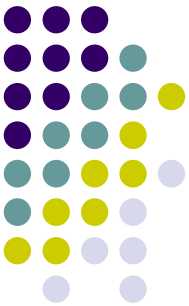
# FORME NORMALI E NORMALIZZAZIONE

Testo di Riferimento
*Elmasri, Navathe, "Sistemi di basi di dati", Pearson 2007*



## Progettazione Basi di Dati

- Tipicamente top-down: da concetti generali (schemi scheletro) per analisi successive fino al livello di dettaglio desiderato
- Come misurare la "bontà" di uno schema di base di dati?
- Una misura è legata alla presenza di ridondanze nei valori memorizzati nelle tuple

# I mali della ridondanza

- La ridondanza è alla base di diversi problemi associati agli schemi relazionali
  - memorizzazione ridondante (spreco spazio memoria)
  - anomalie da
  - inserimento/cancellazione/aggiornamento
- Vincoli di integrità, in particolare dipendenze funzionali, possono essere usate per identificare gli schemi con tali problemi e per suggerire raffinamenti



# Un esempio

| SSN | Nome | pp | esperienza | paga_oraria | ore_lavorate |
|---|---|---|---|---|---|
| 123-22-3666 | Attishoo | 48 | 8 | 10 | 40 |
| 231-31-5368 | Smiley | 22 | 8 | 10 | 30 |
| 131-24-3650 | Smethurst | 35 | 5 | 7 | 30 |
| 434-26-3751 | Guldu | 35 | 5 | 7 | 32 |
| 612-67-4134 | Madayan | 35 | 8 | 10 | 40 |

- La chiave è SSN
- Ipotizziamo che *paga_oraria* sia determinato da *esperienza*
  - Questo genera ridondanze che possono creare problemi…

# Problemi generati dalle ridondanze

- Memorizzazione ridondante
  - La stessa associazione *esperienza-paga* è ripetuta più volte
- Anomalie da aggiornamento
  - Una modifica su una tupla potrebbe non essere seguita da modifiche alle tuple analoghe
- Anomalie da inserimento
  - Non possiamo inserire una tupla per un impiegato senza sapere il valore della paga associata al suo livello di esperienza
- Anomalie da cancellazione
  - Se cancelliamo tutte le tuple con un dato livello di esperienza, perdiamo l'informazione sulla paga associata



# Decomposizione di relazioni

- Principale tecnica di raffinamento decomposizione (sostituire ABCD con, per esempio, AB e BCD, oppure ACD e ABD)
- La decomposizione dovrebbe essere usata con giudizio
  - C'è una ragione per decomporre una relazione?
  - Che problemi causa (se ne causa) una decomposizione?

# Un esempio (segue)

## Possibile decomposizione

| SSN | Nome | pp | esperienza | ore_lavorate |
|---|---|---|---|---|
| 123-22-3666 | Attishoo | 48 | 8 | 40 |
| 231-31-5368 | Smiley | 22 | 8 | 30 |
| 131-24-3650 | Smethurst | 35 | 5 | 30 |
| 434-26-3751 | Guldu | 35 | 5 | 32 |
| 612-67-4134 | Madayan | 35 | 8 | 40 |

## Esperienza è chiave per la seconda relazione

| esperienza | paga_oraria |
|---|---|
| 8 | 10 |
| 8 | 10 |
| 5 | 7 |
| 5 | 7 |
| 8 | 10 |



# DIPENDENZE FUNZIONALI

# Dipendenze funzionali (DF)

Siano X e Y due sottoinsiemi di attributi di una relazione R, e $t_1$ e $t_2$ due tuple in *r*

- Una dipendenza funzionale X → Y vale su una relazione R se, per ogni istanza ammissibile *r* di R

  $\pi_X(t_1) = \pi_X(t_2)$ implica $\pi_Y(t_1) = \pi_Y(t_2)$

  cioè, date due tuple in *r*, se i valori di X sono uguali, allora anche i valori di Y devono essere uguali



# Dipendenze Funzionali (segue)

- Una DF è una affermazione su *tutte* le istanze di relazioni ammissibili per una data applicazione
  - Deve essere identificata in base alla *semantica*
  - Data una istanza ammissibile $r_1$ di R, possiamo controllare se essa viola qualche DF *f*, ma non possiamo dire se *f* vale su R!
- Dire che K è una chiave candidata per R significa che

  K → E (E indica l'insieme di attributi di R)
  - Però, K → E non richiede che K sia *minimale*!

