Basi di Dati
La teoria della normalizzazione
-- terza parte --

Corso B

Esempio

ESAMI(<u>MATR</u>, NomeS, IndirizzoS, CAPS, CodiceFiscaleS, DataNascitaS, <u>Corso</u>, Voto, DataEsame, CodProf, NomeProf, Qualifica, TipoUfficio)

Notazione abbreviata:

ESAMI(<u>MATR</u>, NS, IS, CAP, CF, DN, <u>Co</u>, Vo, DE, CP, NP, Q, TU)

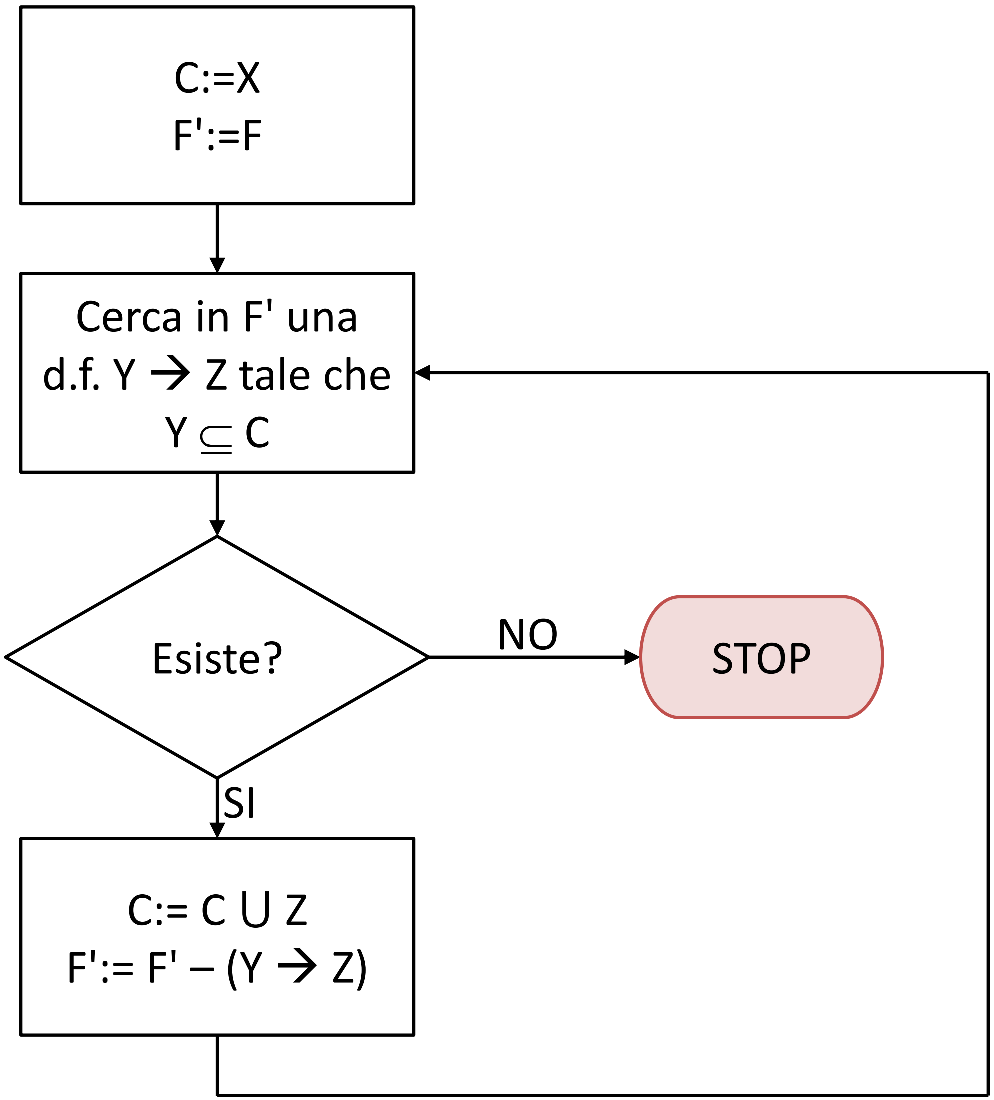
Esempi di vincoli

ESAMI(<u>MATR</u>, NS, IS, CAP, CF, DN, <u>Co</u>, Vo, DE, CP, NP, Q, TU)

Dipendenze funzionali:

- MATR → NS, IS, CF, DN
- CF → MATR
- IS → CAP
- MATR, Co → Vo, DE, CP
- CP → NP, Q
- Q → TU
- CP → Co

Algoritmo per il calcolo di X^+



Forma normale BCNF

Data una relazione *R(A)* in 1NF abbinata a delle dipendenze funzionali *F*, la relazione è in **BCNF** se, per ogni $X \rightarrow Y \in F$ si verifica una delle seguenti condizioni:

1. $Y \subseteq X$ ($X \rightarrow Y$ è una dipendenza **riflessiva**)
2. X è **superchiave** di R

Normalizzazione in BCNF

1. Parto dallo schema di basi di dati $SC = \{(R_i(A_i),F_i)\}$
2. Cerco, all'interno di SC, una $(R_i(A_i),F_i)$ non in BCNF
   – *cioè, cerco in F_i almeno una d.f. $X \rightarrow Y$ non BCNF ovvero*

     $(Y \not\subset X) \wedge (X \text{ non superchiave di } R_i(A_i))$

3. Se non esiste una $X \rightarrow Y$ non BCNF mi fermo (*SC* è già in BCNF)
4. Se esiste una $X \rightarrow Y$ non BCNF occorre effettuare una trasformazione dello schema
   a. Tolgo la relazione non in BCNF da SC
      - $SC := SC - (R_i(A_i),F_i)$
   b. Aggiungo ad SC due nuove relazioni con le relative dipendenze funzionali
      - $SC := SC \cup (R_1(\mathbf{A_i - Y}), F_1)$
      - $SC := SC \cup (R_2(\mathbf{XY}), F_2)$

      dove F_1 è la restrizione di F_i in $R_1(A_i - Y)$ ed F_2 è la restrizione di F_i in $R_2(XY)$
   c. Torno al passo 2 della procedura

III^a Forma Normale (3NF)

Una relazione *(R(A),F)* è in **3NF** (**III^a forma normale**) se, per ogni ogni $X \rightarrow Y \in F$ si verifica una delle seguenti condizioni

1. $Y \subseteq X$ ($X \rightarrow Y$ è **riflessiva**)
2. X è **superchiave**
3. Y sono **attributi primi**

Attributi primi

Riconsideriamo questo esempio:

ESAMI(<u>MATR</u>, <u>Co</u>, Vo, CP)

F = {MATR, Co → Vo, CP; CP → Co}

con la chiavi {MATR,Co}, {MATR,CP}

Considero CP → Co

Il conseguente {Co} è sintatticamente contenuto dentro una chiave della relazione ESAMI

Si dice che Co è un **attributo primo**

Attributi primi (definizione)

Gli attributi Y $\subseteq$ A sono detti **attributi primi**, se $Y \subseteq K$, dove *K* è chiave di *R(A)*

III^a forma normale

La 3NF è meno restrittiva della BCNF, in quanto ammette come dipendenze funzionali, oltre a quelle BCNF, anche quelle il cui conseguente è costituito da attributi primi

Quindi l'esempio: ESAMI(<u>MATR</u>, <u>Co</u>, Vo, CP), con
F = {MATR, Co → Vo, CP; CP → Co} è un esempio di relazione in 3NF, infatti

- MATR,Co → Vo è una d.f. BCNF
- CP → Co è una d. f. 3NF (Co è parte di una chiave)

Esempio

CC(Titolare,NConto,NAgenzia,CittaAgenzia,Direttore,Saldo)

Con F = {

NConto → NAgenzia,CittaAgenzia,Direttore,Saldo;
CittaAgenzia, NAgenzia → Direttore;
Direttore → CittaAgenzia

}

Un direttore può essere direttore di più agenzie purché siano agenzie della stessa città

Esempio

CC(Titolare,NConto,NAgenzia,CittaAgenzia,Direttore,Saldo)

Con F = {

NConto → NAgenzia,CittaAgenzia,Direttore,Saldo;
CittaAgenzia, NAgenzia → Direttore;
Direttore → CittaAgenzia

}

La chiave è {Titolare,NConto}, quindi nessuna delle d.f. di F è in BCNF

Esempio

CC(Titolare,NConto,NAgenzia,CittaAgenzia,Direttore,Saldo)

Con F = {

NConto → NAgenzia,CittaAgenzia,Direttore,Saldo;
CittaAgenzia, NAgenzia → Direttore;
Direttore → CittaAgenzia

}

Decomposizione in BCNF: iniziamo dalla seconda d.f.

