

BASI DI DATI

By Edoardo

BASI DI DATI

Collezione di dati gestiti da un software chiamato DBMS

I dati si trovano nel livello dati, il terzo livello del software a 3 livelli.

DBMS gestisce ogni genere di dati (grandi, persistenti e condivisi) garantendone sempre la disponibilità, privatenza e affidabilità.

MODELLO RELAZIONALE

I vari domini sono chiamati ATTRIBUTI, associati ciascuno a un insieme di valori.
Le righe sono le n-uple.

Si può accedere ad un valore di un attributo di una specifica n-upla in due modi:

$$t[\text{attributo}] \quad o \quad t.\text{attributo} \quad (t = n\text{-upla})$$
$$\downarrow$$
$$t[\text{at}_1, \text{at}_2]$$

Schema Relazione = elenco attributi di una relazione: nome(at₁: string, at₂: int ...)

N.B. il tipo degli attributi vengono spesso omessi.

Può capitare che le informazioni siano incomplete quindi non è consigliabile usare attributi troppo particolari.

Informazioni incomplete "non sono accessibili a tutti" ovvero non tutti possono interpretare la struttura dati.

La soluzione è l'utilizzo del valore NULL che non si confonde con gli attributi mancanti.

La sua interpretazione dipende dalla base di dati.

• Vincoli di integrità

Regole che la base di dati deve rispettare affinché sia ragionevole.

Ese. TABELLA STUDENTE: non possono esserci due studenti con lo stesso matricola.

Può essere visto come una funzione booleana che associa ad ogni istanza vero o falso

+ Intrarelazionali

+ Interrelazionali (tra dati di tabella diverse)

Vincoli Intrarelazionali DI TUPLA

Esprimono condizioni sui valori di ciascuna tupla di una relazione indipendentemente dalle altre tuple.

Si esprime come un'espressione booleana (con AND, OR, NOT) tra attributi e valori. Si dice DI DOMINIO se coinvolge un solo attributo.

Ese.

LORDO	NETTO	RITEN.
-------	-------	--------

$$\rightarrow \text{LORDO} = \text{RITENUTE} + \text{NETTO} \quad (\text{DI TUPLA})$$

VOTO >= 18 AND VOTO <= 30 = VOTO Dominio

$$\alpha \Rightarrow \beta = \text{NOT } \alpha \text{ OR } \beta$$

implica

$$\text{NOT } (\text{VOTO} = \text{"e bade"}) \text{ OR } \text{VOTO} = 30$$

Vincolo Interoazionale DI CHIAVE

SUPERCHIAVE: è un insieme di attributi per i quali non esistono due tuple con gli stessi valori su questi attributi

CHIAVE: se K è superchiave e non contiene altre superchiavi è una superchiave minima e quindi detta chiave

Se non può avere valori nulli si dice chiave **PRIMARIA**

Un vincolo di chiave specifica che una chiave deve avere valori univoci e una chiave primaria non deve avere valori nulli.

Vincolo Interoazionale DI INTEGRITÀ REFERENZIALE (Foreign Key)

Dati uno o più attributi essi devono essere presenti su un'altra tabella come chiave

ES. TABELLA MULTE: lo targa nella tabella deve comporre obbligatoriamente nella tabella AUTO come chiave

Vincolo Interoazionale DI INCLUSIONE

Come sopra ma l'attributo della tabella di arrivio non è una chiave

ALGEBRA RELAZIONALE

O operazione sullo schema (DDL) + operazioni su dati (DML)

Operatori

- Unione, intersezione, differenza
- Ridenominazione
- Selezione
- Proiezione
- Join (naturale, prodotto cartesiano, theta-Join)

Riduiscono una relazione, definita su relazioni e possono essere composti.

Gli operatori insiemistici possono essere usati solo tra relazioni con gli stessi attributi.

N.B. Se in questi abbiamo delle chiavi negli operandi, nel risultato non le avrà più.

Ridenominazione: $RGN_{new_name} \leftarrow old_name (name_tabel)$

Selezione: restituisce le tuple che soddisfano una data caratteristica

$SEL_{condizione} (Tabella)$ la condizione è un'espressione booleana come nei vincoli di tupla

Proiezione: invece di restituire le righe restituisce alcune specifiche colonne

$PROJ_{listattributi} (TABELLA)$

JOIN

A differenza degli altri, crea tuple nuove.

- Naturale

Lavora su due relazioni. Il risultato ha come attributi l'unione degli attributi dei due operandi.
($X_1 X_2 = \text{unione insieme di attributi}$)

Dati due schemi di relazioni $R_1(x_i)$ e $R_2(x_j)$, $R_1 \text{ JOIN } R_2$ è una relazione sui $x_i x_j$ le cui emmple sono

$$\{t \text{ su } x_i x_j \mid \exists t_1 \in R_1 \text{ e } t_2 \in R_2 \mid t[x_i] = t_1 \text{ e } t[x_j] = t_2\}$$

le combinazioni delle emmple con lo stesso valore nell'attributo in comune

Se $X_1 X_2$ non hanno attributi in comune, JOIN naturale calcola il prodotto cartesiano

- Completo

È un JOIN naturale in cui tutte le emmple contribuiscono al risultato. Nel caso contrario è Non completo o addirittura vuoto se nessuna contribuisce

- Theta - Join

Combinazione di un JOIN tra relazioni senza attributi in comune e una selezione: $R_1 \text{ JOIN}_{\text{condizione}} R_2$

Se la condizione (Θ) è composta solo da "=" si dice Equi-Join

Esercizio 4

1) $\text{SEL}_{\text{STIPENDIO} > 40} (\text{IMPIEGATO})$

2) $\text{PROJ}_{\text{MATRICOLA}, \text{NORTE}, \text{ETA}} (\text{SEL}_{\text{STIPENDIO} > 40} (\text{IMPIEGATO}))$

3) $\text{PROJ}_{\text{CAPO}} (\text{SEL}_{\text{STIPENDIO} > 40} (\text{IMPIEGATO}) \text{ JOIN } \text{REN}_{\text{MATRICOLA} \leftarrow \text{IMPIEGATO}} (\text{SUPERVISIONE}))$

