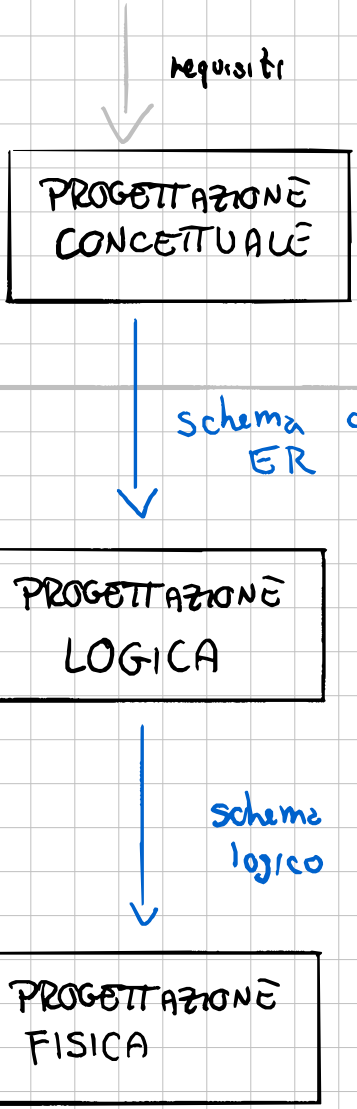


OBBIETTIVI

Tradurre schema concettuale ER in un'altro che segue la logica e i costrutt. del DBMS

DA SCHEMA ER

programmazione logica

A PROGETTAZIONE DBMS

Serve cambiare da oggetti concettuali nello schema E-R e ne produciamo una versione "ristrutturata", senza ISA, attributi multivalore..., non esprimibile come modello relazionale.

↳ da qui si va allo schema relazionale diretto (solo tabelle), per poi adottare l'ultima fase di miglioramento, per essere più efficiente nella base.

La semantica NON cambia! Esso è il principio della progettazione concettuale

"Dello schema ER non si butta via niente"

18:45

FASI:

1. Degradazione dello schema ER: Si ottiene schema ER ristrutturato, vol dire che nessuno confonderà lo schema ER da quello ristrutturato

Ha l'obbiettivo di semplificare lo schema ER, eliminando costrutti non traducibili nello schema ER
7 attività

1. analisi ridondanze
2. eliminazione attr. multivalore
3. eliminazioni attr. composti
4. eliminazione ISA e generalizzazione
5. Scelta di identificatori principali d. entità e relazione
6. Specifica di vincoli esterni

Preparazione ER.

-Per ogni entità E dello schema ER, i vincoli di id rilevanti per l'entità E vengono riportati nello schema, anche quello della relazione!

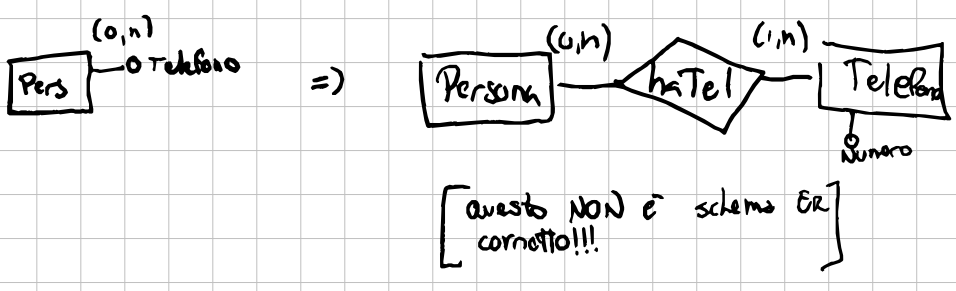
1. Analisi ridondanze.

Esistono ridondanze intensionali (che non si devono essere), ed estensionali (che posso decidere di lasciarle, scegliendo razionalmente)

2. Eliminazione attributo multivalore

Si trasforma l'attributo multivalore in entità in relazione binaria con l'attributo.
Stessa cosa può essere fatta nella relazione!

E' razionale pensarlo!

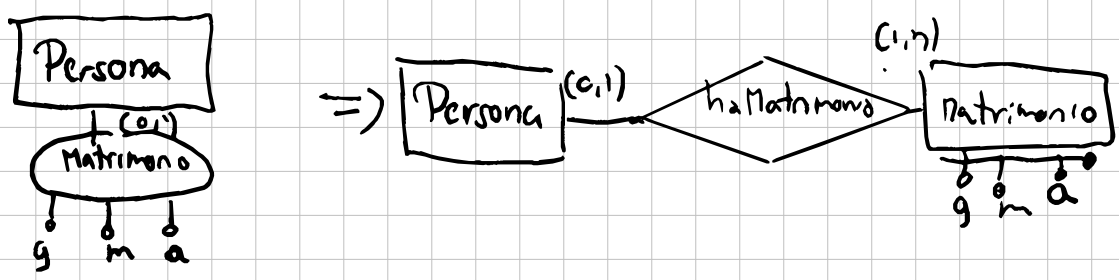


[questo NON è schema ER corretto!!!]