AGENZIE(NAgenzia,CittaAgenzia,Direttore)

F_A = { NAgenzia,CittaAgenzia → Direttore;
Direttore → CittaAgenzia }

Esempio

AGENZIE(NAgenzia,CittaAgenzia,Direttore)

F_A = { NAgenzia,CittaAgenzia → Direttore;
Direttore → CittaAgenzia }

AGENZIE non è in BCNF, ma il conseguente della seconda d.f. {CittaAgenzia} è un attributo primo, quindi AGENZIE è in 3NF

Rimane CC(Titolare,NConto,NAgenzia,CittaAgenzia,Saldo)

Con F = {NConto → NAgenzia,CittaAgenzia,Saldo}

La dipendenza funzionale non è in BCNF

Esempio

AGENZIE(NAgenzia,CittaAgenzia,Direttore)

F_A = { NAgenzia,CittaAgenzia → Direttore;
Direttore → CittaAgenzia }

Scompongo in

StatoCC(NConto,NAgenzia,CittaAgenzia,Saldo) con

F = {NConto → NAgenzia,CittaAgenzia,Saldo} quindi in BCNF

e

TITOLARI(Titolare,NConto) in BCNF

Esempio

AGENZIE(NAgenzia,CittaAgenzia,Direttore)

F_A = { NAgenzia,CittaAgenzia → Direttore;
Direttore → CittaAgenzia }

Non è in BCNF, ma è in 3NF

Scomponendo ulteriormente per ottenere solo relazioni in BCNF perdo la località delle dipendenze funzionali (**esercizio**)

Osservazioni

- Una relazione in 3NF ha delle anomalie: ad esempio Direttore → CittaAgenzia o CP → Co sono d.f. che generano anomalie (la dipendenza funzionale di tipo 3 induce delle anomalie rispetto alla BCNF)
  - Sono però anomalie **tollerate**

- Se una relazione *R(A)* è in BCNF, allora *R(A)* è anche in 3FN (il viceversa non è vero)

Vantaggio della 3NF

Così come per la BCNF c'è un algoritmo per la normalizzazione, anche per la 3NF abbiamo un algoritmo per passare da qualsiasi schema di basi di dati in schemi in 3NF con però delle proprietà interessanti

Sintesi di una relazione in 3NF

La tecnica per trasformare schemi di relazioni in schemi in 3NF (processo detto sintesi in 3NF), parte da una manipolazione delle dipendenze funzionali

Le dipendenze funzionali si possono trasformare in un insieme di dipendenze F' minimale

Insieme di copertura minimale (definizione)

Un insieme F' di dipendenze funzionali è un **insieme di copertura minimale** rispetto ad F quando

1. $F' \equiv F$ (F' è equivalente ad F)
2. ogni $X \rightarrow Y \in F'$ è in forma canonica, cioè Y è un attributo
3. ogni $X \rightarrow Y \in F'$ è priva di attributi estranei
4. ogni $X \rightarrow Y \in F'$ non è ridondante

Insieme di copertura minimale (definizione)

Un insieme F' di dipendenze funzionali è un **insieme di copertura minimale** rispetto ad F quando

2. ogni $X \rightarrow Y \in F'$ è in forma canonica, cioè Y è un attributo

Il punto 2 non è un vincolo, ma solo uno stratagemma per semplificare la trattazione

Esempio

Considero la d.f. MATR,Co $\rightarrow$ Vo,CP

La forma canonica è:

MATR,Co $\rightarrow$ Vo

MATR,Co $\rightarrow$ CP

Ci si arriva per decomposizioni successive

Insieme di copertura minimale (definizione)

Un insieme *F'* di dipendenze funzionali è un **insieme di copertura minimale** rispetto ad *F* quando

1. *F'* è equivalente ad *F*
2. ogni $X \rightarrow Y \in F'$ è in forma canonica, cioè *Y* è un attributo
3. **ogni $X \rightarrow Y \in F'$ è priva di attributi estranei**
4. **ogni $X \rightarrow Y \in F'$ non è ridondante**

3. Attributo estraneo (ripasso)

Abbiamo le seguenti dipendenze funzionali F

- $ABCD \rightarrow E$
- $B \rightarrow C$ (data o dedotta dalle dipendenze date, ad esempio per transitività)

allora **C è un attributo estraneo** e si può cancellare, la dipendenza è allora $ABD \rightarrow E$

4. Dipendenze ridondanti

Consideriamo l'insieme di dipendenze

$$F = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$$

$A \rightarrow C$ è **ridondante** poiché è deducibile dalle dipendenze funzionali $A \rightarrow B$ e $B \rightarrow C$ per transitività

Algoritmo per il calcolo di una copertura minimale

Consideriamo un insieme F in forma canonica

1. F' := F
2. per ogni $X \rightarrow A_i \in F'$
3. per ogni $B \in X$
4. se $A_i \in (X - B)^{+}_{F'}$ allora
5. cancella B da X e aggiorna F'
6. per ogni $X \rightarrow A_i \in F'$
7. $F^* = F' - (X \rightarrow A_i)$
8. se $A_i \in X^{+}_{F^*}$ poni $F' := F^*$

Commento all'algoritmo

Alla riga 1, inizializzo la variabile F' ponendola uguale ad F

L'algoritmo è divisa in due parti:

- **Righe 2 – 5: elimina gli attributi estranei**
- Righe 6 – 8: elimina le d.f. ridondanti

Algoritmo per il calcolo di una copertura minimale

Consideriamo un insieme F in forma canonica

1. F' := F
2. **per ogni $X \rightarrow A_i \in F'$**
3. **per ogni $B \in X$**
4. **se $A_i \in (X - B)^+_{F'}$ allora**
5. **cancella B da X e aggiorna F'**
6. per ogni $X \rightarrow A_i \in F'$
7. $F^* = F' - (X \rightarrow A_i)$
8. se $A_i \in X^+_{F^*}$ poni $F' := F^*$

Commento all'algoritmo

Alla riga 1, inizializzo la variabile F' ponendola uguale ad F

L'algoritmo è divisa in due parti:

- Righe 2 – 5: elimina gli attributi estranei
- **Righe 6 – 8: elimina le d.f. ridondanti**

Algoritmo per il calcolo di una copertura minimale

Consideriamo un insieme F in forma canonica

1. F' := F
2. per ogni $X \rightarrow A_i \in F'$
3. per ogni $B \in X$
4. se $A_i \in (X - B)^+_{F'}$ allora
5. cancella B da X e aggiorna F'
6. **per ogni $X \rightarrow A_i \in F'$**
7. **$F^* = F' - (X \rightarrow A_i)$**
8. **se $A_i \in X^+_{F^*}$ poni $F' := F^*$**

Esempio (attributi estranei)

F' = {ABCD → E, B → C}

L'algoritmo si chiede se A è estraneo:

- Calcola la chiusura di {ABCD} – {A} rispetto ad F': calcolo quindi la chiusura di BCD
  - Posso usare solo B → C, ma C è già nella chiusura, quindi $BCD^+ = BCD$
- Si verifica che il conseguente (E) sia contenuto nella chiusura
  - Non lo è, quindi A non è estraneo (esamino il successivo)

Esempio (attributi estranei)

$F' = \{ABCD \rightarrow E, B \rightarrow C\}$

L'algoritmo si chiede se B è estraneo:

- Calcola la chiusura di {ABCD} – {B} rispetto ad F': calcolo quindi la chiusura di ACD
  - Non ho dipendenze da usare, quindi $ACD^+ = ACD$
- Si verifica che il conseguente (E) sia contenuto nella chiusura
  - Non lo è, quindi B non è estraneo (esamino il successivo)

Esempio (attributi estranei)

F' = {ABCD → E, B → C}

L'algoritmo si chiede se C è estraneo:

- Calcola la chiusura di {ABCD} – {C} rispetto ad F': calcolo quindi la chiusura di ABD
  – Uso B → C, e trovo quindi ABD^+ = ABCD
  – Uso ABCD → E e trovo quindi ABD^+ = ABCDE
- Si verifica che il conseguente (E) sia contenuto nella chiusura
  – E $\in$ ABCDE, quindi C è **estraneo e posso cancellarlo**

Esempio (attributi estranei)

F' = {ABD $\rightarrow$ E, B $\rightarrow$ C}

L'algoritmo si chiede se D è estraneo:

- Calcola la chiusura di {ABD} – {D} rispetto ad F': calcolo quindi la chiusura di AB
  - Posso usare solo B $\rightarrow$ C, quindi AB^+ = ABC
- Si verifica che il conseguente (E) sia contenuto nella chiusura
  - Non lo è, quindi D non è estraneo

Altro esempio

F' = {AB $\rightarrow$ C, B $\rightarrow$ C}

L'algoritmo si chiede se A è estraneo:

- Calcola la chiusura di {AB} – {A} rispetto ad F': calcolo quindi la chiusura di B
  - Posso usare solo B $\rightarrow$ C, quindi B^+ = BC
- Si verifica che il conseguente (C) sia contenuto nella chiusura
  - C $\in$ BC, quindi A è **estraneo e aggiorno F'**

Altro esempio

F' = {B $\rightarrow$ C, B $\rightarrow$ C}, ma F' è un insieme

quindi:

F' = {B $\rightarrow$ C}

Algoritmo per il calcolo di una copertura minimale

Consideriamo un insieme F in forma canonica

1. F' := F
2. per ogni $X \rightarrow A_i \in F'$
3. per ogni $B \in X$
4. se $A_i \in (X - B)^+_{F'}$ allora
5. cancella B da X e aggiorna F'
6. per ogni $X \rightarrow A_i \in F'$
7. $F^* = F' - (X \rightarrow A_i)$
8. se $A_i \in X^+_{F^*}$ poni $F' := F^*$

Esempio (dipendenze ridondanti)

F' = {A → B, B → C, A → C}

L'algoritmo si chiede se A → B è ridondante

- cancella A → B dall'insieme, quindi abbiamo
  F*= {B → C, A → C}
- calcola la chiusura di A rispetto ad F*:
  - Uso A→ C quindi A^+ = AC
- Verifica che il conseguente (B) sia contenuto in A+
  - Non lo è, quindi A→ B non è ridondante
- Si verifica lo stesso per B → C

Esempio (dipendenze ridondanti)

F' = {A → B, B → C, A → C}

L'algoritmo si chiede se A → C è ridondante

- cancella A → C dall'insieme, quindi abbiamo
  F*= {A → B, B → C}
- calcola la chiusura di A rispetto ad F*:
  - Uso A → B quindi A^+ = AB
  - Uso B → C quindi A^+ = ABC
- Verifica che il conseguente (C) sia contenuto in A+
  - A $\in$ ABC, quindi A→ C **è ridondante e aggiorno F'**

Proprietà

La sequenza dei due blocchi di operazioni è rigida, cioè non è possibile invertire i due blocchi

Bisogna sempre prima eliminare tutti gli attributi estranei, poi è possibile eliminare le dipendenze funzionali ridondanti

Controesempio

Considero F'={AB $\rightarrow$ C, C $\rightarrow$ A, B $\rightarrow$ A}

Proviamo ad invertire i due blocchi di codice

Cerchiamo prima le dipendenze funzionali ridondanti:

AB$\rightarrow$ C è ridondante?