4) $\text{PROJ}_{\text{NORTE}, \text{STIPENDIO}} (\text{PROJ}_{\text{CAPO}} (\text{SEL}_{\text{STIPENDIO} > 40} (\text{IMPIEGATO}) \text{ JOIN } \text{REN}_{\text{matricola} = \text{impianto}} (\text{SUPERVISIONE}))$

$\text{JOIN}_{\text{CAPO} = \text{MATRICOLA}} \text{ IMPIEGATO})$

5) $\text{IMPIEGATO} \text{ JOIN}_{\text{mat} = \text{imp}} \text{ SUPERVISIONE} \quad \wedge \quad \text{PROJ}_{\text{CAPO}} (\text{SUPERVISIONE}) \text{ JOIN}_{\text{cap} = \text{mat}} \text{ IMPIEGATO}$

CAPOLITAT | NORTE | ETA | STIPENDIO

$\text{JOIN}_{\text{MATCPO} = \text{CAP}}$

$\text{Sel}_{\text{STIP1} > \text{STIP2}}$

NAT CAP | NORTE | ETA | STIPENDIO

6)
 $\text{PROJ}_{\text{IMPIEGATO}} (\text{REN}_{\text{MATRICOLA} \leftarrow \text{IMPIEGATO}} \text{ SUPER.})$



$- \text{ PROJ}_{\text{CAPO}, \text{MATRICOLA}} (\text{IMPIEGATO} \text{ JOIN}_{\text{MATCPO} = \text{IMPIEGATO}} \text{ SUPERVISIONE})$

$\text{PROJ}_{\text{matricola}} (\text{IMPIEGATO}) - \text{REN}_{\text{matricola} \leftarrow \text{impianto}} (\text{PROJ}_{\text{IMPIEGATO}} (\text{SUPERVISIONE}))$

7) PROJ_{CAPD} SUPERVISIONS - PROJ_{CAPD} (Sel _{$SUPC < 40$} (IMPIEGATO JOIN_{matricola = impiegato} SUPERVISIONE))

NULL non soddisfa nessuna condizione!

ID	ETA'
1	NULL
2	35

→ Sel _{$ETA' > 30$}

ID	ETA'
2	35

Null soddisfa solo IS NULL

- JOIN E STERNO

Prende le tuple che non si combinano e aggiunge NULL ai valori mancanti. Le tuple che verrebbero escluse da un JOIN sono estese con dei NULL

- JOIN SINISTRO (JOIN_{LEFT})

Mantiene tutte le tuple del primo operando, estendendole con NULL se necessario

- JOIN DESTRO (JOIN_{RIGHT})

Come sopra, ma del secondo operando

- JOIN COMPLETO (JOIN_{FULL})

SQL

- CREATE TABLE: crea una tabella vuota specificando attributi e vincoli
es. CREATE TABLE nome_tabella (
 nome_att.1 Dominio [Vincoli]
 ...
 nome_att.n Dominio [Vincoli]
)

N.B. I vincoli sono opzionali

In SQL una tabella è definita come multi-insieme e quindi può contenere duplicati.

- CREATE DOMAIN: crea un nuovo dominio con vincoli e valori di default
es. CREATE DOMAIN nome_dominio
 AS Dominio_Presistente [Default] [Vincoli]

es2 CREATE DOMAIN Voto
 AS Int default null
 check (Value >= 18 AND Value <= 30)

Vincoli Interoazionali

- Singoli attributi *references*
- Più attributi *foreign key*

Sintassi:

- 1) *references* dopo la specifica del dominio
- 2) *foreign key* (att₁, att₂...) *references*

- ALTER TABLE: permette di modificare una tabella. Questo comando mi permette di aggiungere in un secondo momento vincoli con ADD, necessario poiché se dichiara vincoli referenziali prima di aver istanziato la seconda tabella, il sistema mi dà errore.

- DROP TABLE: elimina una tabella
es. DROP TABLE nome_tabella restrict | cascade

AUTOVACUTAZIONE

1) SEL

$\text{ANNO} > 2000 \text{ AND } (\text{CITTÀ} = \text{FIRENZE} \text{ OR } \text{CITTÀ} = \text{ROMA})$

VIAGGIO

2) PROJ (SEL (CITTÀ) JOIN SEL (VIAGGIO))
 Nome NUMABITANTI > 1KK Nome = CITTÀV ANNOV = 2001

3) PROJ ((SEL (PERSONA) JOIN VIAGGIO)
 PERS, REGIONE CITTÀN = CAGLIARI CF = PERS
 JOIN CITTÀ)
 Nome CITTÀV = Nome

4) PROJ (SEL (PERSONA JOIN VIAGGIO)
 Nome, REGIONE ANNOV - ANNON ≤ 18 CF = PERS
 JOIN CITTÀ))
 Nome CITTÀV = Nome

5) PROJ (

Nome, REGIONE
 (PERSONA JOIN CITTÀ) JOIN (REN
 CITTÀN = Nome CF = PERS AND REGIONE = REGIONE1
 VIAGGIO JOIN CITTÀ)))
 Nome CITTÀV = Nome

6) PROJ (

CF, CF2

PROJ (PERSONA JOIN VIAGGIO) JOIN
 CF, CITTÀV, ANNOV CF = PERS CF2 != CF AND CITTÀN = CV2
 AND AN2 = ANNON

REN
 CF2, CV2, AN2 ← CF, CITTÀN, ANNOV
 CF, CITTÀV, ANNOV CF = PERS

(PROJ (PERSONA JOIN VIAGGIO)))

PROJ (REN
 PERS, P2 P2, ANNO2, CITTA2 ← PERS, ANNO1, CITTA1
 (VIAGGIO)

JOIN
 PERS != P2 AND CITTA1 = CITTA2 AND ANNO1 = ANNO2
 VIAGGIO)

7) PROJ (PERSONA) - PROJ (PERSONA JOIN
 CF CF
 CITTA1 = CITTA1
 AND
 CF = PERS.

