scuola

Nome	Chiave	Tipo	Dimensione	Not null
id_classe	PK	carattere	5	X
numero_alunni		numerico		X
aula		numerico		X
id_coordinatore	FK	numerico		X

Chiave primaria: id_classe

Chiave esterna: coordinatore → Docenti.id_docente

devono essere dello stesso tipo

Per esemplificare mostriamo una rappresentazione estensionale dei possibili dati

Docenti (esempio estensionale)

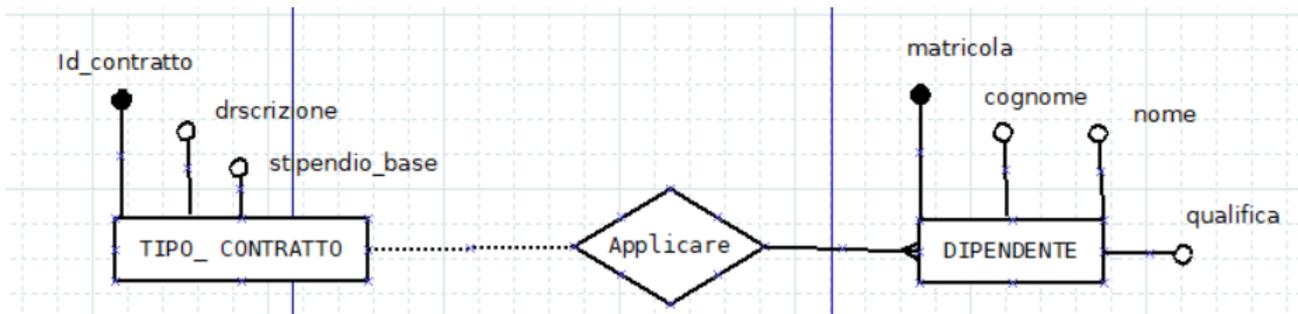
id_docente	Cognome	nome	qualifica	materia
1	Delvecchio	Anna	R	religione
2	Trotti	Domenica	R	inglese
3	Poiatti	Elena	S	sistemi
4	Labozzetta	Pietro	R	lab info
5	Buonanno	Antonio	R	ed fisica
6	Pierone	Lucia	S	lab sistemi
7	Giuliano	Lauro	S	economia
8	Pinna	Laura	R	inglese

Classi (esempio estensionale)

codice_classe	numero_alunni	aula	id_coordinatore
1AINT	21	4	2
2AINT	22	10	8

Nel caso l'associazione fosse facoltativa in entrambe le direzioni, la chiave esterna si potrebbe aggiungere indifferentemente ad una delle due relazioni a piacere.

2. Tipi_Contratto / Dipendenti (associazione 1 a molti che diventa due relazioni)



DB schema:

Tipi_contratto (id_contratto, descrizione, stipendio_base)

Dipendenti (matricola, cognome, nome, qualifica, *tipo_contratto*)(FK in corsivo)

Tabelle

Tipi_contratto

Tabella che contiene informazioni sulle varie tipologie di contratto

Nome	Chiave	Tipo	Dimensione	Not null
id_contratto	PK	numerico		X
Descrizione		carattere	50	X
stipendio_base		numerico		X

chiave primaria: id_contratto

Dipendente

Nome	Chiave	Tipo	Dimensione	Not null
Matricola	PK	carattere	4	X
Cognome		carattere	30	X
Nome		carattere	30	X
qualifica		carattere	30	X
tipo_contratto	FK	numerico		X

Chiave primaria: matricola

Chiave esterna: tipo_contratto → Tipi_contratto.id_contratto

Esempi estensionali:

Tipo_contratti:

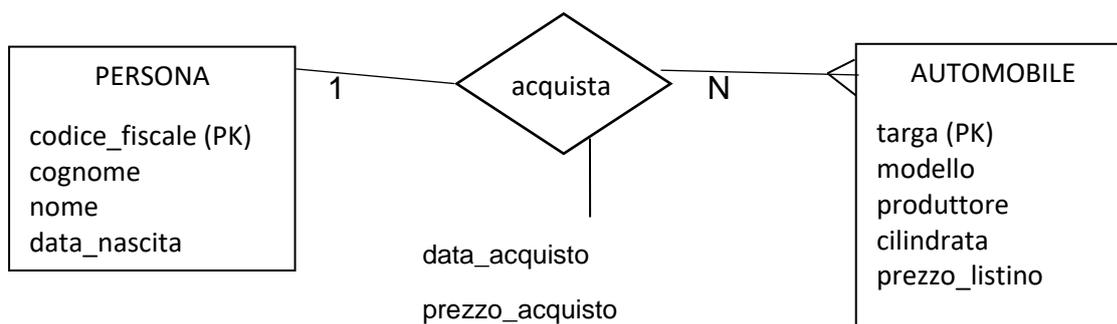
id_contratto	descrizione	stipendio_base
1	impiegato	1200
2	metalmeccanico	1300
3	dirigente	2100

Dipendenti:

matricola	cognome	nome	qualifica	tipo_contratto
i001	Pinna	Luca	addetto buste paghe	1
i002	Licci	Lauro	commerciale	1
m001	Sorpa	Mino	operaio specializzato	2
d001	Cilla	Laura	Responsabile personale	3

3. Persone / Automobili (associazione con attributi 1 a N che diventano due relazioni)

Realizzare lo schema concettuale e logico di clienti (CF, Cognome, nome, data di nascita) che acquistano delle automobili (produttore, modello, targa, cilindrata, prezzo di listino). Si vogliono memorizzare anche la data di acquisto e il prezzo d'acquisto che potrebbe essere diverso dal prezzo di listino (ad esempio per particolari sconti, incentivi, rottamazioni ecc).