Controesempio

Considero F'={AB → C, C → A, B → A}

AB→ C è ridondante?

La chiusura di AB rispetto ad F* = {C → A, B → A} è AB

C non è contenuto in AB quindi AB→ C non è ridondante

Controesempio

Considero $F' = \{AB \rightarrow C, C \rightarrow A, B \rightarrow A\}$

$C \rightarrow A$ è ridondante?

La chiusura di C rispetto ad $F^* = \{AB \rightarrow C, B \rightarrow A\}$ è C

A non è contenuto in C quindi $C \rightarrow A$ non è ridondante

Controesempio

Considero $F'=\{AB \rightarrow C, C \rightarrow A, B \rightarrow A\}$

$B \rightarrow A$ è ridondante?

La chiusura di B rispetto ad $F^* = \{AB \rightarrow C, C \rightarrow A\}$ è B

A non è contenuto in B quindi $B \rightarrow A$ non è ridondante

Controesempio

Considero F'={AB $\rightarrow$ C, C $\rightarrow$ A, B $\rightarrow$ A}

Ora cerco gli attributi estranei (ovviamente mi concentro sulle d.f. con antecedenti composti da almeno due attributi)

Considero solo AB $\rightarrow$ C

Controesempio

Considero F'={AB → C, C → A, B → A}

Considero solo AB → C

Mi chiedo se A è estraneo

- Calcolo la chiusura di B rispetto ad F'
  - con B → A ho B^+ = AB
  - con AB → C ho B^+ = ABC
- Il conseguente C è contenuto nella chiusura, quindi A è un attributo estraneo e

F' = {B → C, C → A, B → A}

Controesempio

Sono quindi rimasto con F' = {B → C, C → A, B → A}

Ma ora si verifica che B → A è ridondante per transitività

Quindi i due passi **non sono invertibili**!

Non unicità di F'

Consideriamo codice fiscale (CF), matricola (MATR) e credenziali SCU (SCU)

Ho tutte le dipendenze funzionali seguenti:

CF → MATR

CF → SCU

MATR → CF

MATR → SCU

SCU → CF

SCU → MATR

Non unicità di F'

E' chiaro che c'è ridondanza

CF → MATR

CF → SCU

MATR → CF

~~MATR → SCU~~

SCU → CF

~~SCU → MATR~~

Per esempio posso cancellare SCU → MATR e MATR → SCU perché posso ottenerle per transitività

Non unicità di F'

E' chiaro che c'è ridondanza

CF → MATR

CF → SCU

MATR → CF

~~MATR → SCU~~

SCU → CF

~~SCU → MATR~~

Si verifica che le dipendenze funzionali rimaste non sono ridondanti (**esercizio**)

Non unicità di F'

E' chiaro che c'è ridondanza

~~CF → MATR~~

CF → SCU

~~MATR → CF~~

MATR → SCU

SCU → CF

SCU → MATR

Potrei però cancellare anche CF → MATR e MATR → CF perché posso ottenerle per transitività

Non unicità di F'

E' chiaro che c'è ridondanza

~~CF → MATR~~

CF → SCU

~~MATR → CF~~

MATR → SCU

SCU → CF

SCU → MATR

Si verifica che le dipendenze funzionali rimaste non sono ridondanti (**esercizio**) e ho un altro insieme F'

Non unicità di F'

Posso generare un altro insieme F' di copertura minimale considerando un ciclo

CF → SCU

SCU → MATR

MATR → CF

E' una copertura minimale più semplice ma è equivalente agli altri casi (**esercizio**)

Altro esempio

Consideriamo la relazione ESAMI con le dipendenze funzionali
$F = \{MATR,CF,Co \rightarrow Vo,\ MATR \rightarrow CF,\ CF \rightarrow MATR\}$

Mostreremo la non unicità di F' in relazione agli attributi estranei (prima invece l'abbiamo mostrata in relazione alle dipendenze ridondanti)

Altro esempio

Consideriamo la relazione ESAMI con le dipendenze funzionali

F = {MATR,CF,Co → Vo, MATR → CF, CF → MATR}

Verifico solo MATR,CF,Co → Vo

Si vede immediatamente che MATR è estraneo se calcolo la chiusura di CF,Co utilizzando CF → MATR

Quindi ho F' = {CF,Co → Vo, MATR→ CF, CF → MATR}

Altro esempio

Consideriamo la relazione ESAMI con le dipendenze funzionali

F = {MATR,CF,Co → Vo, MATR → CF, CF → MATR}

Verifico solo MATR,CF,Co → Vo

Ma anche CF è estraneo (non c'è nessun ordine) se calcolo la chiusura di MATR,Co utilizzando MATR → CF

Quindi ho F' = {MATR,Co→Vo, MATR→CF, CF→MATR}

Complessità dell'algoritmo

Qual è la complessità dell'algoritmo?

Proprietà (complessità)

La complessità dell'algoritmo per il calcolo dell'insieme di copertura minimale è **polinomiale**, infatti

- F è finito
- X è finito
- L'algoritmo per il calcolo della chiusura è **polinomiale**

Sintesi in 3NF di uno schema

A partire da una relazione *$(R(A),F)$*

1. Calcoliamo la copertura minimale F' di F
   - $F' = \{X_1 \rightarrow A_1, X_2 \rightarrow A_2, \ldots, X_n \rightarrow A_n\}$
2. Costruisco un insieme di relazioni
   - $\{R_1(X_1A_1), R_2(X_2A_2), \ldots, R_n(X_n,A_n)\}$
3. Verifico che una o più di queste relazioni contenga una chiave K qualsiasi di $R(A)$
4. Se nessuna relazione contiene chiavi della relazione di partenza aggiungo una nuova relazione $R_{n+1}(K)$

Esempio

ESAMI(<u>MATR</u>,NS,DN,<u>Co</u>,Vo,DE,CP,NP)

F={MATR,Co→ Vo,DE,CP; MATR→NS,DN; CP→NP}

- Scrivo F in forma canonica:
  F = {MATR → NS; MATR→ DN; MATR,Co→ Vo; MATR,Co → DE; MATR,Co→CP; CP → NP}

In questo caso F è già minimale

Esempio

F = {MATR $\rightarrow$ NS; MATR$\rightarrow$ DN; MATR,Co$\rightarrow$ Vo; MATR,Co $\rightarrow$ DE; MATR,Co$\rightarrow$CP; CP $\rightarrow$ NP}

Scrivo la scomposizione:

- R_1(<u>MATR</u>,NS)
- R_2(<u>MATR</u>,DN)
- R_3(<u>MATR,Co</u>,Vo)
- R_4(<u>MATR,Co</u>,DE)
- R_5(<u>MATR,Co</u>,CP)
- R_6(<u>CP</u>,NP)

Esempio

Verifico che la chiave K={MATR,Co} sia contenuta in almeno una delle relazioni

- R_1(<u>MATR</u>,NS)
- R_2(<u>MATR</u>,DN)
- R_3(<u>**MATR,Co**</u>,Vo)
- R_4(<u>**MATR,Co**</u>,DE)
- R_5(<u>**MATR,Co**</u>,CP)
- R_6(<u>CP</u>,NP)

La chiave è contenuta in tre relazioni quindi lo schema è in 3NF

Altro esempio

CC(<u>Titolare,NConto</u>,NAgenzia,CittaAgenzia,Saldo)

Con F = {NConto $\rightarrow$ NAgenzia, NConto $\rightarrow$ CittaAgenzia, NConto $\rightarrow$ Saldo} già in forma canonica (e minimale)

Costruisco la sintesi:

- R_1(<u>NConto</u>,NAgenzie)
- R_2(<u>NConto</u>,CittaAgenzia)
- R_3(<u>NConto</u>,Saldo)

Altro esempio

CC($\underline{\text{Titolare,NConto}}$,NAgenzia,CittaAgenzia,Saldo)

- R_1($\underline{\text{NConto}}$,NAgenzie)
- R_2($\underline{\text{NConto}}$,CittaAgenzia)
- R_3($\underline{\text{NConto}}$,Saldo)

La chiave K={Titolare,NConto} non è contenuta in nessuna relazione quindi aggiungo la relazione

- R_4($\underline{\text{Titolare,NConto}}$)

Proprietà dell'algoritmo

Consideriamo una qualsiasi $R_i(X_iA_i)$: la relazione è stata generata da $X_i \rightarrow A_i$, quindi X_i è superchiave di R_i, ma si dimostra che X_i è **chiave**, non solo superchiave, di R_i

Infatti, supponiamo che X_i sia solo superchiave e quindi esista un sottinsieme $W \subset X_i$ tale che $F' \vdash W \rightarrow A_i$

Allora, $X_i \rightarrow W$ per riflessività, quindi per transitività, $X_i \rightarrow A_i$, cioè $X_i \rightarrow A_i$ è **ridondante**!

Di conseguenza $W \rightarrow A_i$ non può esistere, e X_i è **chiave**

Osservazione

Analizziamo, dal punto di vista combinatorio, tutte le dipendenze funzionali di $R_i(X_iA_i)$ che si ereditano dall'insieme F'

Posso avere dipendenze del tipo $A_i \rightarrow W$ con $W \subseteq X_i$?