8) PROJ (PERSONA) -
 CF, CITTA1

PROJ (PERSONA JOIN PROJ
 CF, CITTA1 CF = PERS PERS.
 SEL (VIAGGIO JOIN
 REGIONE != TOSCANA CITTA')))
 CITTA1 = NOME

10) (PROJ VIAGGIO JOIN REN (PROJ VIAGGIO)) -
 PERS P2 ← PERS PERS

PROJ (REN
 PERS, P2 P2, ANNO2, CITTA2 ← PERS, ANNO1, CITTA1
 (VIAGGIO)

JOIN
 PERS = P2 OR CITTA1 != CITTA2 OR ANNO1 != ANNO2
 VIAGGIO)

8) PROJ (VIAGGIO) -
 PERS

PROJ (PERS

PROJ (VIAGGIO JOIN
 PERS, CITTA1, REGIONE CITTA')
 CITTA1 = NOME

JOIN
 P2 = PERS AND CITTA1 ≠ CITTA2 AND REGIONE = R2

REN (PROJ (VIAGGIO JOIN
 P2, C2, R2 ← PERS, CITTA1, REGIONE CITTA'))
 CITTA1 = NOME

II) PROJ

CF, CITTÀ N

(PERSONA JOIN

CF = PERS

PROJ

PERS

VIAGGIO

JOIN

PROJ

NOTE

SEL

(CITTÀ)

-

PROJ

PERS, NOTE

(VIAGGIO JOIN

NOTE = CITTÀ N

SEL

REGIONE = LIGURIA

(CITTÀ))

MANIPOLAZIONE DATI SQL

- **INSERT**: aggiunge delle n-uple alla tabella

SINTASSI: `INSERT INTO nome_tabella [(ATTRIBUTI)]
values (Valori)`

es. `INSERT INTO persona (name, età, reddito)
values ("Pino", 25, 52)`

- **DELETE**: rimuove le n-uple che soddisfano una data condizione

SINTASSI: `DELETE FROM nome_tabella [WHERE condizione]`

NB. Se non specifico la condizione il where sarà true di default e cancellerà tutta la tabella.

- **UPDATE**: modifica le n-uple che soddisfano una data condizione

SINTASSI: `UPDATE nome_tabella
SET Attributo = <Espressione | Select | Null | ...> [WHERE condizione]`

es. `UPDATE persona SET reddito = 45 WHERE name = "Pino"`

SCHEMA

`CREATE SCHEMA <Nome>`

crea una nuova base di dati

`USE <Nome>`



Una volta chiamata lo "use", posso iniziare a lavorare con la mia nuova base di dati.

INTERROGAZIONI IN SQL

- **SELECT**

SINTASSI: `SELECT Attributo... Attributo FROM Tabella ... tabella [WHERE condizione]`

NB. In SQL gli attributi si denotano con "nome_schema.nome_tabella.nome_attributo"

Se la FROM è solo da una tabella, il risultato sarà una proiezione sugli attributi specificati.

Il risultato di una **query**, SQL lo inserisce in una nuova tabella.

SELECT * seleziona tutti gli attributi

- **ALIAS (AS)**: servir per rinominare gli attributi o le tabelle

es. `Select nome AS N, età AS E FROM persona AS p`

NB. In SQL posso avere n-uple uguali. Se non voglio avere i doppioni utilizzo **SELECT DISTINCT**

Precedenza Operatori Logici: 1. NOT 2. AND 3. OR

Condizione Like: confronta un attributo di tipo stringa con un'espressione regolare
es. WHERE nome LIKE 'A_d%'

nomi da iniziali per A seguita da un corrotore qualunque, inci d' e poi una sequenza qualsiasi

"—" = un corollario "..." = una sequenza di corollari

In SQL, JOIN si applica mettendo due tabelle dopo FROM specificando dopo WHERE i campi in comune (se ci sono)

CS. SELECT R1.A1, R2.A4 FROM R1, R2 WHERE R1.A2 = R2.A3
R1(A1,A2) & R2(A2,A4)

Se utilizzo due o più volte lo stesso elemento nella form

Ex. From R₁, R₂, R₁ → From R₁ As X, R₂, R₁ As Y

È necessario ridenominarli

ESEDAO:

PROSPADRE (PATERNITÀ) JOIN FIGLIO = NOME SEL REDATO > 20 (PERSONE)

```
SELECT DISTINCT PATERNITÀ.PADRE  
FROM PATERNITÀ, PERSONE  
WHERE PATERNITÀ.NOME = PERSONE.NOME AND PERSONE.PREDDITO > 20
```

PROJ_HADRE PADRE (PATERNITÀ) NON PATERNITÀ

```
SELECT MADRE , PADRE  
FROM MATERNITÀ , PATERNITÀ  
WHERE MATERNITÀ . FIGLIO = PATERNITÀ . FIGLIO
```

SEL (
 $R_2 > R$)

Peso (PATERNITÀ JOIN PERSONE) JOIN PROJ (REN_{R2 ← REDDITO}(PERSONE))

```
SELECT f.NOME, f.REDDITO, p.REDDITO  
FROM PERSONE p, PATERNITA' t, PERSONE f  
WHERE p.NOME = t.PADRE AND t.FIGLIO = f.NOME AND f.REDDITO > p.REDDITO
```

JOIN esplicito

...
From t_1 JOIN t_2 ON condizione (è possibile continuare con JOIN ... on)

NATURAL JOIN

From t_1 natural join t_2

NB. poco usato

JOIN ESTERNO

- left join
- right join
- full join

— ORDER BY

Dopo lo where

SINTASSI: ORDER BY Attributo (desc nel caso discendente)

— LIMIT

Dopo lo where orderby e indica il numero di righe massime

Operatori Aggregati

- Count: conta il numero di n-uple $\rightarrow \text{count}(\ast)$

conta il numero elementi $\neq \text{NULL}$ compresi le copie $\rightarrow \text{count(Att.)}$

conta il numero di elementi distinti $\rightarrow \text{count(distinct Att.)}$

- SUM , AVG (media) , min , max

Lavorano su argomenti o espressione (non \ast)

Sum e Avg argomenti numerici e tempo

Min e Max su argomenti ordinabili

E.S. SELECT AVG ("Attributo")

FROM TABELLAVOTI

Restituisce una tabella con solo attributo = AVG (Attributo) e la media di tutti i voti della tabella TABELLAVOTI

- GROUP BY

group by Att. per ogni valore di Att. crea dei gruppi formati da valori uguali

es.

matricola | voto | corso

140	30	BD
33	30	AM
74	30	SDC

group by
voto

Select Voto, count(*) as NumES → Voto e numero di esami con quel voto

From ESAME

group by Voto → avremo maggiori pic. vot. uguali : ne prendo 1 per ogni voto

Esercizio: ✓

SELECT ETA, max(redatto)

FROM PERSONA

WHERE ETA > 18

Group By ETA

È possibile applicare dei filtri sulla selezione dei gruppi dopo lo group by
Si utilizza having condizione

UNIONE

SELECT ...