# Chiusura di un insieme di DF

- Una DF *f* è implicata da un insieme *F* di DF se *f* è soddisfatta quando sono soddisfatte le DF in *F*
  - Date alcune DF (in genere quelle *semanticamente evidenti*), possiamo inferire DF aggiuntive
    - cf → rid, rid → parcheggio *implica* cf → parcheggio
- Si definisce chiusura di *F* ($F^+$) l'insieme di tutte le DF che sono implicate da *F*
- Regole di Armstrong per calcolare la chiusura
  *X, Y, Z sono insiemi di attributi*
  - **Riflessività**: Se $Y \subseteq X$, allora X → Y
  - **Aumento**: Se X → Y, allora XZ → YZ per ogni Z
  - **Transitività**: Se X → Y e Y → Z, allora X → Z



# Chiusura di un insieme di DF (segue)

- Regole addizionali (derivate da Regole Armstrong)
  - **Unione**: Se X → Y e X → Z, allora X → YZ
  - **Decomposizione**: Se X → YZ, allora X → Y e X → Z

Esempio: *Contratti*(cid, fid, gid, did, pid, qta, val)

- C è la chiave: C → CFGDPQV
- Un progetto (G) acquista ciascun pezzo (P) usando un singolo contratto: GP → C
- Un dipartimento (D) acquista al più un pezzo da un fornitore (F): FD → P

- GP → C, C → CFGDPQV implica GP → CFGDPQV
- FD → P implica FDG → GP
- FDG → GP e GP → CFGDPQV implica FDG → CFGDPQV

# Chiusura degli attributi

- Calcolare la chiusura di un insieme di DF può essere costoso
  - La dimensione della chiusura è esponenziale nel numero di attributi!
- Di solito è di interesse verificare se una data DF X → Y è nella chiusura di un insieme di DF *F*.
  - Si calcola la chiusura dell'attributo X (indicato con $X^+$) rispetto *F*
    - insieme di tutti gli attributi A tali che X → A sia in $F^+$
  - Si controlla se Y è in $X^+$
- *F* = {A → B, B → C, CD → E} implica A → E?
  - Cioè, A → E è nella chiusura $F^+$?
    In maniera equivalente, è E in $A^+$?

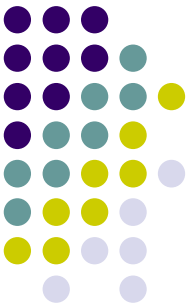


# Algoritmo per il calcolo della chiusura degli attributi

$X^+ = X$;
repeat
    old$X^+ = X^+$;
    for *ogni dipendenza funzionale* $U \rightarrow V$ in *F* do
        if $U \subseteq X^+$
        then $X^+ = X^+ \cup V$;
until ($X^+$ = old$X^+$);

Questo algoritmo può essere utilizzato per ottenere tutte le chiavi candidate di una relazione

*Si verifica per quali insiemi di attributi X, la chiusura contiene tutti gli attributi della relazione*



# Forme normali

- Se una relazione è in una certa forma normale (BCNF, 3NF, etc) siamo certi che alcuni problemi sono evitati/minimizzati.
  Ciò può aiutarci a decidere se decomporre la relazione sia utile.
- Ruolo delle DF nel rilevare la ridondanza
  - Consideriamo una relazione R con 3 attributi, ABC.
    - Non ci sono DF: non c'è ridondanza
    - È data A → B: diverse tuple potrebbero avere lo stesso valore A, e se ciò si verifica, esse avranno tutte lo stesso valore B!

# Superchiavi, chiavi e attributi primi

- Una superchiave di uno schema $R = \{A_1, A_2, \ldots, A_n\}$ è un insieme di attributi $S \subseteq R$ tale che non esistono due tuple $t_1$ e $t_2$ tali che $t_1[S] = t_2[S]$

  Una chiave *K* è una superchiave minimale

  la rimozione di un qualsiasi attributo fa perdere a *K* la proprietà di essere superchiave

  Uno schema può avere più *chiavi candidate*
- Un attributo di *R* è detto attributo primo se esso è membro di una *qualche chiave candidata* di *R.*



# Forme Normali

- Vedremo le 4 forme normali principali
  - BCNF (Boyce-Codd Normal Form)
  - 3NF (Third Normal Form)
  - 2NF (Second Normal Form)
  - 1NF (First Normal Form)
- Ciascuna forma normale implica le forme normali precedenti
- Una relazione soddisfa almeno la 1NF

  *Il dominio di ciascun attributo comprende solo* valori atomici (indivisibili) *e il valore di ciascun attributo in una tupla deve essere un* valore singolo *del dominio di quell'attributo*

# Violazione della 1NF attributi multivalore

**(a)**

**DEPARTMENT**

| Dname | Dnumber | Dmgr_ssn | Dlocations |
|---|---|---|---|

**(b)**

**DEPARTMENT**

| Dname | Dnumber | Dmgr_ssn | Dlocations |
|---|---|---|---|
| Research | 5 | 333445555 | {Bellaire, Sugarland, Houston} |
| Administration | 4 | 987654321 | {Stafford} |
| Headquarters | 1 | 888665555 | {Houston} |

**(c)**

**DEPARTMENT**

| Dname | Dnumber | Dmgr_ssn | Dlocation |
|---|---|---|---|
| Research | 5 | 333445555 | Bellaire |
| Research | 5 | 333445555 | Sugarland |
| Research | 5 | 333445555 | Houston |
| Administration | 4 | 987654321 | Stafford |
| Headquarters | 1 | 888665555 | Houston |

**Figure 10.8**
Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Example state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.