Osservazione

Analizziamo, dal punto di vista combinatorio, tutte le dipendenze funzionali di $R_i(X_i A_i)$ che si ereditano dall'insieme F'

Posso avere dipendenze del tipo $A_i \rightarrow W$ con $W \subseteq X_i$?

Sì, perché X_i è chiave, quindi W sono attributi primi

Osservazione

Analizziamo, dal punto di vista combinatorio, tutte le dipendenze funzionali di $R_i(X_iA_i)$ che si ereditano dall'insieme F'

Posso avere dipendenze del tipo $WA_i \rightarrow Y$ con $W,Y \subseteq X_i$?

Osservazione

Analizziamo, dal punto di vista combinatorio, tutte le dipendenze funzionali di $R_i(X_iA_i)$ che si ereditano dall'insieme F'

Posso avere dipendenze del tipo $WA_i \rightarrow Y$ con $W,Y \subseteq X_i$?

Sì, perché Y sono attributi primi

Osservazione

Analizziamo, dal punto di vista combinatorio, tutte le dipendenze funzionali di $R_i(X_i A_i)$ che si ereditano dall'insieme F'

Posso avere dipendenze del tipo $W \rightarrow A_i$ con $W \subset X_i$?

Osservazione

Analizziamo, dal punto di vista combinatorio, tutte le dipendenze funzionali di $R_i(X_iA_i)$ che si ereditano dall'insieme F'

Posso avere dipendenze del tipo $W \rightarrow A_i$ con $W \subset X_i$?

No, perché $X_i \rightarrow A_i$ sarebbe ridondante

Osservazione

Analizziamo, dal punto di vista combinatorio, tutte le dipendenze funzionali di $R_i(X_i A_i)$ che si ereditano dall'insieme F'

Posso avere dipendenze del tipo $W \rightarrow Y$ con $W, Y \subset X_i$?

Osservazione

Analizziamo, dal punto di vista combinatorio, tutte le dipendenze funzionali di $R_i(X_i A_i)$ che si ereditano dall'insieme F'

Posso avere dipendenze del tipo $W \rightarrow Y$ con $W, Y \subset X_i$?

No, perché Y sarebbe attributo estraneo e la chiusura di $(X_i - Y)$ conterrebbe A_i

Proprietà dell'algoritmo

Per quanto detto

la sintesi $\{R_1(X_1A_1), R_2(X_2A_2), \ldots, R_n(X_n,A_n), [R_{n+1}(K)]\}$ è uno schema di relazioni in 3NF

Proprietà (mantenimento della località delle dipendenze)

Ogni dipendenza funzionale $X_i \rightarrow A_i$ della copertura minimale F' si trova nella restrizione della corrispondente relazione $R_i(X_i A_i)$ per costruzione, quindi la decomposizione **mantiene la località delle dipendenze**

Esempio

CC(Titolare, NConto, NAgenzia, CittaAgenzia, Saldo, Direttore, Qualifica, Stipendio)

Titolare	NConto	NAgenzia	CittaAgenzia	Saldo	Direttore	Qualifica	Stipendio
Rossi	333	1	TO	100	Bianchi	F3	50000
Verdi	444	2	TO	50	Bruno	F2	60000
Franco	333	1	TO	100	Bianchi	F3	50000
Pippo	444	2	TO	50	Bruno	F2	60000

Esempio

CC(<u>Titolare, NConto</u>, NAgenzia, CittaAgenzia, Saldo, Direttore, Qualifica, Stipendio)

F' = {

1. NConto → NAgenzia
2. NConto → CittaAgenzia
3. NConto → Saldo
4. NAgenzia, CittaAgenzia → Direttore
5. Direttore → Qualifica
6. Qualifica → Stipendio
7. Direttore → CittaAgenzia

}

F' è già una copertura minimale

Esempio

1. Calcoliamo una copertura minimale (F' già minimale)
2. Costruisco una relazione per ogni dipendenza funzionale
   - R1(<u>NConto</u>,NAgenzia)
   - R2(<u>NConto</u>,CittaAgenzia)
   - R3(<u>NConto</u>,Saldo)
   - R4(<u>NAgenzia,CittaAgenzia</u>,Direttore)
   - R5(<u>Direttore</u>,Qualifica)
   - R6(<u>Qualifica</u>,Stipendio)
   - R7(<u>Direttore</u>,CittaAgenzia)

Esempio

3. Verifico se qualche relazione contiene almeno una chiave della relazione di partenza F = {Titolare,NConto}
   - R1(<u>NConto</u>,NAgenzia)
   - R2(<u>NConto</u>,CittaAgenzia)
   - R3(<u>NConto</u>,Saldo)
   - R4(<u>NAgenzia,CittaAgenzia</u>,Direttore)
   - R5(<u>Direttore</u>,Qualifica)
   - R6(<u>Qualifica</u>,Stipendio)
   - R7(<u>Direttore</u>,CittaAgenzia)
4. Non avendola trovata, creo una relazione su K:
   - R8(<u>Titolare,NConto</u>)

Esempio

Materializziamo le $R_i(X_iA_i)=\Pi_{X_iA_i}(r(A))$ e $R_{n+1}(K)= \Pi_K(r(A))$

R1

NConto	NAgenzia
333	1
444	2

R2

NConto	CittaAgenzia
333	TO
444	TO

R3

NConto	Saldo
333	100
444	50

R4

NAgenzia	CittaAgenzia	Direttore
1	TO	Bianchi
2	TO	Bruno

R5

Direttore	Qualifica
Bianchi	F3
Bruno	F2

R6

Qualifica	Stipendio
F3	50000
F2	60000

R7

Direttore	CittaAgenzia
Bianchi	TO
Bruno	TO

R8

Titolare	NConto
Rossi	333
Verdi	444
Franco	333
Pippo	444

Esempio

- Proprietà verificata: la sintesi mantiene tutte le dipendenze, infatti ognuna delle dipendenze di F' la troviamo all'interno della relazione corrispondente
- Proprietà verificata: tutte le relazioni sono in 3NF, infatti le dipendenze sono tutte BCNF
- Proprietà verificata: la sintesi si ottiene in tempo polinomiale

- **Proprietà da verificare**: la sintesi garantisce la decomposizione con join senza perdita

Proprietà (decomposizione con join senza perdita)

La sintesi in 3NF garantisce la decomposizione con join senza perdita, ovvero

$r(A) = r_1(X_1A_1) \bowtie r_2(X_2A_2) \bowtie \dots \bowtie r_n(X_nA_n) \bowtie r_{n+1}(K)$

E' un'uguaglianza insiemistica, quindi dobbiamo dimostrare la doppia inclusione

1. $r(A) \subseteq r_1(X_1A_1) \bowtie r_2(X_2A_2) \bowtie \dots \bowtie r_n(X_nA_n) \bowtie r_{n+1}(K)$
2. $r_1(X_1A_1) \bowtie r_2(X_2A_2) \bowtie \dots \bowtie r_n(X_nA_n) \bowtie r_{n+1}(K) \subseteq r(A)$

Ma la prima è dimostrata per il **Lemma**

Proprietà (decomposizione con join senza perdita)

La sintesi in 3NF garantisce la decomposizione con join senza perdita, ovvero

$r(A) = r_1(X_1A_1) \bowtie r_2(X_2A_2) \bowtie \ldots \bowtie r_n(X_nA_n) \bowtie r_{n+1}(K)$

Devo dimostrare

$$r_1(X_1A_1) \bowtie r_2(X_2A_2) \bowtie \ldots \bowtie r_n(X_nA_n) \bowtie r_{n+1}(K) \subseteq r(A)$$

Dimostrazione

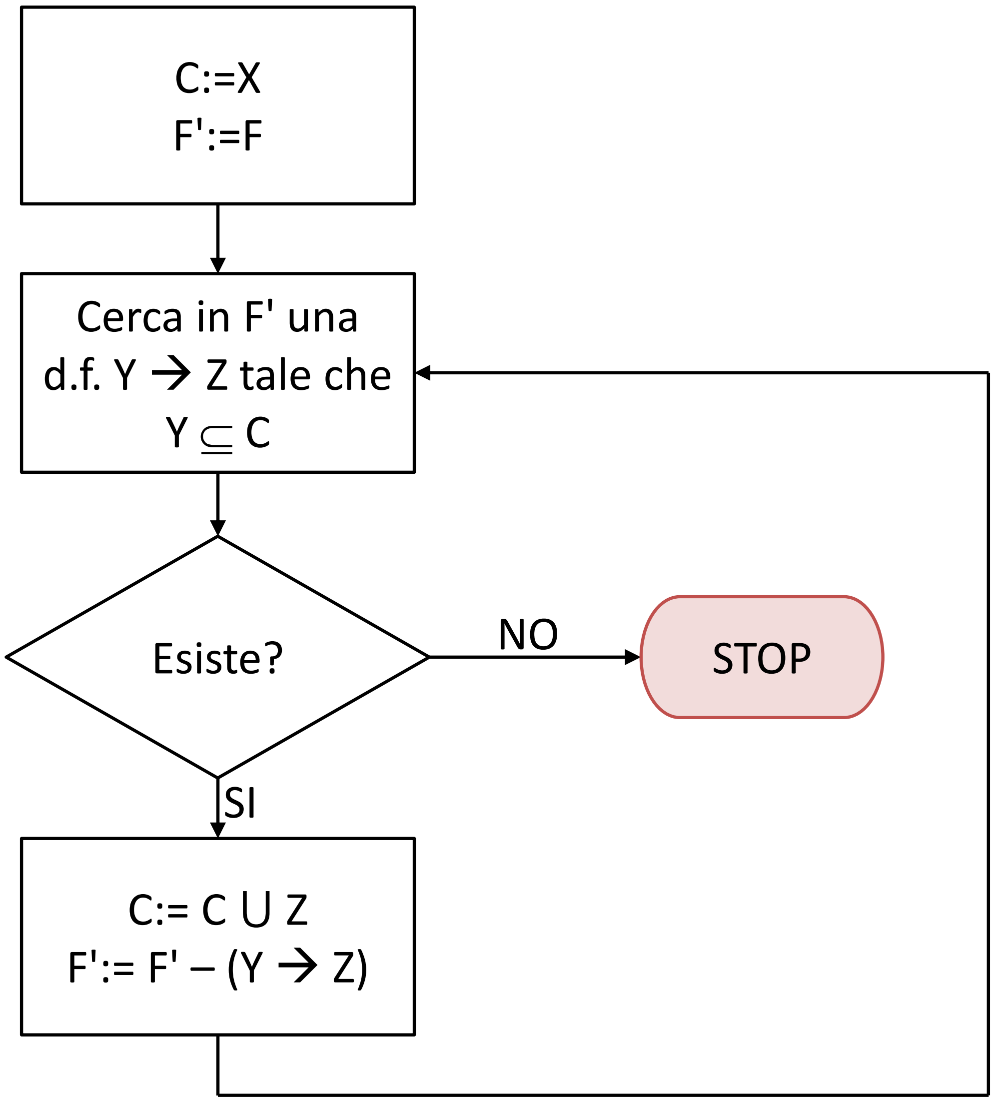
Dobbiamo prendere in esame una tupla generica del natural join dimostrare che è contenuta in *r(A)*