DB schema:

Persona (codice_fiscale, cognome, nome, data_nascita)

Automobile (targa, modello, produttore, cilindrata, prezzo_listino, *codice_fiscale_cliente*, data_acquisto, prezzo_acquisto)

Tabelle:

Persone

Tabella contenente i dati anagrafici delle persone che acquistano le automobili

Nome	Chiave	Tipo	Dimensione	Not null
codice_fiscale	PK	carattere	16	X
cognome		carattere	30	X
Nome		carattere	30	X

data_nascita		data/ora		X
--------------	--	----------	--	---

Chiave primaria: codice_fiscale

Automobile

Tabella che contiene i dati relativi alle automobili acquistate

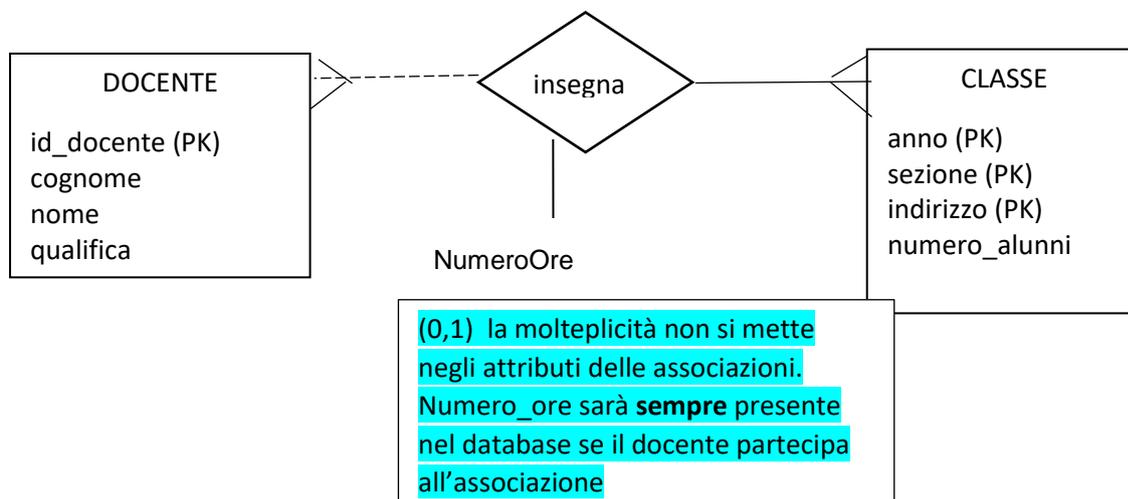
Nome	Chiave	Tipo	Dimensione	Not null
Targa	PK	carattere	7	X
modello		carattere	20	X
produttore		carattere	20	X
cilindrata		numerico		X
prezzo_listino		numerico		X
codice_fiscale_cliente	FK	carattere	16	X
data_acquisto		data		X
prezzo_acquisto		numerico		X

Chiave primaria: targa

Chiave esterna: codice_fiscale_cliente → Persone.codice_fiscale

4. Docenti / Classi (associazione con attributi N a N che diventano tre relazioni)

Rappresentare i docenti di una scuola e le classi in cui essi insegnano. Si vuole conoscere quali docenti insegnano in ogni classe e in quali classi insegna ogni docente. Alcuni insegnanti possono non insegnare (ad esempio gli insegnanti di sostegno). Si vuole conoscere quante ore di insegnamento ogni docente svolge in ogni classe. **Far fare esercizio (in matita!) e poi far apportare la modifica indicata nell'esercizio seguente e le modifiche in base alle osservazioni seguenti.**



Db schema:

Docenti (id_docente, cognome, nome, qualifica)

Classi (anno, sezione, indirizzo, numero_alunni,)

Insegnamento (id_docente, anno, sezione, indirizzo, numero_ore)

Tabelle:

Docenti

Tabella che contiene i dati anagrafici dei docenti

Nome	Chiave	Tipo	Dimensione	Not null
id_docente	PK	numerico		X
cognome		carattere	30	X
nome		carattere	30	X
qualifica		carattere	30	X

Chiave primaria: id_docente

Classi

Tabella che contiene le informazioni su ciascuna classe della scuola

Nome	Chiave	Tipo	Dimensione	Not null
anno	PK	numerico		X
sezione	PK	carattere	3	X
indirizzo	PK	carattere	3	X
numero_alunni		numerico		X

Chiave primaria: {anno, sezione, indirizzo}

Insegnamenti

Tabella che deriva dalla associazione Insegna fra Docente e Classe

Nome	Chiave	Tipo	Dimensione	Not null
docente	PK/FK	numerico		X
anno	PK/FK	numerico		X
sezione	PK/FK	carattere	3	X
indirizzo	PK/FK	carattere	3	X
numero_ore		numerico		X

Chiave primaria: docente, anno, sezione, indirizzo

Chiave esterna: docente → Docenti.id_docente,

classe → Classi.anno, Classi.sezione, Classi.indirizzo

Fornendo un esempio con l'aspetto estensionale dei dati si può apprezzare come da queste tre tabelle sia possibile risalire tutte le informazioni relative agli insegnamenti:

Docenti (esempio estensionale)

id_docente	cognome	nome	qualifica
1	Delvecchio	Anna	R
2	Trotti	Domenica	R
3	Poiatti	Elena	S
4	Labozzetta	Pietro	R
5	Buonanno	Antonio	R
6	Pierone	Lucia	S
7	Giuliano	Lauro	S
8	Pinna	Laura	R
9	Insegnante di	Sostegno	R

Classi (esempio estensionale)

anno	sezione	indirizzo	numero_alunni
1	A	INT	21
2	A	INT	22
3	A	INT	19
4	A	INT	22
5	A	INT	15

Insegnamenti (esempio estensionale)

docente	classe	sezione	indirizzo	numero_ore
2	1	A	INT	4
2	2	A	INT	4
2	4	A	INT	10
1	1	A	INT	1
1	2	A	INT	1
1	3	A	INT	1
1	4	A	INT	1
1	5	A	INT	1
6	2	A	INT	7
6	3	A	INT	7
6	5	A	INT	4
8	1	A	INT	5
8	2	A	INT	5
8	3	A	INT	4
8	4	A	INT	4
7	1	A	INT	3
7	2	A	INT	3
7	3	A	INT	3

Verificare che grazie al modello relazionale è possibile, ad esempio, rispondere alle seguenti domande:

Quali sono i docenti che insegnano nella classe 5AINT ?