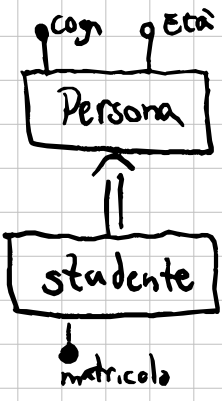
3. Eliminazione attributo composto

Se l'attributo ha cardinalità (1,1) si associano direttamente gli attributi componenti all'entità

Se invece si ha (0,1) si può rendere l'attributo composto un'entità, con gli attributi componenti come attributi di entità



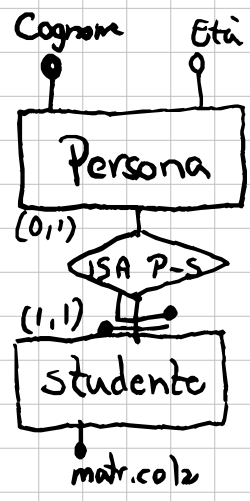
4. Eliminazione ISA



la metodologia è imporre di eliminare la relazione ISA

=)

si sostituisce la relazione ISA con una relazione.
Nella relazione di dx c'è disgiunzione!



ogni studente è in relazione con persone, ma NON È UNA PERSONA.
C'è sempre una degradazione, ma tutto della relazione ISA è mantenuto

Se le entità sono disgiunte nello schema originale, es

bisogna aggiungere vincoli esterni se c'è un ISA tra 2 entità ISA di una stessa entità.
Visto che non sono disgiunte e_1 ed e_2 devono avere lo stesso valore di attributi!

Anche nel caso di relazioni disgiunte si usa il vincolo esterno!

5 Scelta identificatori principali per entità e per relazione.
Esse saranno le chiavi primarie!

Ogni identificatore principale di entità o di relazione dovesser essenziale!

Per ogni entità è necessario:

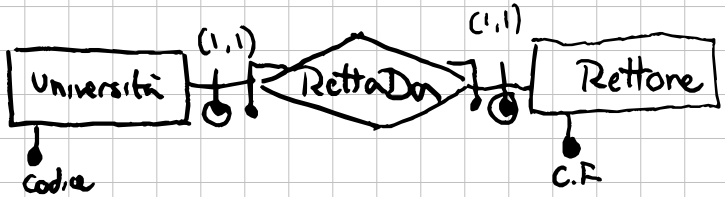
- Individuare almeno un identificatore (anche inventandosi codice)
- Scegliere tra gli identificatori quello essenziale

Criterio:
- essenzialità
- semplicità
- preferenza tra identificatori interni
- per quelli esterni preferire identificatori derivati da isa

Sull' identificatore scelto si fa un cerchio

Cicli tra identificatori principali esterni

Si supponga di avere una situazione simile



Essa non è dinamica

configurazione da ESCLUDERE
Non possono esistere cicli nell' identificatore principale!
[verificare cicli con grafo di identificatori esterni]

Scelta d. id per relazioni

Se nella relazione R partecipa ad identificatori principali esterni d. E, allora non serve, id. d. E è già d. R

Se non ce ne sono per E allora si possono usare quelli impliciti

Esempio slide Progettazione logica - 44

6. Specifica vincoli esterni

Se ci sono state modifiche d. qualcosa che tocca i vincol. esterni bisogna modificarla

TRADUZIONE DIRETTA

ha lo scopo di tradurre lo schema ER ristrutturato, che quindi ha ora tutti i suoi costrutti esprimibili nel modello relazionale, in uno schema direttamente traducibile per dbms

consiste nelle attività di

1. Traduzione delle entità
2. Traduzione delle relazioni
3. Traduzione di vincoli esterni
4. Riformulazione delle operazioni specifiche

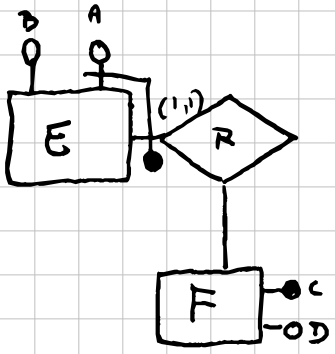
Traduzione di entità

Ogni entità dello schema viene tradotta in una relazione gli attributi sono:

- gli attributi in E (tutti non null tranne se con card (0,1))
- gli attributi derivanti dall'accorpamento di ER-relazioni di esse vengono assunte:
  - chiavi primarie delle relazioni da accorpare che corrispondono alle entità partecipanti
  - eventuali attributi della relazione

vincoli possibili richiesti

- primary key (per vincolo di id principali)
- key (per vincoli di cardinalità massima 1 dell'entità partecipante a R relazione accorpata, o per altri vincoli di id non partecipanti)
- include (per vincoli di cardinalità minima 1,)
- foreign key (per rappresentare partecipazione obbligatoria dell'entità alle relazioni)



istanze (1, F) = $\{f_1\}$
istanze (1, E) = $\{e_1, e_2\}$

istanze (1, A) = $\{\langle e_1, 10\rangle, \langle e_2, 20\rangle\}$

istanze (1, R) = $\{\langle E: e_1, F: f_1\rangle, \langle E: e_2, F: f_1\rangle\}$

istanze (1, C) = $\{\langle f_1, c_1\rangle\}$
istanze (1, D) = $\{\langle f_1, d_1\rangle\}$
istanze (1, B) = $\{\langle e_1, b_1\rangle, \langle e_2, b_2\rangle\}$

F

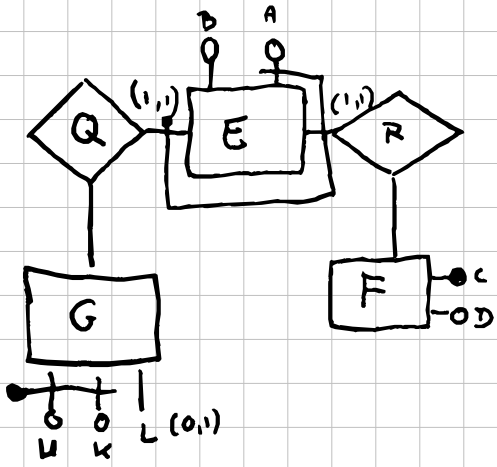
C	D
c_1	d_1

E

A	F	B
10	c_1	d_1
20	c_1	b_2

foreign key $E[F] \subseteq F[C]$

Se c_1 fosse vincolo esterno di relazione è facile



F(C, D)

G(H, K, L*)

E(A, F, GH, GK, B)

foreign key $E[F] \subseteq F[C]$
foreign key $E[GH, GK] \in G[H, K]$

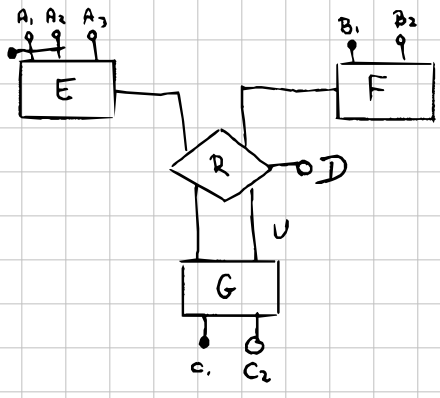
Se Nella relazione Q nel ruolo g ci sia (1, n) si cambia il vincolo della inclusione $G[H, K] \subseteq E[GH, GK]$

Non tutte le inclusioni sono foreign key!

Nel caso in cui R o Q abbia attributi di relazione esse saranno accorpate in E

Si chiameranno SUPERCHIAVI IMPLICITE, nelle relazioni non accorpate ad una tabella, le chiavi primarie delle entità che partecipano alla ER-relazione.

ESEMPIO



$\Rightarrow$

$E(\underline{A_1}, \underline{A_2}, A_3)$

$F(\underline{B_1}, B_2)$

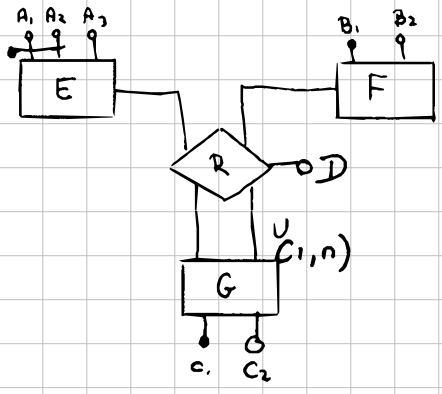
$G(\underline{C_1}, C_2)$

$R(\underline{EA_1}, \underline{EA_2}, \underline{FB_1}, \underline{GC_1}, \underline{UC_1}, D)$

ci servono vincoli di foreign key su R

$$\begin{bmatrix} \text{foreign key } R[EA_1, EA_2] \subseteq E[A_1, A_2] \\ \text{foreign key } R[FB_1] \subseteq F[B_1] \\ \text{foreign key } R[GC_1] \subseteq G[C_1] \\ \text{foreign key } R[UC_1] \subseteq G[C_1] \end{bmatrix}$$

complichiamo lo schema:

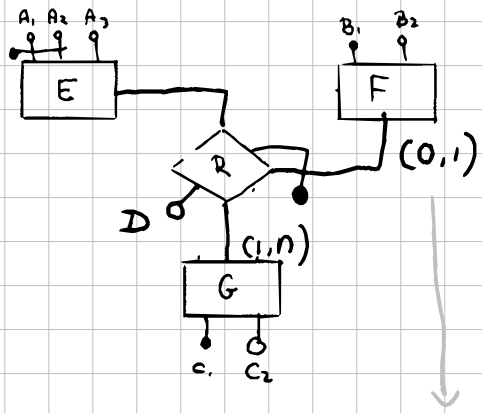


c'è un vincolo di cardinalità sulla partecipazione di G nel ruolo U in R

devo aggiungere vincolo di inclusione su G

$G(\underline{C_1}, C_2)$ → inclusione $G[C_1] \subseteq R[U]$

ESEMP. 2



$E(\underline{A_1}, \underline{A_2}, A_3)$

$F(\underline{B_1}, B_2)$

$G(\underline{C_1}, C_2)$

$R(EA_1, EA_2, \underline{F}, G, D)$

l'identificatore principale cambia! Non è implicito

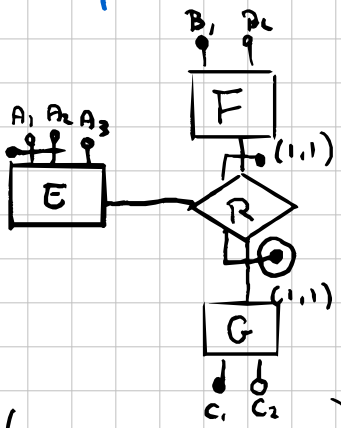
le foreign key rimangono tutte!

$$\begin{bmatrix} \text{foreign key } R[EA_1, EA_2] \subseteq E[A_1, A_2] \\ \text{foreign key } R[FB_1] \subseteq F[B_1] \\ \text{foreign key } R[GC_1] \subseteq G[C_1] \end{bmatrix}$$

su G ci sono le inclusioni

inclusione $G[C_1] \subseteq R[GC_1]$

complichiamo un.n



$R(EA_1, EA_2, F, \underline{G}, D)$

(per vincoli R)

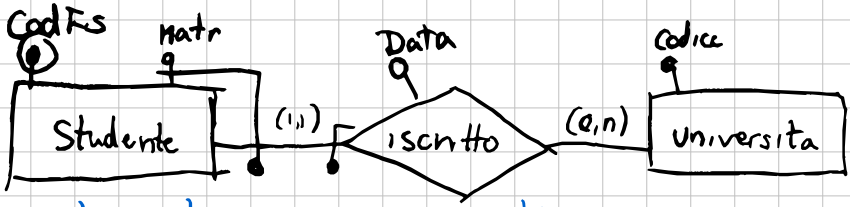
foreign key $R[EA_1, EA_2] \subseteq E[A_1, A_2]$
foreign key $R[FB_1] \subseteq F[B_1]$
foreign key $R[GC_1] \subseteq G[C_1]$
chiave F

G è identificatore! F è chiave ma non primaria

F è chiave non primaria in R, quindi si usa vincolo chiave

$F(\underline{B_1, B_2})$ foreign key $F[B_1] \subseteq R[F]$

$G(\underline{C_1, C_2})$ foreign key $G[C_1] \subseteq R[G]$



Vincol. esterni non principali:

iscritto (~~Studente~~ codFis, Universita, Data)
  foreign ...
  foreign ...
  key (matr, universita)

studente (codFis, Matr)
  foreign Studente[codFis] ⊂ iscritto[studente]

universita (Codice)

serve VINCOLO esterno, attuabile con trigger:

"nell' equigoin tra studente ed iscritto negli attributi codFis e studente, gli attributi Matricola e Universita formano una chiave"

Modello di costo

Ci possono essere dei problemi di traduzione in SQL riguardante l'EFFICIENZA, in modo da trovare la MIGLIORE rappresentazione

Ad esempio, ci potrebbero essere problemi per:
- valori null:
- 2 relazioni R_1 ed R_2 con chiavi K_1 e K_2 tali che valga $R_1[K_1] \subseteq R_2[K_2]$ e $R_2[K_2] \subseteq R_1[K_1]$
- ridondanze lasciate nelle tabelle

Cosa succede nel DBMS?

file nella memoria secondaria allocata al DBMS, dove saranno inseriti tutti i DATABASE.