Dimostrazione

Dobbiamo prendere in esame una tupla generica del natural join dimostrare che è contenuta in *r(A)*

Prima però facciamo un richiamo all'algoritmo di chiusura

Algoritmo per il calcolo di X^+



Algoritmo di chiusura (ripasso)

- Consideriamo una chiave *K*
- L'algoritmo di chiusura prende l'insieme *K* e lo inserisce nella chiusura
- Poi cerca, nell'insieme *F'* (copertura minimale), la dipendenza funzionale che ha l'antecedente contenuto in *K*
- *K* è chiave quindi, necessariamente, $K^+_{F'} = A$ (tutti gli attributi della relazione), quindi, prima o poi, l'algoritmo utilizzerà tutte le dipendenze funzionali

Algoritmo di chiusura (ripasso)

- Senza perdita di generalità, presumo che la prima dipendenza funzionale trovata sia proprio $X_1 \rightarrow A_1$
- *C* conterrà quindi $K \cup A_1$
- Successivamente l'algoritmo considererà $X_2 \rightarrow A_2$ e così via
- Ipotizzo nell'insieme di copertura minimale l'ordine indotto dall'algoritmo di chiusura di *K* rispetto alle dipendenze funzionali minimali
- La chiusura prima o poi produrrà l'intero insieme degli attributi di *A*

Algoritmo di chiusura (ripasso)

- La chiusura prima o poi produrrà l'intero insieme degli attributi di *A*

- Quindi troverà sempre uno stadio per cui l'antecedente di una dipendenza funzionale minimale è contenuto nel passo di chiusura

- **Tutte** le dipendenze funzionali vengono sempre utilizzate

Dimostrazione

Scrivo il natural join per passi in questo modo:

- $r_{n+1}(K) \bowtie r_1(X_1A_1)$
  perché con l'algoritmo di chiusura ho fatto l'ipotesi che X_1 sia l'insieme di attributi contenuto in K
- $(r_{n+1}(K) \bowtie r_1(X_1A_1)) \bowtie r_2(X_2A_2)$
  perché con l'algoritmo di chiusura ho fatto l'ipotesi che X_2 sia l'insieme di attributi contenuto in $K'=K \cup A_1$
- $(\ldots((r_{n+1}(K) \bowtie r_1(X_1A_1)) \bowtie r_2(X_2A_2)) \bowtie \ldots) \bowtie r_n(X_nA_n)$
  sempre per l'ordinamento indotto dall'algoritmo di chiusura

Dimostrazione

- Ricordiamo (definizione decomposizione con join senza perdita) che le $R_i(A_iX_i)$ sono proiezioni di R

- Una tupla t qualunque prodotta dal natural join avrà determinati valori in corrispondenza di K, gli stessi valori che aveva anche in $r(A)$, in corrispondenza degli attributi K più altri che non conosco

	K	**A – K**
t	///	???

Dimostrazione

Mostreremo ora che anche la restante parte della tupla ha valori noti

Abbiamo detto che X_1 è contenuto in K, quindi se prendiamo la tupla t_1 *di* r_1 che contribuisce al natural join la prima volta, di questa tupla conosco un pezzettino (X_1) tutto contenuto in K

	X1	A1
t1	***	+++

Dimostrazione

Abbiamo detto che X_1 è contenuto in K, quindi se prendiamo la tupla t_1 *di* r_1 che contribuisce al natural join la prima volta, di questa tupla conosco un pezzettino (X_1) tutto contenuto in K

Quello che la tupla aveva in r fuori da X_1 e A_1 non è mi è noto

	K – X1	**X1**	**A1**	**A – K – A1**
t1	???	***	+++	???

Dimostrazione

Nella mia ipotesi costruttiva, X_1 ha gli stessi valori che ha un pezzo di *K* della tupla *t*

All'interno di *r* ci dovevano essere *t* e t_1 che condividevano esattamente gli stessi valori in corrispondenza di X_1, ma $X_1 \rightarrow A_1$ è un vincolo valido in *r(A)*, pertanto, se ci sono due tuple in *r* che condividono gli stessi valori di X_1, necessariamente devono condividere gli stessi valori di A_1

	K – X1	X1	A1	A – K – A1
t	///	***	+++	???
t1	???	***	+++	???

Dimostrazione

Pertanto il natural join $r_{n+1}(K) \bowtie r_1(X_1 A_1)$ aggancia il pezzo $X_1 A_1$ al pezzo K producendo un pezzo di una tupla sicuramente presente in r

	K – X1	X1	A1	A – K – A1
t	///	***	+++	???

Dimostrazione

Prendiamo X_2 e la tupla t_2 *di* r_2 che contribuisce al natural join la seconda volta

Di questa tupla conosco un pezzettino (X_2) che per costruzione è contenuto in $K \cup A_1$

	X2	A2
t2	$$$	%%%

Ma il vincolo $X_2 \rightarrow A_2$ ci dice che t[A_2]=t_2[A_2]

	K – X1 – X2	X1	X2	A1	A2	A – K – A1 – A2
t	///	***	$$$	+++	%%%	???
t2	???	???	$$$	???	%%%	???

Dimostrazione

Pertanto il natual join $(r_{n+1}(K) \bowtie r_1(X_1A_1)) \bowtie r_2(X_2A_2)$ aggancia il pezzo X_2A_2 al pezzo KA_1 producendo un altro pezzo di una tupla sicuramente presente in r

	K – X1 – X2	X1	X2	A1	A2	A – K – A1 – A2
t	///	***	$$$	+++	%%%	???

Dimostrazione

Alla fine, dopo l'ultimo join con $r_n(X_nA_n)$, ottengo:

	K – X1 – X2 … – Xn	**X1**	**X2**	**…**	**Xn**	**A1**	**A2**	**…**	**An**	**A – K – A1 – A2 – … – An**
t	///	***	$$$	…	&&&	+++	%%%	…	###	???

Ma $A - K - A_1 - A_2 - \ldots - A_n = \emptyset$, quindi la tupla è interamente ricostruita

	K – X1 – X2 … – Xn	**X1**	**X2**	**…**	**Xn**	**A1**	**A2**	**…**	**An**
t	///	***	$$$	…	&&&	+++	%%%	…	###

Dimostrazione

La tupla t è ricostruita con i passi di natural join, ma sapendo che la tupla t arriva da r per proiezione, abbiamo la certezza che la tupla ricostruita dal natural join era una tupla presente nella relazione di partenza

Osservazione

- L'ordinamento delle relazioni non è indispensabile alla dimostrazione

- Tuttavia considero l'ordinamento indotto dall'algoritmo per il calcolo della chiusura della chiave K, al fine di semplificare la dimostrazione

Proprietà

- Proprietà verificata: la sintesi mantiene tutte le dipendenze

- Proprietà verificata: tutte le relazioni sono in 3NF

- Proprietà verificata: la sintesi si ottiene in tempo polinomiale

- **Proprietà verificata**: la sintesi garantisce la decomposizione con join senza perdita

Esempio

Riprendiamo la sintesi in 3NF del nostro esempio

- R1(<u>NConto</u>,NAgenzia)
- R2(<u>NConto</u>,CittaAgenzia)
- R3(<u>NConto</u>,Saldo)
- R4(<u>NAgenzia,CittaAgenzia</u>,Direttore)
- R5(<u>Direttore</u>,Qualifica)
- R6(<u>Qualifica</u>,Stipendio)
- R7(<u>Direttore</u>,CittaAgenzia)
- R8(<u>Titolare,NConto</u>)

Possiamo ridurre questi schemi?

Prima semplificazione

Quando alcuni schemi condividono la stessa chiave X, si possono sostituire con un'unica relazione

Dati $R_1(\underline{X}A_1)$, $R_2(\underline{X}A_2)$,…, $R_h(\underline{X}A_h)$, posso sostituire $R_1, R_2, \ldots, R_h$ con un'unica relazione $R'_h(\underline{X}A_1A_2 \ldots A_h)$

La relazione R'_h, mantiene la 3FN?

Prima semplificazione

La relazione R'_h, mantiene la 3FN?

Consideriamo un esempio:

$R_1(\underline{X}A)$, $R_2(\underline{X}B)$, $R_3(\underline{X}C)$, $R_4(\underline{X}D)$ e la semplificazione $R'(\underline{X}ABCD)$

Può risultare, in R', una dipendenza $B \rightarrow C$ (non di tipo 1, 2 o 3)?