UNION

SELECT ...

Union all mantiene i duplicati!

In SQL è possibile fare l'unione anche tra tabelle con attributi diversi prendendolo come risultato quelli della prima (saltarmente)

DIFERENZA

Lo sintassi è come quella della union, ma bisogna scrivere EXCEPT

INTERSEZIONE

INTERSECT, come sopra.

SELECT NIDIFICATE

Quelle interne sono racchiuse tra parentesi tonde !

es.

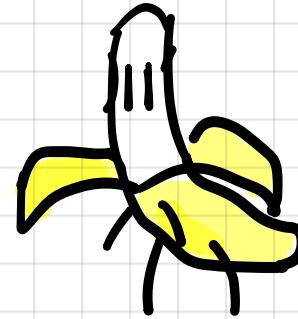
SELECT nome, reddito

FROM PERSONE

WHERE nome = (SELECT PADRE

FROM PATERNITÀ

WHERE FIGLIO = "FRANCO")



In questo caso lo seconda select restituisce un solo valore.

In caso contrario posso mettere

- =Any che ritorna vero se è vero per una qualunque tupla del risultato della select annidata

- =All che ritorna vero solo se è vero per ogni tupla risultante

in = =Any

not in = <> All

- EXISTS ritorna true se la select associata ritorna un insieme non vuoto

Select Nome, ETÀ

FROM Persone

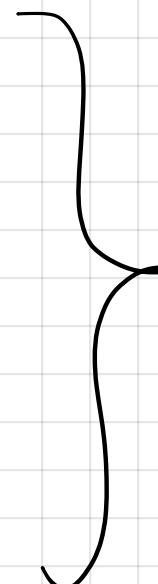
WHERE nome IN (Select madre

FROM maternità

WHERE figlio IN (SELECT nome

FROM PERSONE

WHERE ETÀ < 18))



Equivalenti

Select Nome, ETÀ

FROM maternità, persone p

WHERE madre = nome AND figlio IN (Select nome

FROM persone

WHERE età < 18)

Seppur più leggibili le query annidate possono creare problemi ai DBMS

Un elemento x è visibile in un modulo m se è stato dichiarato in m oppure x è visibile nel modulo in cui m è definito, e in m non è definito un altro elemento con lo stesso nome di x

es

Select nome, età

from persone, maternità

where nome = madre AND exists (Select * from persone

Where età < 18 AND figlio = nome

Nome viene attribuito alla tabella persone intorno!

```

Select Nome , ETÀ
From persone P
Where nome in (Select madre
    From maternità
    Where figlio in (SELECT nome
        From persone
        Where P.ETÀ - TIA < 20))

```

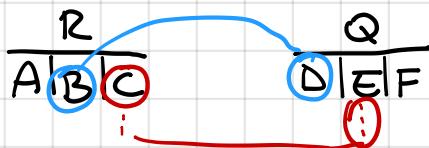
Le query indicate possono comparire anche nella from: vale la regola della visibilità delle query fatte nella where

Vincoli Integrità in SQL

(clausola check dopo il tipo dell'attributo nella create table o (sempre nella "create table" come vincolo intrazionalone)

es.1 Sesso character not null check(sesso in ('M','F'))

es.2 CHECK (STIPENDIO <= (SELECT ...))



Create table Q (...)

Create Table R (

A char(20) primary key

B char(20) references Q(D)

C number(100)

check (C in (select E from Q))

)

I vincoli di integrità si possono usare a posto
create assertion Nome check (Condizione)

VIEW

Tabelle virtuali

Create view NomeView [(listAttrv.)] or Select

Dopo aver creato lo view, NomeView potrà essere usata per calcolare lo select specificato.

PROGETTAZIONE CONCETTUALE → PROGETTAZIONE LOGICA

Modello ER

Modello entità - relazione

ENTITÀ:

classe di oggetti di interesse per l'applicazione, con esistenza autonoma e proprietà comuni
Ogni entità ha un nome che la identifica

Semantica:

SCHEMA S



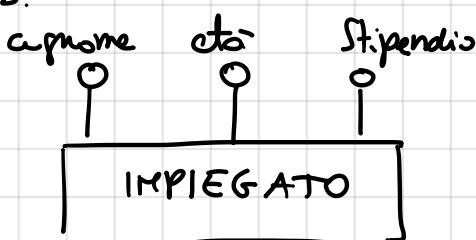
I(Istanze di S) = insieme di oggetti distinti (istanze di studente) = Istanze (I, Studente)

Istanze (I, Studente) è anche detta estensione di Studente nell'istanza I dello schema S

Attributo di Entità:

Proprietà di un' entità di interesse per l'applicazione

ES.

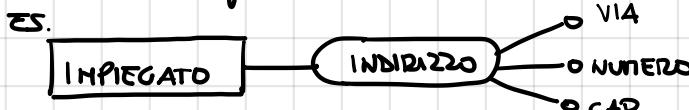


Il tipo dell' attributo può essere specificato:
cognome / stringa, ma solitamente è specificato
nel dizionario dei dati

Attributo: istanze (I, ε) → Dominio

L'attributo è una funzione

Attributi Composti



RELAZIONE:

Non presenta un legame tra entità. Il grado è dato dal numero delle entità coinvolte

ES.



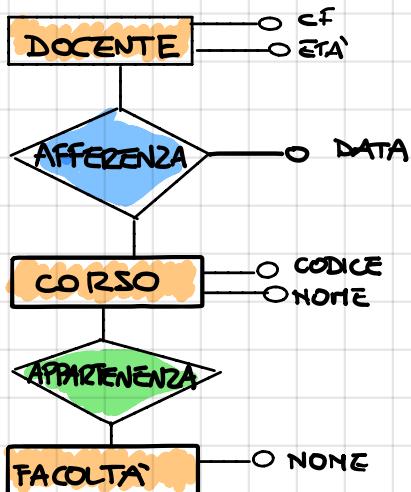
Semantica:

Relazione (E, F) = Insieme di coppie (x, y) tali che $x \in$ istanza di E e $y \in$ istanza di F

Stessa cosa vale con relazioni n-arie.