↳ Ogni DATABASE / TABELLA ha un file di riferimento
ogni file è fatto di PAGINE, ognuna con un'indirizzo, che al suo tempo contengono RECORD

Ciò è importante perché l'accesso al record è possibile unicamente LEGGENDO / SCRIVENDO un'intera pagina.

↳ VERO OBBIETTIVO di ottimizzazione è sul numero di trasferimenti

Per una relazione R denotiamo con

- $N_P(R)$ numero di pagine in memoria occupate da R
- $N_{TP}(R)$ numero di tuple per ogni pagina.

Alcune stime

- la selezione / proiezione su R ha costo $N_P(R)$
- il join di R con Q si basa su un doppio ciclo:
  - se non ci sono indici vi sarà il prodotto cartesiano: costo $N_P(R) \times N_P(Q)$

Criteri generali per trovare problemi

- Relazioni con
  - tante tuple
  - tanti attributi
- Attributi con tanti valori null:
- Proiezione è costosa
- Join costoso (quasi sempre)
- verifica vincoli (quasi sempre)

Per una relazione R rilevante occorre tentare di tenere basso $N_P(R)$, al limite aumentando $N_{TP}(R)$.

Aumentare $N_{TP}(R)$ può significare SPEZZARE le relazioni. E per eliminare la necessità di join costosi si potrebbe accorpare le relazioni.

- Operazioni di decomposizione
- Operazioni di accorpamento

In entrambi i casi, le relazioni originarie possono essere ricostruite attraverso la definizione di opportune view

DECOMPOSIZIONE VERTICALE

R

$\underline{k}$	α	β

$\Rightarrow$

R1

$\underline{k}$	α

R2

$\underline{k}$	β

↑ uguali ↑

[Si fa questo per tabelle RILEVANTI (molte tuple)]

Si applica quando gli accessi ad R avvengono prevalentemente in modo separato sugli attributi α rispetto agli attributi β. - Raramente servono entrambe -

DECOMPOSIZIONE ORIZZONTALE per facilitarne l'accesso

$\underline{k}$	α	β

$\Rightarrow$

$\underline{k}$	α	β
k_{11} ⋮ k_{1n}		

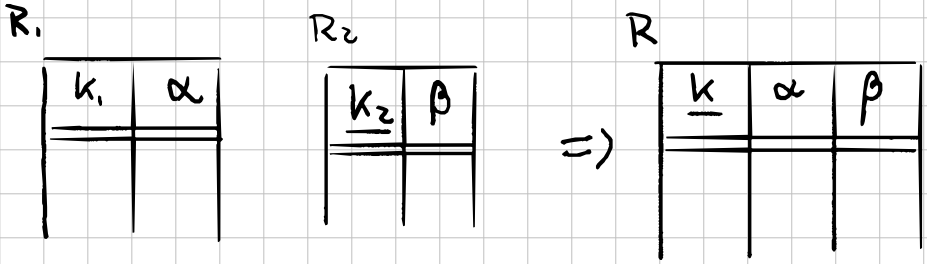
$\underline{k}$	α	β
k_{21} ⋮ k_{2n}		

RISTRUTTURAZIONE SCHEMA LOGICO

Da fare secondo il carico applicativo, riadattandolo secondo requisiti particolari.

- **operazioni di accorpamento**
  - unione di 2 o più relazioni al fine di produrne da esse una relazione più performante a livello di accesso di pagine.

(la vista deve essere sempre costruibile)



dalla chiave della K_1 ci sta una foreign key su K_2 E VICEVERSA.
Utile quando, se si usa R_1, si fa spesso un join con R_2.
Si risolve unendo una volta sola
- ovviamente vincoli su R_1 ed R_2 dovranno anch'essi essere accorpati -

Nel caso di relazioni debolmente accoppiate possiamo comunque mettere un attributo che può essere NULL (outer join) In caso di attributi (0,1) può essere utile aggiungere un FLAG che mi dice se una tupla della relazione padre è anche una della rel. figlia. (c verificare con vincolo di integrità eventuali attributi con CHECK).

Le tabelle sono lascamente accoppiate quando un'attributo che non è chiave è foreign key dell'altra.
L'operazione necessaria da fare è di accorpare la relazione R_1 principale SENZA eliminare la R_2.

Si chiama "accorpamento per denormalizzazione"