Quali sono i docenti che NON insegnano nella classe 5A INT?

Quante ore insegna il docente Pinna?

Quali docenti insegnano meno di 18 ore?

Quanti docenti insegnano nella classe 1?

Quanti alunni ha il prof Pinna?

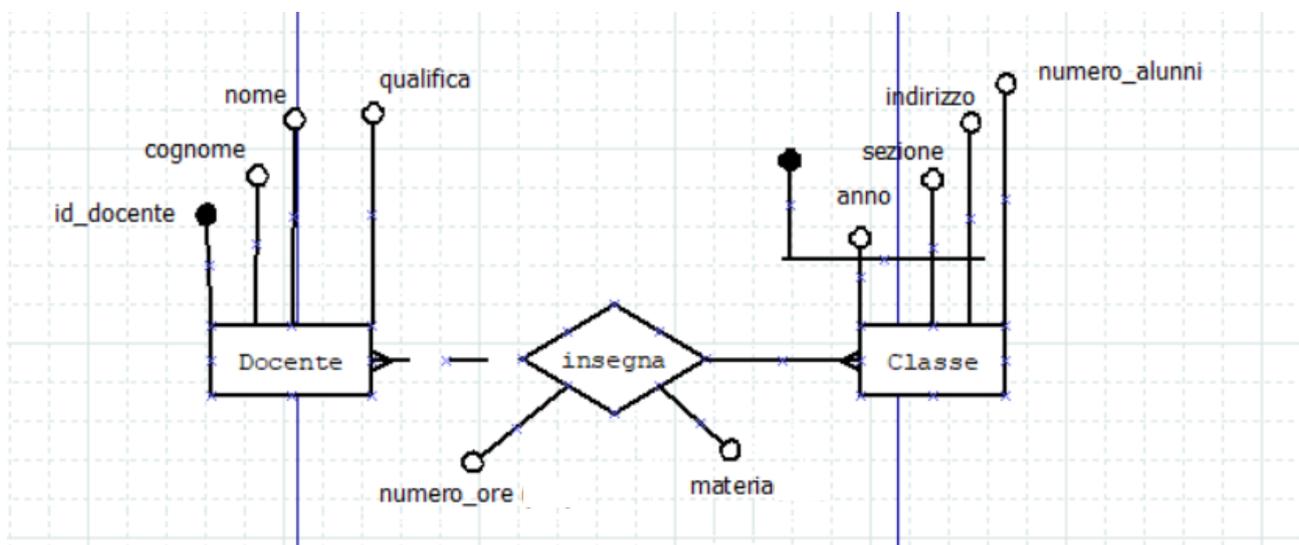
Ecc....

Quando realizzeremo queste tabelle con un DBMS avremo uno strumento (le query) che consentirà di rispondere a domande simili a queste ma in maniera più veloce.

ESERCIZIO (fare)

Supponiamo di voler memorizzare anche le materie che ciascun docente insegna nelle vari classi, si tenga presente che ogni docente può insegnare più materie sia nella stessa classe che in classi diverse.

Soluzione:



DB SCHEMA

Docenti (id_docente, cognome, nome, qualifica)

Classi (anno, sezione, indirizzo, numero_alunni)

Insegnamenti (docente, anno, sezione, indirizzo, materia, numero_ore)

TABELLE

Docenti

Tabella che contiene i dati anagrafici dei docenti

Nome	Chiave	Tipo	Dimensione	Not null
id_docente	PK	numerico		X
cognome		carattere	30	X
nome		carattere	30	X
qualifica		carattere	30	X

Chiave primaria: id_docente

Classi

Tabella che contiene le informazioni su ciascuna classe della scuola

Nome	Chiave	Tipo	Dimensione	Not null
anno	PK	numerico		X
sezione	PK	carattere	3	X
indirizzo	PK	carattere	3	X
numero_alunni		numerico		X

Chiave primaria: {anno, sezione, indirizzo}

Insegnamenti

Tabella che rappresenta gli insegnamenti dei docenti nelle classi.

Nome	Chiave	Tipo	Dimensione	Not null
docente	PK/FK	numerico		X
anno	PK/FK	numerico		X
sezione	PK/FK	carattere	3	X
indirizzo	PK/FK	carattere	3	X
materia	PK	carattere	50	X
numero_ore		numerico		X

Materia diventa PK poiché senza materia l'insieme (docente, anno, sezione, indirizzo) non è univoco

Chiave primaria: docente, anno, sezione, indirizzo, materia

Chiavi esterne: docente → Docenti.id_docente,

classe → Classi.anno, Classi.sezione, Classi.indirizzo

Fornendo un esempio con l'aspetto estensionale dei dati si può apprezzare come da queste quattro tabelle sia possibile risalire tutte le informazioni relative agli insegnamenti, anche relativamente alle materie insegnate.