Gli attributi di relazione si esprimono come gli attributi delle entità

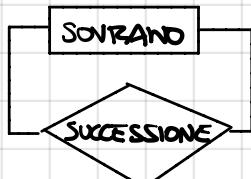
ES.



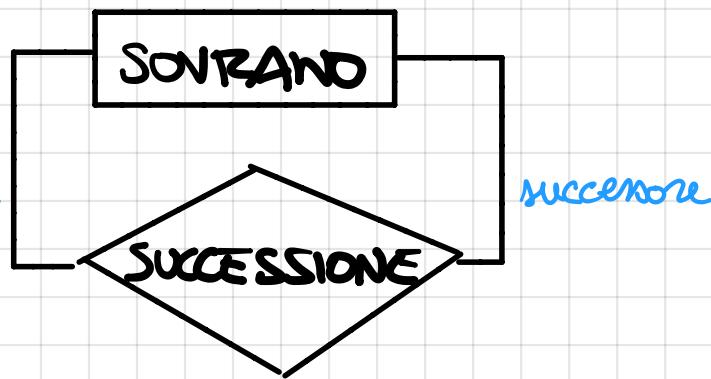
RUOLO:

Le relazioni possono coinvolgere due o più volte lo stesso entità

es.



In questo caso è necessario specificare dei ruoli per distinguere le entità

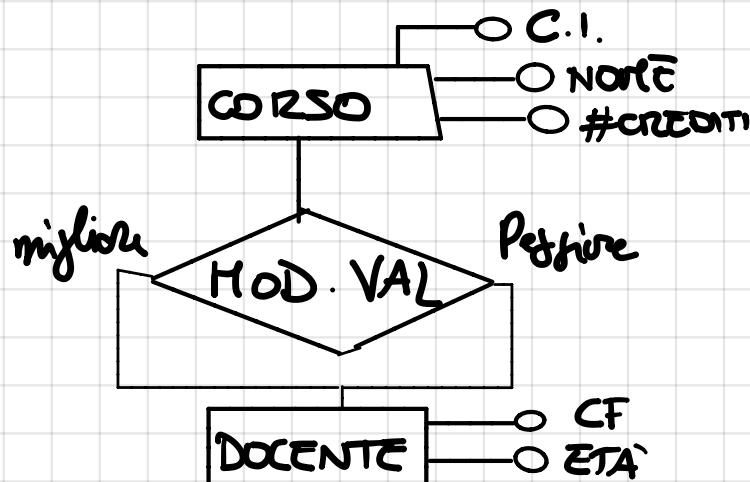


Si può scegliere uno dei due ruoli: in questo caso quello scelto prenderà il nome dell'entità stessa.

$$\text{ISTANZE}(\text{I}, \text{SONRANO}) = \{\text{Romolo, Numa, Tullio}\}$$

$$\text{ISTANZE}(\text{I}, \text{SUCCESSIONE}) = \{(\text{Predicatore: Romolo, Successe: Numa}), (\text{Predicatore: Numa, Successe: Tullio}), \dots\}$$

es.



RELAZIONE ISA TRA ENTITÀ



Le istanze di studente sono sempre sotto-istanze di persona
Un'entità può avere al massimo una entità padre

Istanze $(I, E_1) \subseteq$ Istanze (I, E_2)

La relazione ISA è riflessiva e transitiva

EREDITARIETÀ

Ogni proprietà del padre (attributi ecc) sono anche proprietà del figlio (nel diagramma non bisogna "ripeterle" per il figlio)

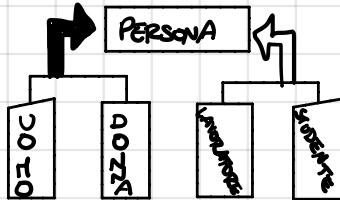
Un'entità può avere più entità figlie (generalizzazione)

La generalizzazione può essere completa (figlio 1 + ... + figlio n = padre) ↑

o non completa ↑↑

Le figlie sono posizioni del padre

Un'entità può essere padre di più generalizzazioni



Si può definire una relazione ISA tra relazioni se le due relazioni hanno stesso grado e ruoli.
e se l'entità corrispondente al ruolo della relazione "figlia" è ISA dell'entità relativa al ruolo della relazione "padre".

VINCOLI DI INTEGRITÀ NEL MODELLO ER

Vincoli di cardinalità: limite minimo e massimo alla partecipazione di una entità in un certo ruolo (relazioni)



Se non specifico è sottinteso (0,n)

I vincoli di cardinalità vengono ereditati dalle entità ISA

Se non specificate sono come le entità padre, altrimenti posso "specializzarle" ulteriormente
la cardinalità massima di due entità ISA è vincolata dal fatto che la massima del figlio

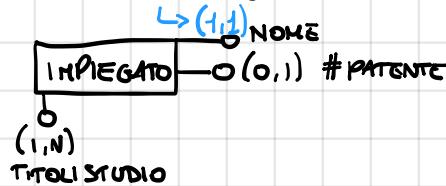
è \leq di quella del padre

Mentre le cardinalità minime non sono correlate.

Vincoli di Coordinanza (Attributi):

- indicare opzionalità
- indicare attributi multivalue

Se si mette 1 si considera (1,1)

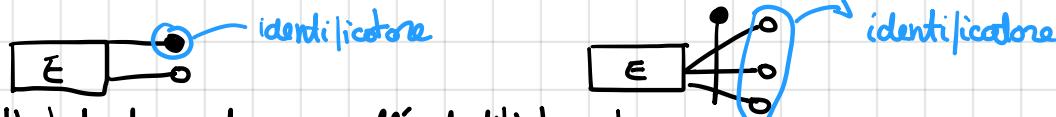


Vincolo di Identificazione (Entità):

proprietà che specificano due istanze della stessa entità (identificatore)

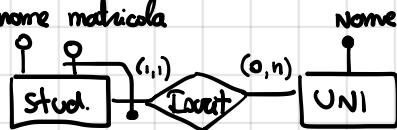
INTERNI: solo attributi dell'entità

ESTERNI: attributi dell'entità e/o ruoli



Attributi che partecipano all'identificatore hanno coordinanza (1,1) (stessa cosa per ruoli)

cognome matricola



Non esistono due studenti con stessa matricola e stessa università

Esempio:



L'identificatore è "collegato" con il vincolo di coordinanza di F (0..1)

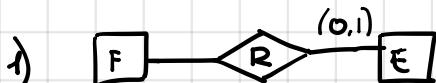
Vincoli di Identificazione (Relazione):

come per le entità, i vincoli di identificazione specificano diverse istanze di una relazione.