Prima semplificazione

Può risultare, in R', una dipendenza $B \rightarrow C$ (non di tipo 1, 2 o 3)?

Ipotizziamone l'esistenza.

Esiste una $X \rightarrow B$ per ipotesi, ma allora, per transitività, $X \rightarrow C$ è ridondante e non può comparire nell'insieme di copertura minimale (**contraddizione**)

Esempio

Nell'esempio otteniamo:

- R3'(NConto,NAgenzia, CittaAgenzia,Saldo)
- R7'(NAgenzia,CittaAgenzia,Direttore)
- R5(Direttore,Qualifica,CittaAgenzia)
- R6(Qualifica,Stipendio)
- R8(Titolare,NConto)

Si può semplificare ulteriormente?

Seconda semplificazione

Date due relazioni $R_i(X_iA_i)$ e $R_j(X_jA_j)$ dove per ipotesi $X_jA_j \subset X_iA_i$, allora la relazione R_j si può cancellare (e in R_i ci sarà la dipendenza funzionale $X_j \rightarrow A_j$)

La dipendenza $X_j \rightarrow A_j$ è necessariamente di tipo 3

Esempio

Lo schema di R7 è contenuto nello schema di R4

- R1(NConto,NAgenzia)
- R2(NConto,CittaAgenzia)
- R3(NConto,Saldo)
- **R4(NAgenzia,CittaAgenzia,Direttore)**
- R5(Direttore,Qualifica)
- R6(Qualifica,Stipendio)
- **R7(Direttore,CittaAgenzia)**
- R8(Titolare,NConto)

Esempio

Lo schema di R7 è contenuto nello schema di R4

- R1(<u>NConto</u>,NAgenzia)
- R2(<u>NConto</u>,CittaAgenzia)
- R3(<u>NConto</u>,Saldo)
- **R4(<u>NAgenzia,CittaAgenzia</u>,Direttore)**
- R5(<u>Direttore</u>,Qualifica)
- R6(<u>Qualifica</u>,Stipendio)
- R8(<u>Titolare,NConto</u>)

con F4 = { NAgenzia,CittaAgenzia → Direttore,
Direttore → CittaAgenzia }

Conclusione

Mettendo insieme le due semplificazione possiamo indicare il processo di sintesi in modo semplificato, lavorando sin da subito sul compattamento delle dipendenze funzionali

Sintesi in 3NF di uno schema

A partire da una relazione *(R(A),F)*

1. Calcoliamo la copertura minimale *F'* di *F*
2. Accorpa in *F'* tutte le dipendenze con lo stesso antecedente (**prima semplificazione**)
3. Costruisco un insieme di relazioni di sintesi
4. Verifico che una o più di queste relazioni contenga una chiave *K* qualsiasi di *R(A)*
5. Se nessuna relazione contiene chiavi della relazione di partenza aggiungo una nuova relazione $R_{n+1}(K)$
6. Fondere le relazioni incluse (**seconda semplificazione**)

Esempio

CC(<u>Titolare, NConto</u>, NAgenzia, CittaAgenzia, Saldo, Direttore, Qualifica, Stipendio)

F' = {

1. NConto → NAgenzia
2. NConto → CittaAgenzia
3. NConto → Saldo
4. NAgenzia, CittaAgenzia → Direttore
5. Direttore → Qualifica
6. Qualifica → Stipendio
7. Direttore → CittaAgenzia

}

Esempio

F' = {

1. NConto → NAgenzia, CittaAgenzia, Saldo
2. NAgenzia, CittaAgenzia → Direttore
3. Direttore → Qualifica,CittaAgenzia
4. Qualifica → Stipendio

}

Calcoliamo la sintesi:

- CC(<u>NConto</u>,NAgenzia, CittaAgenzia,Saldo)
- AGENZIE(<u>NAgenzia,CittaAgenzia</u>,Direttore) (con Direttore → CittaAgenzia)
- DIRETTORI(<u>Direttore</u>,Qualifica,CittaAgenzia)
- STIPENDI(<u>Qualifica</u>,Stipendio)
- TITOLARI(<u>Titolare,NConto</u>)

Conclusione

Alcune delle relazioni generate dalla sintesi in 3NF sono in BCNF: sono le relazioni in cui non ci sono dipendenze di tipo 3 (ad esempio, relazione CC, DIRETTORI, mentre AGENZIE non è in BCNF)

Si consiglia quindi di progettare sempre in 3NF per via di tutti i vantaggi:

- mantenimento delle dipendenze
- decomposizione con join senza perdita
- complessità polinomiale

Conclusione

Alla fine, si può analizzare quanto ottenuto: alcune relazioni saranno in BCNF e le anomalie saranno annullate, dove invece appare la dipendenza funzionale di tipo 3 rimane invece l'anomalia

Ad esempio in AGENZIE, ci possono essere anomalie introdotte dalla d.f. Direttore → CittaAgenzia

Esempio

Considero ora la relazione ESAMI e la copertura minimale di F

F' = {

CF → MATR

MATR → NS

MATR → IS

MATR → DN

MATR → CF

IS → CAP

MATR, Co → Vo

MATR, Co → DE

MATR, Co → CP

CP → NP

CP → Q

Q → TU

CP → Co

}

Esempio

Accorpiamo le dipendenze funzionali con stesso antecedente

F' = {
CF → MATR
MATR → NS
MATR → IS
MATR → DN
MATR → CF
IS → CAP
MATR, Co → Vo
MATR, Co → DE
MATR, Co → CP
CP → NP
CP → Q
Q → TU
CP → Co
}

F' = {
CF→MATR
MATR → NS,IS,DN,CF
IS → CAP
MATR, Co → Vo,DE,CP
CP → NP,Q
Q → TU
CP → Co
}

Esempio

Costruiamo la sintesi in 3NF

F' = {
CF → MATR
MATR → NS,IS,DN,CF
IS → CAP
MATR, Co → Vo,DE,CP
CP → NP,Q
Q → TU
CP → Co
}

STUDENTI(MATR,NS,IS,DN,CF) con CF → MATR
INDIRIZZI(IS,CAP)
ESAMI(MATR,Co,Vo,DE,CP) con CP → Co
DOCENTI(CP,NP,Q)
QUALIFICHE(Q,TU)

Qui non ho bisogno di aggiungere la relazione con la chiave, in quanto ESAMI ingloba già la chiave della relazione di partenza

Osservazione

- La dimostrazione della proprietà di decomposizione con join senza perdita funziona anche se non c'è la relazione $R_{n+1}(K)$

- Basta partire comunque dalla relazione con la chiave (ESAMI)

- Quando calcolo la chiusura ho sicuramente tutti gli attributi dello schema di partenza

ER e normalizzazione

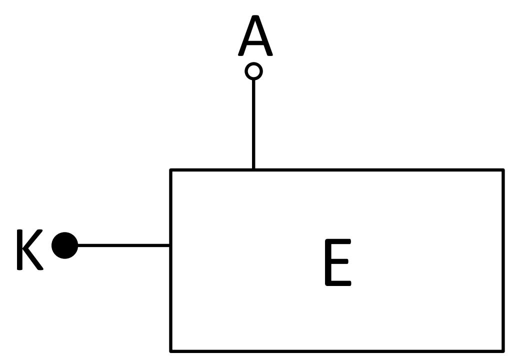
Cercheremo di indagare su cosa c'è di "normalizzazione" nella progettazione logica dell'Entity-Relationship

Cercheremo quindi di capire dove sono le dipendenze funzionali all'interno dello schema concettuale

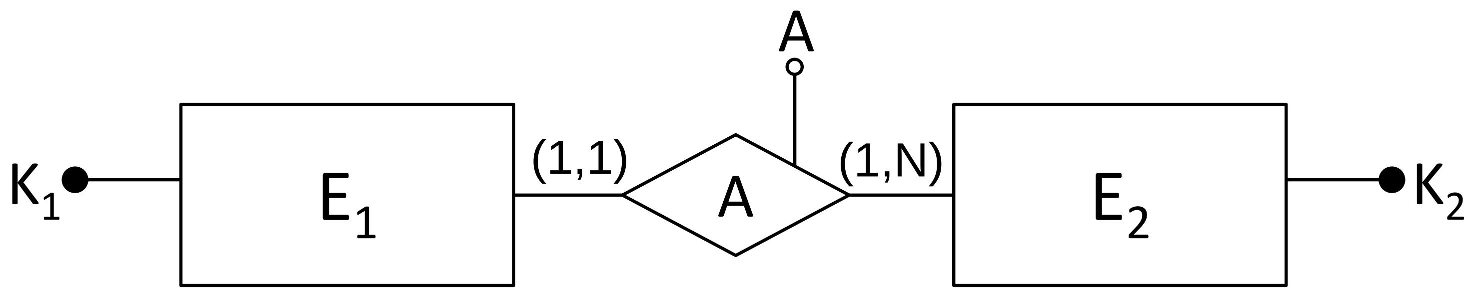
Vedremo ora come inserire delle dipendenze funzionali implicite negli schemi ER

Primo caso: entità

- Nella progettazione logica, sappiamo come tradurre in relazione l'entità E con identificatore K e attributi A
- L'entità esprime la dipendenza funzionale $K \rightarrow A$
- La traduzione diventa $R_E(\underline{K},A)$