Docenti (esempio estensionale)

id_docente	cognome	nome	qualifica
1	Delvecchio	Anna	R
2	Trotti	Domenica	R
3	Poiatti	Elena	S
4	Laini	Gian Marco	R
5	Buonanno	Antonio	R
6	Pierone	Lucia	S
7	Giuliano	Lauro	S
8	Pinna	Laura	R
9	Insegnante di	Sostegno	R

Classi (esempio estensionale)

anno	sezione	indirizzo	numero_alunni
1	A	INT	21
2	A	INT	22
3	A	INT	19
4	A	INT	22
5	A	INT	20
5	B	INT	17

Insegnamenti (esempio estensionale)

docente	classe	sezione	indirizzo	materia	numero_ore
2	1	A	INT	inglese	3
2	2	A	INT	inglese	3
2	4	A	INT	inglese	3
1	1	A	INT	religione	1
1	2	A	INT	religione	1
1	3	A	INT	religione	1
1	4	A	INT	religione	1
1	5	A	INT	religione	1
6	4	A	INT	Italiano	7
6	3	A	INT	Italiano	4
6	3	A	INT	Storia	3
8	1	A	INT	inglese	4
8	2	A	INT	inglese	4
8	3	A	INT	inglese	4
8	4	A	INT	inglese	4
7	1	A	INT	Storia	3
7	2	A	INT	Storia	3

3	3	A	INT	Informatica	6
3	4	A	INT	Informatica	6
4	5	A	INT	Informatica	6
4	5	B	INT	Informatica	6
4	5	A	INT	TPS	4
4	5	B	INT	TPS	4

Verificare che grazie al modello relazionale è possibile, ad esempio, rispondere alle seguenti domande:

Quante ore di storia insegna la prof.ssa Pierone in 3A INT?

Quante ore di italiano insegna la prof.ssa Pierone in 3A INT?

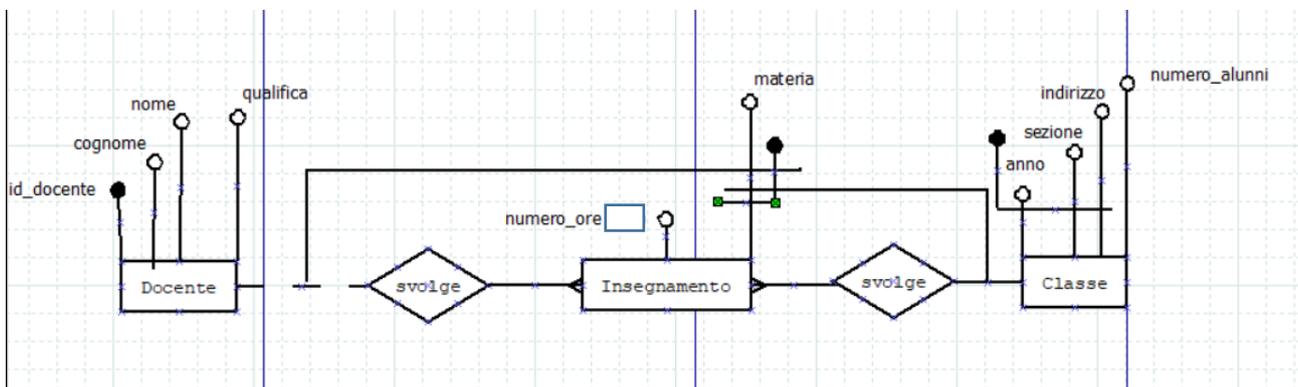
Quali sono i docenti che insegnano più di una materia nella stessa classe?

Quali docenti non insegnano in alcuna classe?

Ecc....

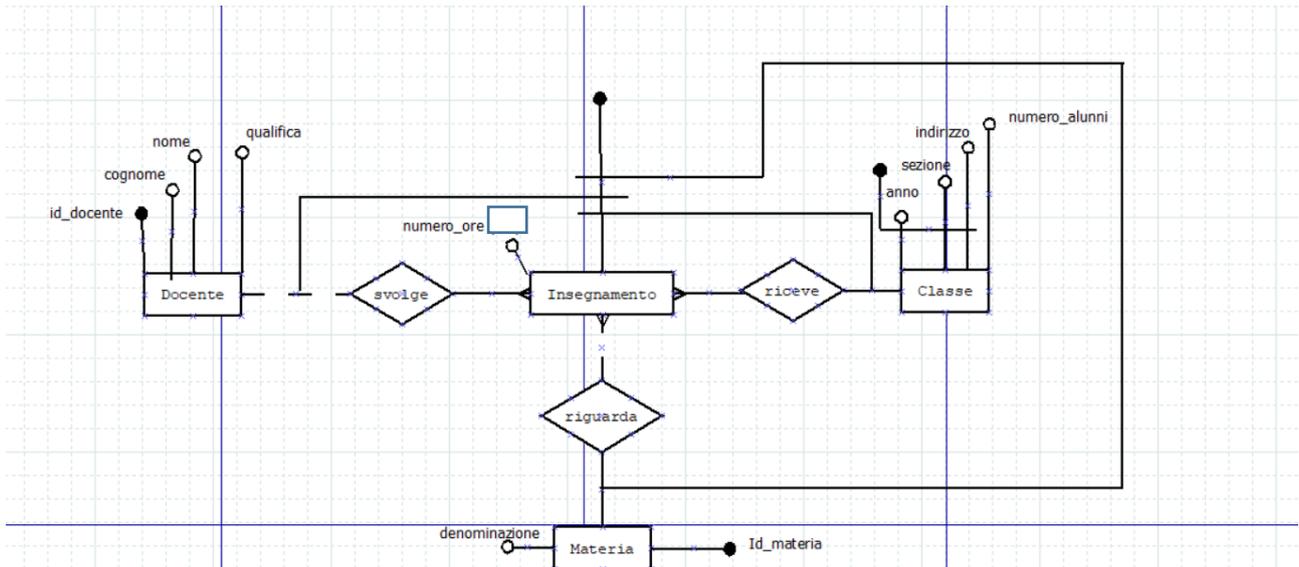
OSSERVAZIONE 1:

Si può verificare che, trasformando una l'associazione N-N insegnamento in due associazioni 1-N, applicando la regola di derivazione si sarebbe ottenuto lo stesso risultato.