Per le relazioni vi è un identificatore implicito dato dai ruoli che non permettono "duplicate".

Gli identificatori di relazione "partono" direttamente dal rombo della relazione.

In alcuni casi vincoli di identificazione o coordinanza su entità implicano vincoli (chiamati) di identificazione nelle relazioni.

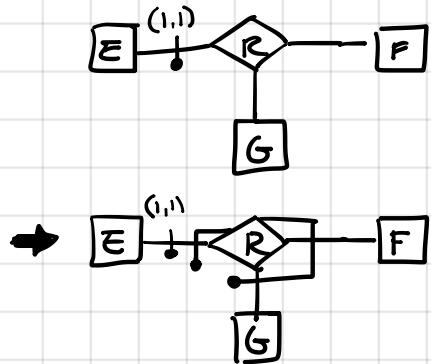


Primo caso: entità E partecipa ad una relazione R nel ruolo U con vincolo di coordinanza minima uguale a 1



N.B. possono essere omessi nello schema

2) Secondo caso: ϵ partecipa ad una relazione R nel ruolo U e tale ruolo è un identificatore per ϵ (quindi coordinate di U in Z = (1,1))

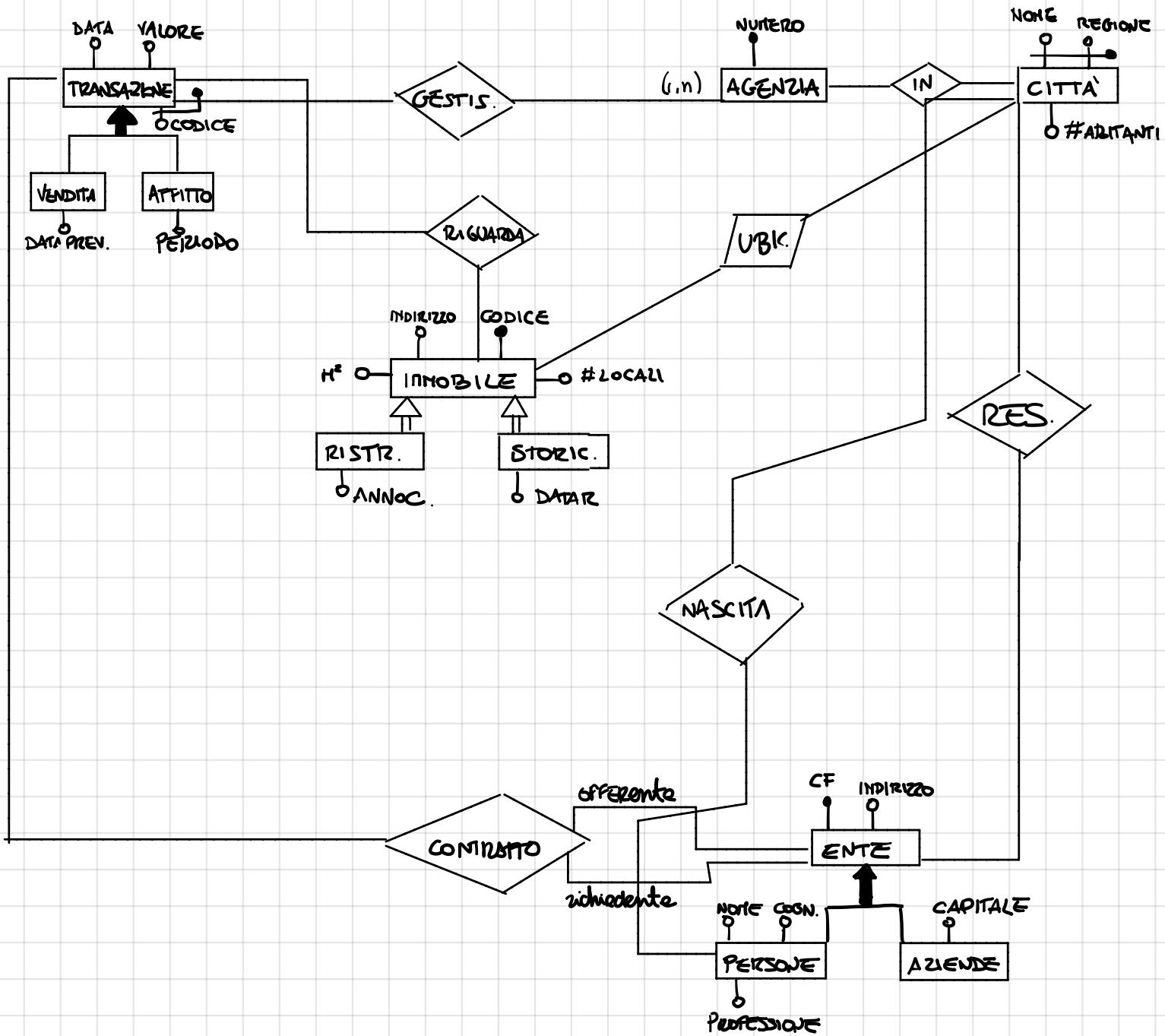


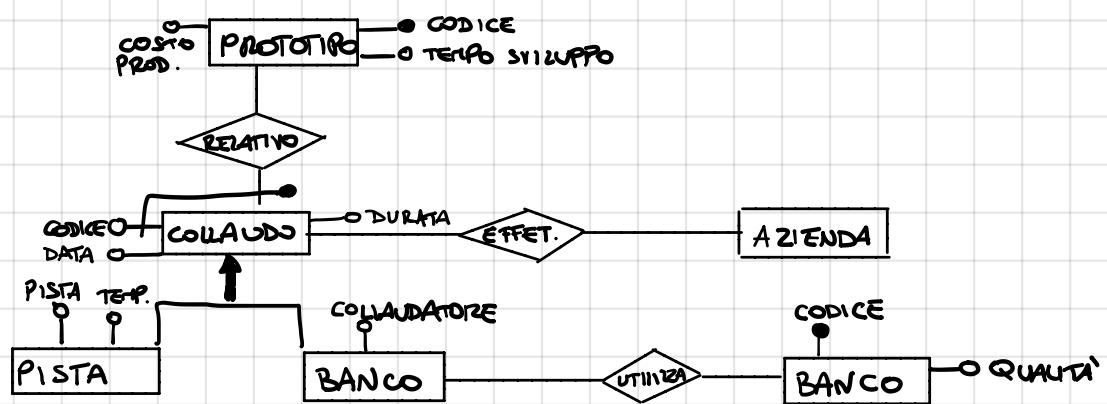
Tutte le "restrizioni" che non si possono esprimere tramite vincoli si esprimono attraverso formalismi opportuni (es. linguaggio matematico) o in italiano
Questi sono chiamati vincoli **esterni**

Lo schema concettuale è formato dal diagramma ER e il **dizionario dei dati**

Tale dizionario è composta da 4 tavole

- 1) Entità: nome, descrizione, attributi, identificatori
- 2) Attributi: Attributo, Entità/Relazione, descrizione, Dominio
- 3) Relazioni: Relazione, descrizione, componenti
- 4) Vincoli esterni





PROGETTAZIONE LOGICA

Trasformare lo schema per il database.

Schema concettuale + informazioni sul carico applicativo → schema logico + vincoli aggiuntivi + documentazione associata

Le informazioni sul carico applicativo indicano come verrà utilizzata la base di dati → renderla efficiente in base all'uso che se ne fa

Venne applicato un **modello di costo**:

- tempo di esecuzione
 - spazio di memoria
- ⇒ Bisogna conoscere il volume dei dati (# istanze e dimensione attributi) e le caratteristiche delle operazioni (tipo, frequenza e dati coinvolti)

La valutazione dovrà comunque approssimativa

TABELLA VOLUMI

Concetto	Costrutto	Volumi
nome 1	Entità	# 1
nome 2	Relazione	# 2

TABELLA OPERAZIONI

op	Tipo	Frequenza
1	Interattiva	tot / giorno
2	Batch	tot / sett.
...		

Per ogni operazione devo fare lo stima di accessi

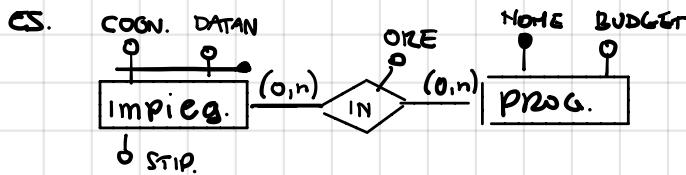