Secondo caso: associazione 1:N

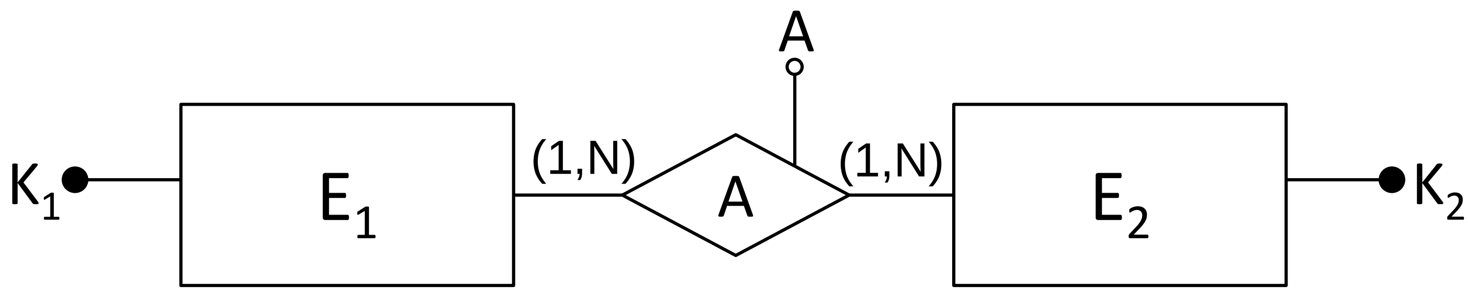


Sappiamo che c'è la traduzione $R_{E1}(\underline{K_1},\ldots)$, $R_{E2}(\underline{K_2},\ldots)$

Un'istanza di entità E_1 vede un'unica istanza di entità E_2, quindi ho $K_1 \rightarrow K_2,A$

La traduzione in relazionale diventa $R_A(\underline{K_1},K_2,A)$, che può essere incorporata nella R_{E1}

Terzo caso: associazione N:M

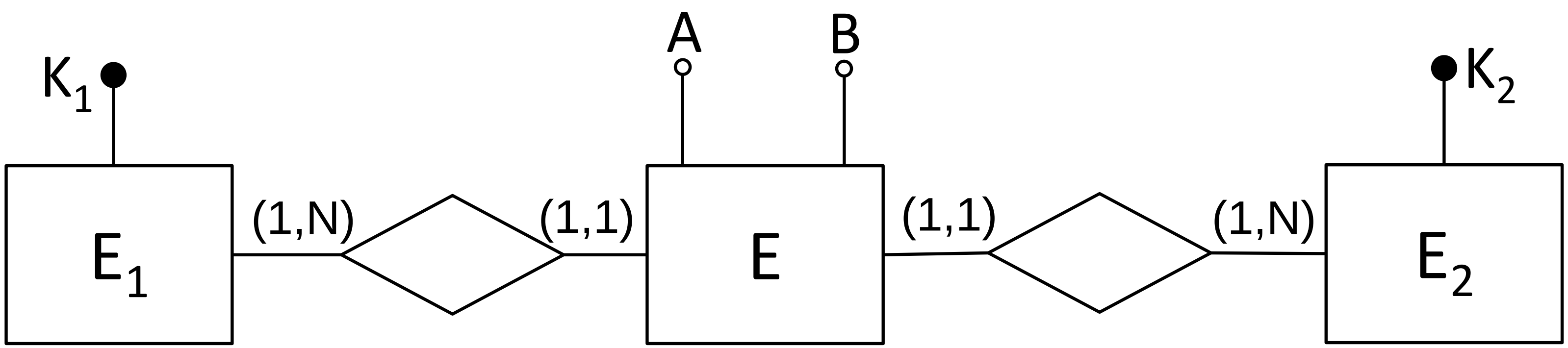


Sappiamo che c'è la traduzione $R_{E1}(\underline{K}_1,\ldots)$, $R_{E2}(\underline{K}_2,\ldots)$

Se esiste un'associazione tra un'istanza di E_1 e un'istanza di E_2, allora la coppia è abbinata ad un preciso valore dell'attributo A e ho $K_1K_2 \rightarrow A$

La traduzione in relazionale diventa $R_A(\underline{K}_1,\underline{K}_2,A)$

Quarto caso: entità debole

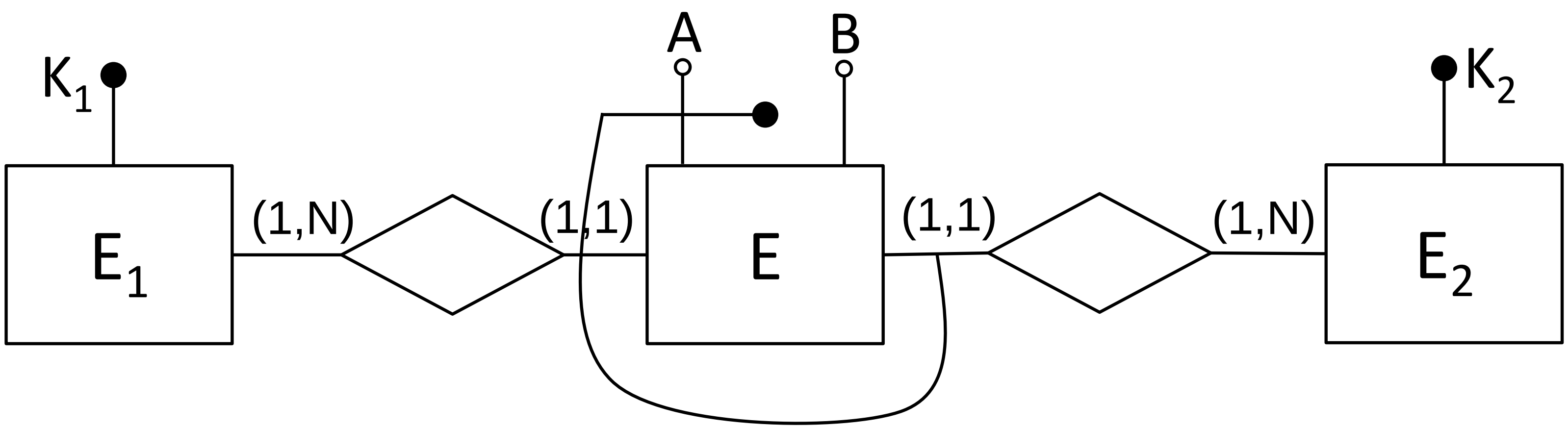


Sappiamo che la traduzione delle due entità identificanti è $R_{E1}(\underline{K_1},\ldots)$, $R_{E2}(\underline{K_2},\ldots)$

Presa un'istanza di E1 abbinata ad un'istanza di E2 e ad un determinato valore di A, abbiamo $K_1,K_2,A \rightarrow B$

La traduzione in relazionale diventa $R_E(\underline{K_1},\underline{K_2},\underline{A},B)$

Quarto caso: entità debole

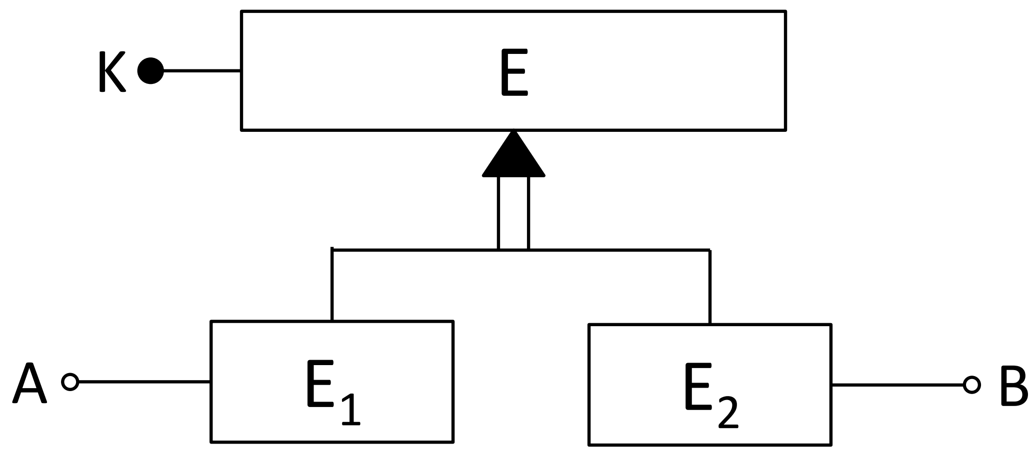


Sappiamo che la traduzione delle due entità identificanti è $R_{E1}(\underline{K}_1,\ldots)$, $R_{E2}(\underline{K}_2,\ldots)$

Presa un'istanza di E1 abbinata ad un'istanza di E2 e ad un determinato valore di A, abbiamo $K_1,K_2,A \rightarrow B$

La traduzione in relazionale diventa $R_E(\underline{K}_1,\underline{K}_2,\underline{A},B)$

Quinto caso: gerarchia



Uno degli schemi di progettazione logica per la traduzione della gerarchia consiste nel creare una relazione per ogni entità

Nelle gerarchie le sotto-entità ereditano tutte le proprietà dell'entità principale, quindi K identifica anche E_1 ed E_2 e valgono le d.f. $K \rightarrow A$ e $K \rightarrow B$

L'entità E è tradotta come $R_E(\underline{K},\ldots)$, le altre $R_{E1}(\underline{K},A,\ldots)$ e $R_{E2}(\underline{K},B,\ldots)$

Osservazione

La dipendenza funzionale nell'ER è determinata dall'identificazione

Se abbiamo fornito la casistica esaustiva delle dipendenze funzionali esprimibili nell'ER, che **forma normale** ha lo schema relazionale prodotto dalla progettazione logica?

Osservazione

La dipendenza funzionale nell'ER è determinata dall'identificazione

Se abbiamo fornito la casistica esaustiva delle dipendenze funzionali esprimibili nell'ER, che **forma normale** ha lo schema relazionale prodotto dalla progettazione logica?