OSSERVAZIONE 2:

Si noti che l'attributo "materia" assume un numero limitato di valori (italiano, storia ecc..) che si ripetono numerose volte. Questo può rendere utile introdurre una entità materia (*id_materia*, denominazione). In tal caso si otterrebbe la seguente soluzione:



DB SCHEMA

Docenti (id_docente, cognome, nome, qualifica)

Classi (anno, sezione, indirizzo, numero_alunni)

Materie (id_materia, denominazione)

Insegnamenti (docente, anno, sezione, indirizzo, materia, numero_ore)

TABELLE

Classi e Docenti rimangono uguali.

Materie

Tabella che rappresenta gli insegnamenti dei docenti nelle classi.

Nome	Chiave	Tipo	Dimensione	Not null
id_materia	PK	numerico		X
denominazione		carattere	50	X

Chiave primaria: id_materia

Insegnamenti

Tabella che rappresenta gli insegnamenti dei docenti nelle classi.

Nome	Chiave	Tipo	Dimensione	Not null
docente	PK/FK	numerico		X
anno	PK/FK	numerico		X

sezione	PK/FK	carattere	3	X
indirizzo	PK/FK	carattere	3	X
materia	PK/FK	numerico		X
numero_ore		numerico		X

Occupi meno byte rispetto alla denominazione della materia

Chiave primaria: docente, anno, sezione, indirizzo, materia

Chiavi esterne: docente → Docenti.id_docente,

classe → Classi.anno, Classi.sezione, Classi.indirizzo

materia → Materie.id_materia

Esempio estensionale

Materie (esempio estensionale)

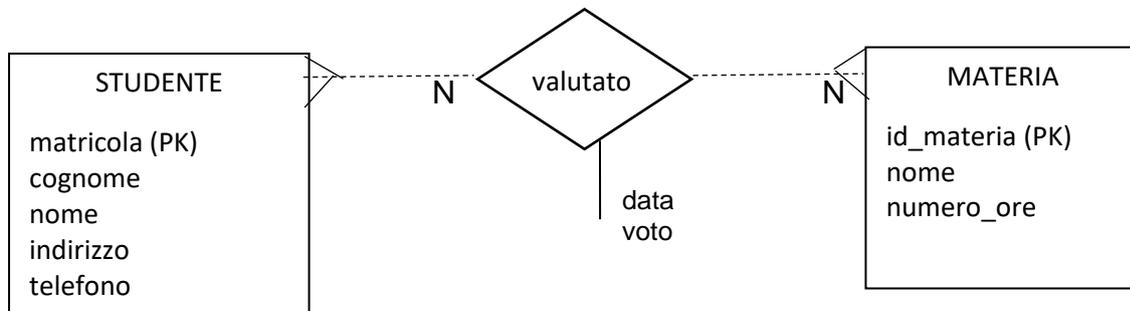
id_materia	denominazione
1	Italiano
2	Storia
3	Informatica
4	TPS
5	Sistemi e reti
6	inglese
7	religione

Occupi meno byte rispetto alla denominazione della materia

Insegnamenti (esempio estensionale)

docente	classe	sezione	indirizzo	materia	numero_ore
2	1	A	INT	6	3
2	2	A	INT	6	3
2	4	A	INT	6	3
1	1	A	INT	7	1
1	2	A	INT	7	1
1	3	A	INT	7	1
1	4	A	INT	7	1
1	5	A	INT	7	1
6	4	A	INT	1	7
6	3	A	INT	1	4
6	3	A	INT	2	3
8	1	A	INT	6	4
8	2	A	INT	6	4
8	3	A	INT	6	4
8	4	A	INT	6	4
7	1	A	INT	2	3
7	2	A	INT	2	3
3	3	A	INT	3	6
3	4	A	INT	3	6
4	5	A	INT	3	6
4	5	B	INT	3	6
4	5	A	INT	4	4
4	5	B	INT	4	4

5. Studenti / Materia (associazione con attributi N a N che diventano tre relazioni). Nella relazione Valutazione, per formare la chiave primaria è stato aggiunto l'attributo Data dell'associazione "valutare" alle due chiavi esterne *Matricola* e *CodiceMateria*)



Db Schema:

Studenti (matricola, cognome, nome, indirizzo, telefono)

Materie (id_materia, nome, numero_ore)

Valutazioni (matricola, id_materia, data, voto) (data è necessario per formare PK)

Tabelle:

Studenti

Tabella che contiene i dati anagrafici degli studenti

Nome	Chiave	Tipo	Dimensione	Not null
matricola	PK	numerico		X
cognome		carattere	30	X
nome		carattere	30	X
indirizzo		carattere	30	X
telefono		carattere	30	X

Chiave primaria: matricola

Materie

Tabella che contiene informazioni sulle materie

Nome	Chiave	Tipo	Dimensione	Not null
id_materia	PK	numerico		X
nome		carattere	30	X
numero_ore		carattere	30	X

Chiave primaria: id_materia

Valutazioni

Tabella che contiene data e voto delle valutazioni di ogni studente

Nome	Chiave	Tipo	Dimensione	Not null
matricola	PK/FK	numerico		X
id_materia	PK/FK	carattere	30	X
data	PK	data/ora		X
voto		numerico		X

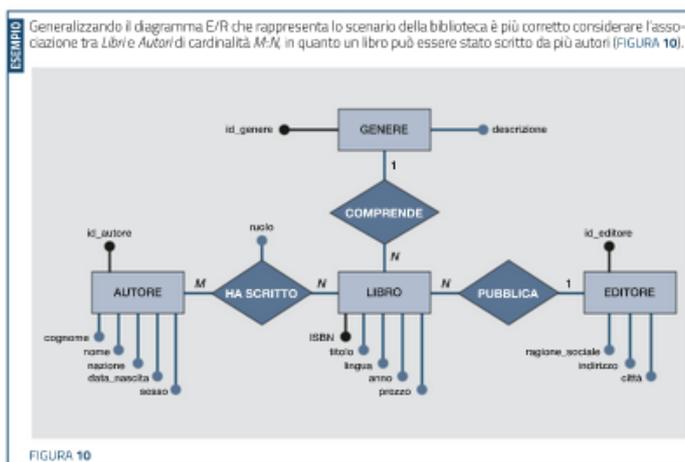
Chiave primaria: matricola, id_materia, data

Chiavi esterne: matricola → Studenti.matricola
id_materia → Materie.id_materia

ESEMPIO.