Concetto	Costrutto	Accessi	Tipo
nome	Entità/Rel.	#	L
...			→ Lettura

Obiettivo: ottimizzare le prestazioni!

D) Ristrutturazione modello ER

Eliminare (senza cambiare il senso dello schema) i costituti non riproducibili nello schema logico tenendo conto onde dell'efficienza

TRADUZIONE RELAZIONE



Impiegato (Cognome, DatoN, Stipendio)

Prog. (Nome, Budget)

IN (Cognome, DatoN, Orze, Progetto)

foreign key: IN[Progetto] ⊆ Progetto [Nome]

foreign key: IN [Cognome, DatoN] ⊆ Impiegato [Cognome, DatoN]

Eliminazione Attributi Multivalue



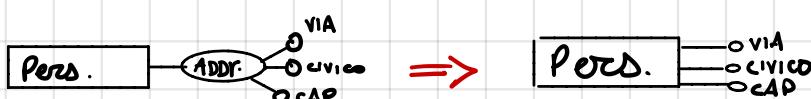
Attributi multivalue vengono scomposti in entità che contengono i valori e li si associa all'entità di portanza (Personna)

Per attributi multivalue sulle relazioni si trasforma la relazione in entità e si crea una nuova relazione come sopra. Le entità che sono collegate alla relazione verranno connesse ad essa con ulteriori relazioni nuove (a cascata)

Variante: Se lo cardinalità di tale attributo è (1,n) posso direttamente aggiungere un'entità che rappresenta l'attributo alla relazione

Se lo cardinalità è (1,1) posso fare lo stesso cosa ma devo partire dalle entità originali della relazione

Eliminazione Attributi Composti



+ Vincolo esterno per averli tutti e 3

Alternativa:



Eliminare ISA

Una relazione $E \text{ ISA } F$ è sostituita da una relazione in cui E ha cardinalità $(1,1)$ e $F(0,1)$ con un identificatore su E perché invoco

NB. $E \text{ ISA } F \Leftrightarrow \begin{matrix} 1 \\ E \end{matrix}$

Nel caso avessi due ISA sulla stessa entità con un attributo con lo stesso nome tradisco come spiegato, ma aggiungo anche un vincolo esterno per tale attributo

Eliminazione Generalizzazione

Come sopra, ma ho aggiunto un vincolo esterno che specifica la partecipazione di una singola relazione

NB. ISA e Generalizzazioni su relazioni sono espresse tramite vincoli esterni

Identificatori

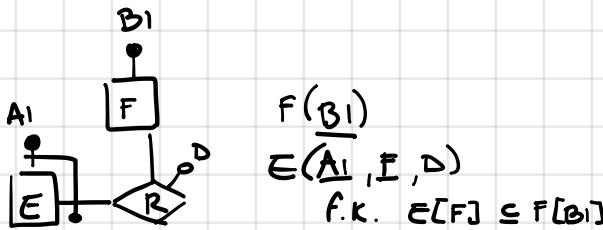
Gli identificatori di entità o relazione DEVE essere **essenziale**

Le entità devono avere almeno un identificatore e se sono più di uno bisogna sceglierne uno principale