E' in BCNF e mantiene la dipendenza espressa nello schema concettuale (sotto forma di chiave)

Proprietà dell'ER

La progettazione concettuale ER dal punto di vista delle dipendenze funzionali esprimibili nello schema concettuale conduce a schemi BCNF che mantengono le dipendenze, tramite la progettazione logica

Ricordiamo però che nella BCNF possiamo perdere delle dipendenze, come mai?

Nell'ER non riusciamo ad esprimere tutte le dipendenze funzionali possibili

Esempio

Riprendiamo la relazione ESAMI

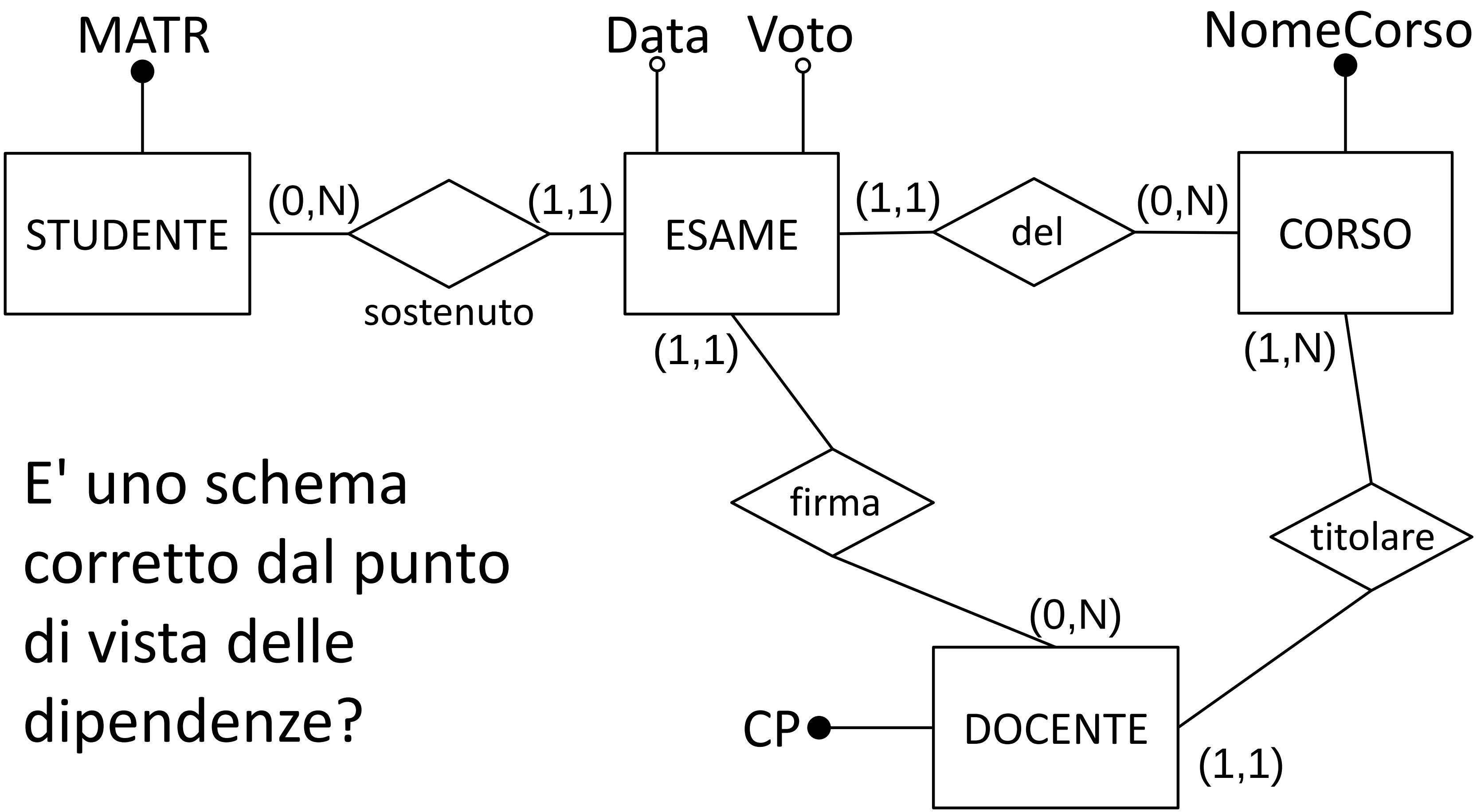
ESAMI(MATR,Co,DE,Vo,CP)

Con le dipendenze funzionali MATR,Co $\rightarrow$ DE,Vo,CP e CP $\rightarrow$ Co

La d.f. CP$\rightarrow$ Co non è BCNF (la relazione è in 3NF)

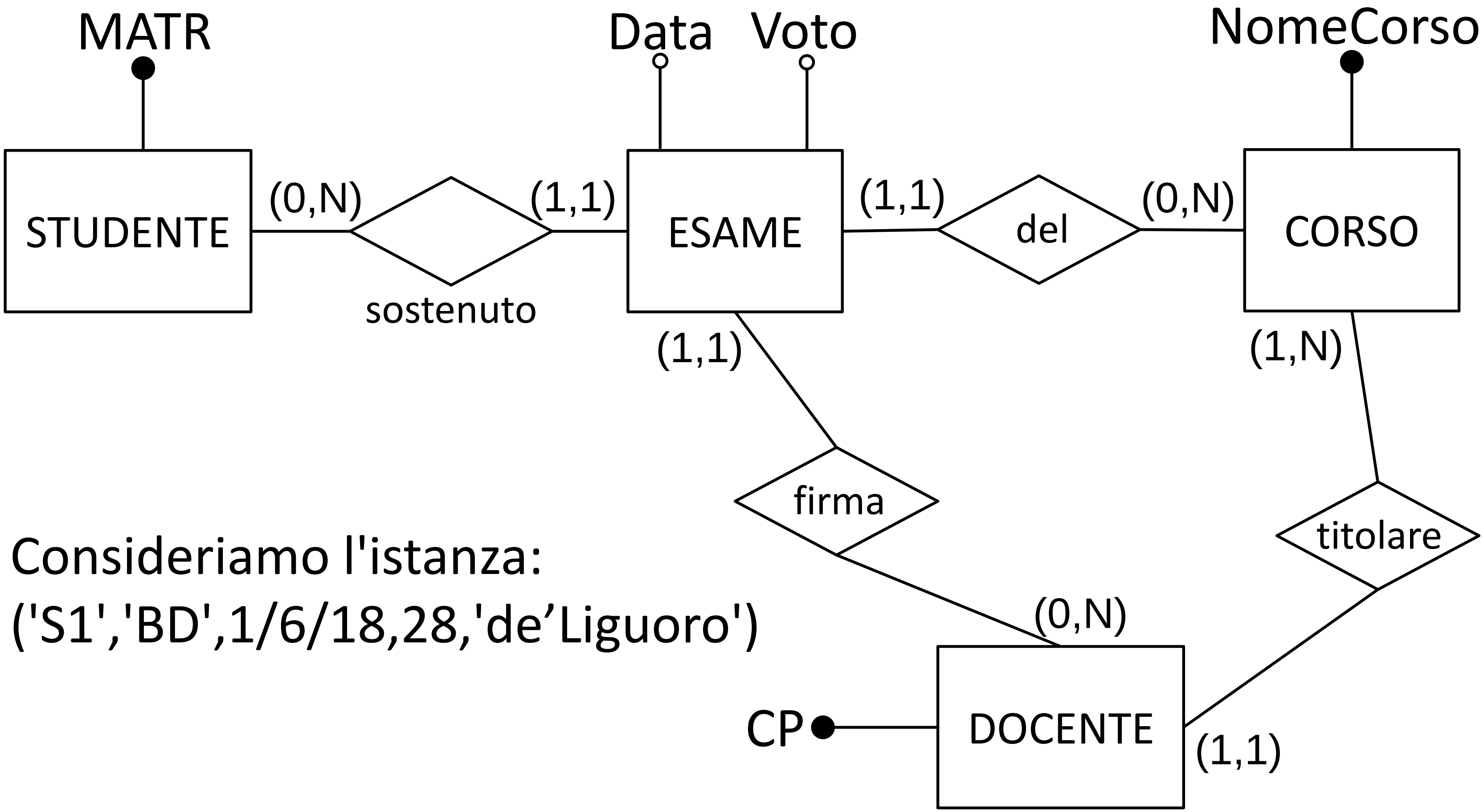
Proviamo a inventare uno schema concettuale che esprima le dipendenze funzionali

Esempio



E' uno schema corretto dal punto di vista delle dipendenze?

Esempio



Consideriamo l'istanza:
('S1','BD',1/6/18,28,'de’Liguoro')

Esempio

Le dipendenze, prese insieme, ci dicono che, dato un esame (MATR e Co) a firmare quell'esame dev'essere il docente titolare di quel corso (CP → Co)

L'istanza ('S1','BD',1/6/18,28,'de'Liguoro') è corretta dal punto di vista dell'ER, non delle dipendenze funzionali, infatti 'de'Liguoro' → 'Algoritmi'

L'ER non impone che il firmatario sia titolare del corso

Il ciclo ESAME → DOCENTE → CORSO non si chiude

Osservazione

Come si può intervenire?

Il progettista affronta la realtà smussandola con l'ER dal punto di vista delle dipendenze funzionali

Deve quindi chiedersi se non ci sono dipendenze (vincoli) non espressi dallo schema concettuale

Se ci sono, deve esprimerle come regole di business

Conclusione

L'Entity-Relationship, dal punto di vista delle dipendenze funzionali esprimibili con le identificazioni, produce schemi BCNF, ma gli schemi non sono in grado di esprimere intrecci di vincoli

Si richiede sempre un'attenta analisi successiva dello schema concettuale rispetto alla realtà rappresentata