Applicando le regole di derivazione, generiamo le relazioni dal diagramma E/R dell'esempio "biblioteca".

Partiamo da diagramma E/R:



- Partiamo dall'entità Autore derivando con la regola 1 la relazione Autori (plurale). Lo schema risultante è il seguente:

Autori (**id autore**, cognome, nome, nazione, data_nascita, sesso)

- Dall'entità Libro, sempre con la regola 1, deriviamo la relazione Libri

Libri (**ISBN**, titolo, lingua, anno, prezzo)

poiché l'entità è in associazione 1 a N con l'entità Genere, e con l'entità Editore aggiungiamo come chiavi esterne le chiavi primarie di tali entità

Libri (**ISBN**, titolo, lingua, anno, prezzo, *id_genere*, *id_editore*)

- Dall'associazione N a M fra Autore e Libro, con la regola 7 deriviamo la relazione Autori-Libri

Autori_Libri (**id autore**, **ISBN**, ruolo)

- Sempre con la regola 1 deriviamo le relazioni Generi ed Editori

Generi (**id genere**, descrizione)

Editori (**id editore**, ragione_sociale, indirizzo, città)

Lo **schema logico completo della base di dati** è dato quindi dai seguenti **schemi**:

Libri (**ISBN**, titolo, lingua, anno, prezzo, *id_genere*, *id_editore*)

Autori (**id autore**, cognome, nome, nazione, data_nascita, sesso)

Editori (**id editore**, ragione_sociale, indirizzo, città)

Generi (**id genere**, descrizione)

Autori_Libri (**id autore**, **ISBN**, ruolo)

Possiamo mostrare con le tabelle un possibile stato in un determinato istante della base di dati (p. A45 del libro)

Autori

id_autore (PK)	cognome	nome	nazione	data_nascita	sesto
A001	Tolkien	John	Inghilterra	03/01/1892	M
A002	Collodi	Carlo	Italia	24/11/1826	M
A003	Meini	Giorgio	Italia	23/03/1965	M
A004	Formichi	Fiorenzo	Italia	06/10/1954	M

Editori

id_editore (PK)	ragione_sociale	indirizzo	citta
E099	Zanichelli	Via irnerio, 34	Bologna
E100	Mondadori	Via Bianca di Savoia, 12	Milano
E101	RCS	Via Rizzoli, 8	Milano

Generi

id_genere (PK)	descrizione
G01	Fantasy
G99	Tecnologia
G90	Scienze

Libri

ISBN (PK)	titolo	lingua	anno	prezzo	id_genere (FK)	id_editore (FK)
123-456-789	Pinocchio	Italiano	1990	10,00	G01	E100
987-654-321	The Hobbit	Inglese	1999	10,00	G01	E101
012-345-678	The Lord of the Rings	Inglese	2001	25,00	G01	E101
876-543-210	Informatica	Italiano	2012	20,00	G99	E099

Autori_Libri

id_autore (PK) (FK)	ISBN (PK) (FK)	ruolo
A002	123-456-789	autore
A001	987-654-321	autore
A001	012-345-678	autore
A004	876-543-210	autore
A003	876-543-210	coautore

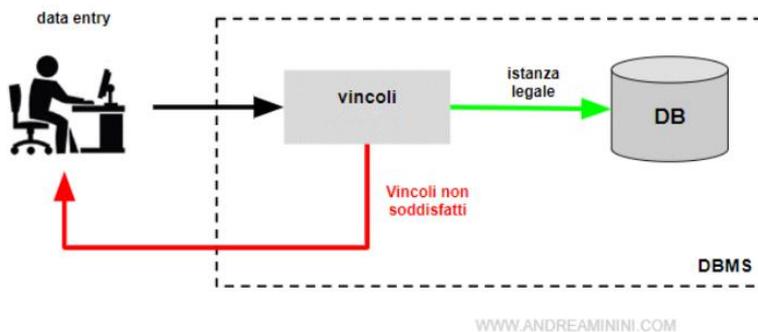
OSSERVAZIONE IMPORTANTE:

La chiave esterna (FK foreign key) serve per “collegare” le tuple di una tabella con le tuple della tabella associata.

Ad esempio nella tabella Libri per ottenere i dati dell’editore del libro Pinocchio basta “seguire” la chiave esterna id_editore.

VINCOLI DI INTEGRITA'

I vincoli di integrità in un database sono regole che assicurano la validità e la coerenza dei dati. Questi vincoli possono essere applicati a livello di schema e sono fondamentali per mantenere **la qualità dei dati** memorizzati. Quando, attraverso una qualsiasi applicazione si cercherà di aggiungere dei dati, soltanto le istanze che rispettano i vincoli di integrità, potranno essere aggiunte al database.



Definizione

1. Come già detto i **vincoli di integrità** sono le **condizioni** che devono essere rispettate dalle istanze del DB affinché i dati siano **significativi e validi**.
2. Ogni vincolo può essere espresso come una funzione booleana che, relativamente ad un'istanza del DB, può assumere valore vero o falso. L'istanza viene inserita nel DB solo se l'espressione è vera, quindi se il vincolo viene rispettato. I DBMS offrono degli strumenti per facilitare l'inserimento dei vincoli.

Prima di elencare tutti i possibili vincoli definiamo un vincolo particolarmente importante. Il "vincolo di integrità referenziale". Questo vincolo è espresso dalla chiave esterna di un data base relazionale ed è definito nel seguente modo.

Vincolo di integrità referenziale: garantisce che, date due tabelle T1 e T2, dove l'attributo K è chiave primaria di T1 e chiave esterna di T2, non sia possibile aggiungere un riga alla tabella T2 se il valore assunto da K in questa riga non è già presente in T1.