TRADUZIONE ENTITÀ

Accorpamento

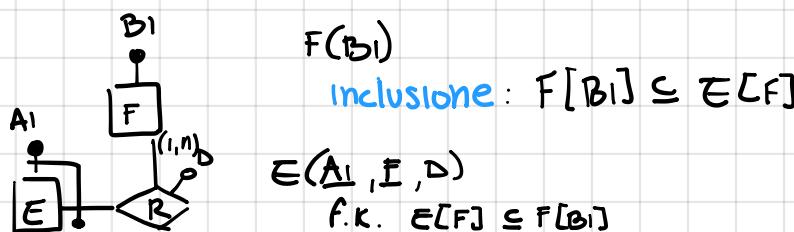
1) Se una relazione è parte di un' identificatore principale esterno di un' entità E con ruolo v e ha cardinalità $(0,n)$ su tutti gli altri ruoli essa viene accorpata ad E



$F(B_1)$
 $\in(A_1, F, \Delta)$
 f.k. $E[F] \subseteq F[B_1]$

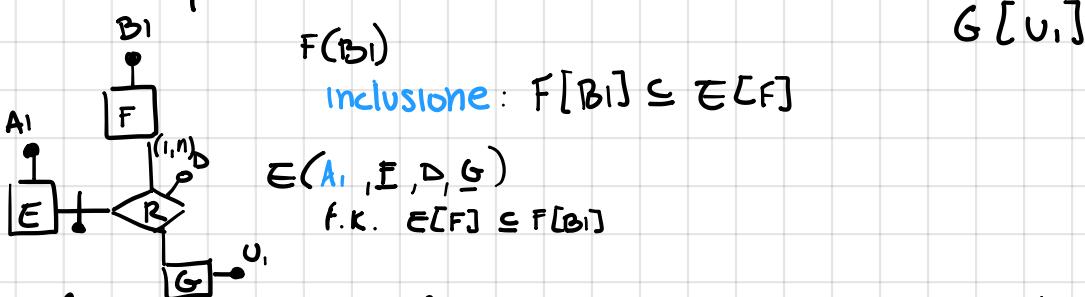
N.B. In E verranno inseriti come attributi gli attributi della relazione R e tutte le chiavi primarie delle entità che vi partecipano

2) Cosa come sopra ma con cardinalità $(1,n)$



$F(B_1)$
Inclusione: $F[B_1] \subseteq E[F]$
 $\in(A_1, F, \Delta)$
 f.k. $E[F] \subseteq F[B_1]$

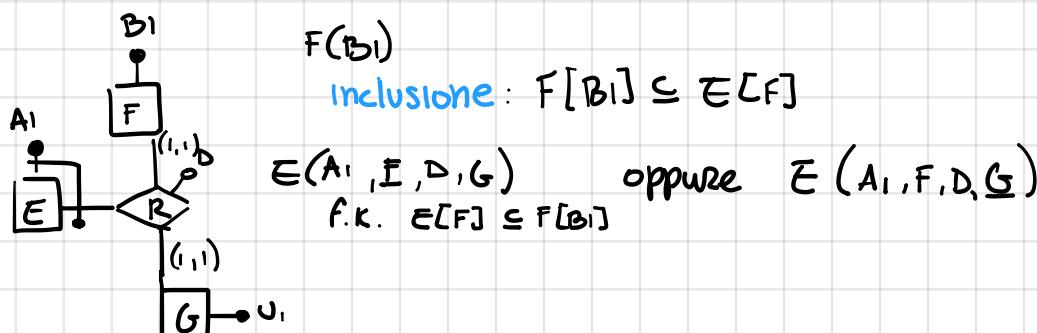
3) R è identificatore principale esterno (no "una pole") di un' entità E con ruolo v e ha cardinalità di qualche altro ruolo $\neq (1,n)$



$F(B_1)$
Inclusione: $F[B_1] \subseteq E[F]$
 $\in(A_1, F, \Delta, G)$
 f.k. $E[F] \subseteq F[B_1]$

Come il caso 1 e 2 ma lo chiave primaria di E non contiene attributi di E

4) R è identificatore principale esterno di un' entità E con ruolo v e danti dei ruoli hanno cardinalità minima 1



$F(B_1)$
Inclusione: $F[B_1] \subseteq E[F]$
 $\in(A_1, F, \Delta, G)$ oppure $\in(A_1, F, D, G)$
 f.k. $E[F] \subseteq F[B_1]$

Come i casi 1 e 2 ma la chiave primaria di E deve essere scelta tra le altre chiavi primarie delle entità coinvolte in R

Il caso 4 si affianca alla derivazione di relazioni ISA o generalizzazioni

Regole Generali

Ogni relazione Q dello schema entità-relazione non accorpata diventa una relazione nel schema relazionale che ha come attributi gli stessi attributi di Q e le chiavi primarie delle entità che partecipano a Q

Lo scelta della chiave primaria si basa sull'identificatore principale

JAVA DB CONNECTIVITY

Utilizzare un DBMS da software

Ulre dell'applicazione che usa le API e il DBMS che fornisce i dati abbiamo:

- Gestore dei driver
- Driver

I driver trasformano le richieste dell'opp. in un formato comune al DBMS e viceversa per le risposte/erori (library)

Nel driver manager viene caricato il driver opportuno al DBMS usato

Caricamento driver

```
Class.forName("nome.driver");
```

così facendo il driver viene caricato e istanziato nel driver manager. Da questo momento potremo usare il driver manager per aprire connessioni sul DBMS supportato dal driver caricato

Connessione

```
Connection conn = DriverManager.getConnection(url, username, pw);
```

URL = "Jdbc: sub-protocol: url database"

es. Jdbc:mysql://localhost:3306/test

Intercettore

L'intercettore avviene tramite classi che derivano da una "Connection"

• Statement

Oggetto che consente di eseguire una richiesta

es. executeUpdate (creazione, modifica, eliminazione)

executeQuery

Tali funzioni prendono in input una normale query sotto forma di stringa

• ResultSet

Statement.executeQuery("query") ritorna un ResultSet simile ad un cursore con lo risultante della query (parte da -1, ha il metodo next()...)

RS.getxxx(y) ritorna la colonna con nome y o di numero y NB. xxx è il tipo della colonna

Prepared Statement

Uno Statement che ommette parametri

Lo query sarà come al solito ma può avere dei "?" che verranno sostituiti a runtime con setxxx(n, v)

xxx come per getxxx

n = # ordine del "?"

v = valore da sostituire

Chiusura Connessione

```
Connection.close()
```