Ad esempio non è possibile aggiungere una riga alla tabella Libri se il valore di id_editore della nuova riga non è già presente nella tabella Editori.

(in altre parole: è necessario che ogni nuova riga nella tabella con chiave esterna, abbia un valore della chiave esterna che è **già presente** come valore della chiave primaria nella tabella collegata)

Vediamo ora tutti i vincoli di integrità e classifichiamoli.

I vincoli si suddividono in:

- vincoli **interrelazionali** (all'interno di una sola relazione)
- vincoli **intrarelazionali** (fra elementi diverse relazioni)

INTRA-RELAZIONALI	INTER-RELAZIONALI
<p>Vincoli di chiave</p> <ol style="list-style-type: none"> 1. il valore della chiave primaria non deve essere null 2. il valore della chiave primaria non deve ripetersi nelle diverse tuple 	<p>Vincolo di integrità referenziale</p>
<p>Vincoli di tupla Interessano una sola tupla indipendentemente dai valori assunti dalle altre tuple.</p> <p>Se interessa un solo attributo è detto anche vincolo di attributo o vincolo di dominio.</p> <ol style="list-style-type: none"> 1. L'attributo non può assumere valore null. esempio: <code>data_nascita!=null</code> (solo se questa espressione è vera il valore <code>data_nascita</code> verrà aggiunto al database) 2. L'attributo può assumere solo determinati valori esempio: <code>voto>=18 and voto<=30</code> 3. Vincolo di tupla su più attributi, si esprime con un predicato dell'algebra di boole che coinvolge più attributi. Esempio: l'attributo booleano "lode" può avere valore "true" solo se l'attributo "voto" ha valore 30). $!(lode=true \text{ and } voto!=30)$ Si può esprimere Anche come: $!(lode=true) \text{ or } voto=30$ 	

Già espresso anche nel diagramma e/r con la cardinalità minima=1 dell'attributo. L'obbligatorietà dell'associazione (1) indica che una fk non può assumere valore null



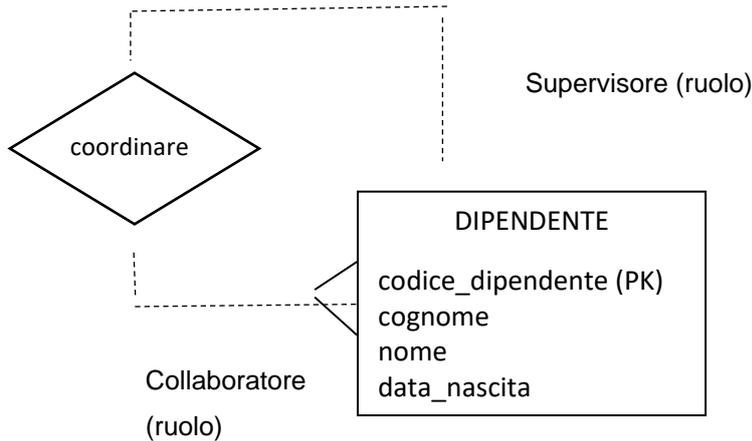
4. Vincolo di unicità: l'attributo (o insieme di attributi) non può ripetersi nelle tuple

Esempio: in una tabella utenti ci sono sia `id_utente` che `codice_fiscale`, se all'`id_utente` è assegnato il vincolo di chiave, si può aggiungere un ulteriore vincolo per indicare che il codice fiscale non possa ripetersi, questo è un vincolo di unicità

UNICO (`codice_fiscale`)

Caso particolare di progettazione logica, associazione ricorsiva:

Rappresentiamo il modello E/R dei dipendenti di un'azienda dove alcuni dipendenti sono anche supervisori e coordinano un certo numero di persone (anche essi dipendenti). I dipendenti coordinati (hanno il ruolo di collaboratori) dipendono da un solo supervisore e ci sono alcuni dipendenti che non dipendono da alcun supervisore.



Ci si comporta come in una normale associazione 1 a N ossia si dovrebbero creare le tabelle delle due entità (ma in questo caso la tabella è solo una) e si aggiunge la chiave esterna alla relazione dell'entità che svolge il ruolo **a molti**. In questo caso dunque, poiché le due entità sono la stessa entità, la tabella è una sola

Dipendenti: (codice dipendente, cognome, nome, data_nascita, *codice_supervisore*)

FK: codice_supervisore → Dipendenti.codice_dipendente

Il CodiceSupervisore sarà il CodiceDipendente del dipendente che svolge il ruolo di supervisore. Si osservi che, poiché ci sono dipendenti che non hanno supervisore, la chiave esterna deve poter assumere valore null.

Dipendenti

codice_dipendente	cognome	nome	data_nascita	codice_supervisore
d01	Rossi	Piero	1/1/2000	null
d02	Bianchi	Pino	2/2/200	d01
d03	Verdi	Lino	1/3/2000	d01
d04	Neri	Lina	1/3/2001	d01
d05	Silli	Mina	2/2/2000	null
d06	Minas	Claudio	1/1/1987	d05

Si noti che, grazie al modello relazionale, è possibile rispondere alle seguenti interrogazioni sulla base di dati:

Chi è il supervisore di Rossi Piero?

Chi è il coordinatore di Neri Lina?

Chi sono i supervisori?

Quanti collaboratori ha Rossi Piero?

Chi sono i dipendenti che non sono supervisori e che non hanno supervisore?

Soluzione alternativa (comunque valida)

Nel caso si volesse evitare di avere molte tuple con valore NULL nel campo codice_supervisore per i dipendenti che non hanno supervisore, un'altra soluzione è quella di derivare una tabella dall'associazione Coordinare in cui sono presenti solamente le tuple dei dipendenti che hanno il supervisore:

Dipendenti: (codice_dipendente, cognome, nome, nata_nascita)

Coordinamenti: (codice_dipendente, *codice_supervisore*)

FK: codice_supervisore: Dipendente.codice_dipendente

Dipendenti

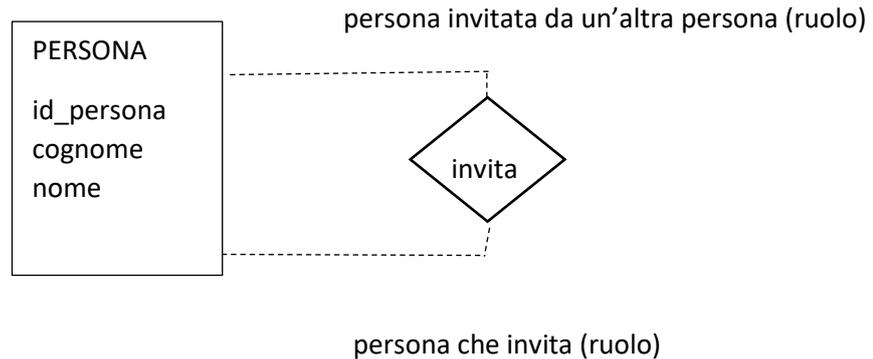
codice_dipendente	cognome	nome	data_nascita
d01	Rossi	Piero	1/1/2000
d02	Bianchi	Pino	2/2/200
d03	Verdi	Lino	1/3/2000
d04	Neri	Lina	1/3/2001
d05	Silli	Mina	2/2/2000
d06	Mins	Claudio	1/1/1987

Coordinamenti

codice_dipendente	codice_supervisore
d02	d01
d03	d01
d04	d01
d06	d05

Analogamente si trattano le associazioni ricorsive 1 a 1 a N a N

Esempio 1 a 1: alla cerimonia di matrimonio ogni persona può invitare al più un'altra persona



Regole di lettura:

Ogni persona può essere invitata da un'altra persona (lo è quando non è invitata direttamente dagli sposi)

Ogni persona può invitare un'altra persona (oppure no)

Come si determinano le relazioni?

Si crea la tabella dell'entità 1, la tabella dell'entità 2 non c'è perché l'entità 2 è sempre l'entità 1, si aggiungono quindi la chiave esterna e l'eventuale attributo dell'associazione (che in questo caso non c'è) sempre alla tabella 1.

Persona: (id_persona, cognome, nome, *persona_invitata*)

(oppure si potrebbe aggiungere la persona invitante, a scelta)

FK: *persona_invitata* → Persona.id_persona

id_persona	cognome	nome	persona_invitata
s1	Cani	Piero	s4
s2	Gatti	Pino	s6
s3	Lupi	Lino	null
s4	Topi	Lina	null
s5	Sillia	Mina	null
s6	Fintus	Claudio	null

Soluzione alternativa.

Persone: (id_persona, cognome, nome)

Inviti: (id_invitante, id_invitato)

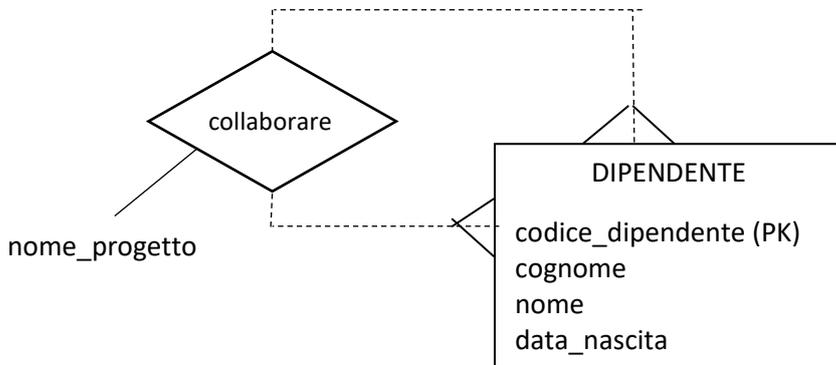
Persone

id_persona	cognome	nome
s1	Cani	Piero
s2	Gatti	Pino
s3	Lupi	Lino
s4	Topi	Lina
s5	Sillia	Mina
s6	Fintus	Claudio

Inviti

id_invitante	id_invitato
s1	s4
s2	s6

Esempio N a N: ad esempio i dipendenti collaborano fra loro per dei progetti. Ogni dipendente può collaborare con altri dipendenti. Alcuni dipendenti possono non collaborare con alcun dipendente.



Si crea la tabella dell'entità 1, la tabella dell'entità 2 non c'è perché l'entità 2 è sempre l'entità 1, si crea la tabella della associazione con chiavi esterne e gli attributi dell'associazione (nome_progetto)

Dipendenti: (codice_dipendente, cognome, nome, data_nascita)
 PK: codice_dipendente

Collaborazione: (nome_progetto, codice_collaboratore1, codice_collaboratore2)

PK: codice_collaboratore1, codice_collaboratore1, nome_progetto ←

FK: codice_collaboratore1: → Dipendenti.codice_dipendente
 codice_collaboratore2: → Dipendenti.codice_dipendente

nome_progetto è necessario nella PK per distinguere il caso in cui due dipendenti collaborano in più progetti

Dipendenti

codice_dipendente	cognome	nome	data_nascita
d01	Rossi	Piero	1/1/2000
d02	Bianchi	Pino	2/2/200
d03	Verdi	Lino	1/3/2000
d04	Neri	Lina	1/3/2001
d05	Silli	Mina	2/2/2000
d06	Mins	Claudio	1/1/1987

Collaborazioni

codice_colaboratore1	codice_colaboratore2	nome_progetto
d01	d02	alfa
d01	d03	alfa
d02	d03	alfa
d06	d05	beta

d06	d01	beta
d01	d05	beta

Se d01 collabora con d02 la collaborazione inversa è ovvia, non va indicata.

Se d01 collabora con d02 e con d03 va indicato, se è vero, che d02 collabora con d03, perché non è detto che sia vero.