# Violazione della 1NF relazioni annidate

**(a)**

**EMP_PROJ**

| | | Projs | |
|---|---|---|---|
| Ssn | Ename | Pnumber | Hours |

**(b)**

**EMP_PROJ**

| Ssn | Ename | Pnumber | Hours |
|---|---|---|---|
| 123456789 | Smith, John B. | 1 | 32.5 |
| | | 2 | 7.5 |
| 666884444 | Narayan, Ramesh K. | 3 | 40.0 |
| 453453453 | English, Joyce A. | 1 | 20.0 |
| | | 2 | 20.0 |
| 333445555 | Wong, Franklin T. | 2 | 10.0 |
| | | 3 | 10.0 |
| | | 10 | 10.0 |
| | | 20 | 10.0 |
| 999887777 | Zelaya, AliciaJ. | 30 | 30.0 |
| | | 10 | 10.0 |
| 987987987 | Jabbar, Ahmad V. | 10 | 35.0 |
| | | 30 | 5.0 |
| 987654321 | Wallace, Jennifer S. | 30 | 20.0 |
| | | 20 | 15.0 |
| 888665555 | Borg, James E. | 20 | NULL |

**(c)**

**EMP_PROJ1**

| Ssn | Ename |
|---|---|

**EMP_PROJ2**

| Ssn | Pnumber | Hours |
|---|---|---|

**Figure 10.9**
Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a *nested relation* attribute PROJS. (b) Example extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP_PROJ into relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

# Seconda Forma Normale (2NF)

- Una relazione *R* con DF *F* è in 2NF se
  - *ogni* attributo non-primo *A* di *R* dipende funzionalmente in modo *completo* da *ogni chiave* di *R*
- X → A è una dipendenza funzionalmente completa se eliminando qualunque attributo Y da X la dipendenza funzionale non sussiste più, cioè (X – {Y}) → A non è una DF di *R*
- Dipendenze funzionalmente *non* complete sono dette dipendenze parziali
- In una relazione in 2NF ogni attributo non-primo non è parzialmente dipendente da alcuna chiave



# Dipendenze transitive

- La definizione di *dipendenza transitiva* è utile per illustrare la terza forma normale (3NF)

- Una dipendenza funzionale X → A è una dipendenza transitiva se esiste un insieme di attributi Z che
  - non è una chiave candidata
  - non è un sottoinsieme di una chiave

  tale che vale

  X → Z e Z → A

# Terza Forma Normale (3NF)

- Una relazione è in 3NF se soddisfa la 2NF e nessun attributo non-primo dipende in modo transitivo da qualche chiave (Codd, 1972)
- Definizione generale
  Una relazione R con DF *F* è in 3NF se, per tutte le $X \rightarrow A$ in $F^+$
  - $A \subseteq X$ (DF banale) *oppure*
  - X è una superchiave di R *oppure*
  - A è un attributo primo di R



# Violazione della 3NF

- La 3NF viene violata se esiste una DF $X \rightarrow A$ che non soddisfa *entrambe* le condizioni della 3NF. I casi possibili sono i seguenti
  - X non è chiave né sottoinsieme di chiave e A è attributo non-primo
    - C'è una catena di DF $K \rightarrow X$ $X \rightarrow A$, cioè una **dipendenza transitiva**
  - X è un sottoinsieme di qualche chiave K
    - Memorizziamo la coppia (X, A) in maniera ridondante
    - è una **dipendenza parziale** (è violata anche 2NF)

# Violazione della 3NF

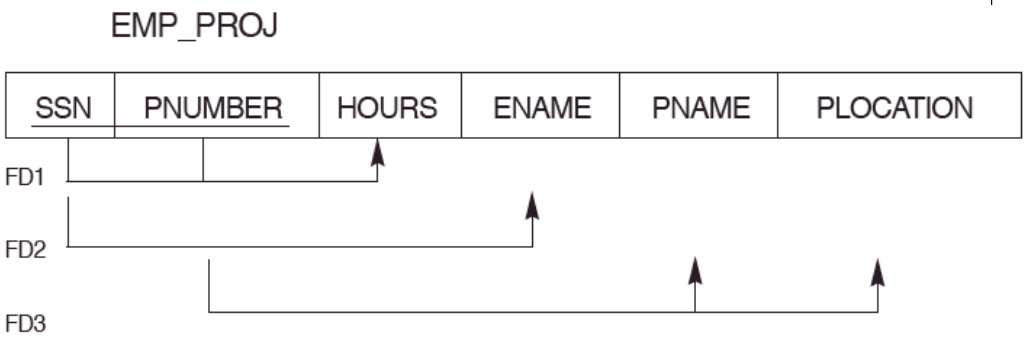

L'attributo non primo DMGRSSN dipende transitivamente dalla chiave SSN

SSN → DMGRSSN

SSN → DNUMBER e DNUMBER → DMGRSSN



# Violazione della 2NF




La chiave è (SSN,PNUMBER), ma valgono le DF parziali

- SSN → ENAME
- PNUMBER → PNAME
- PNUMBER → PLOCATION

# Forma normale di Boyce-Codd (BCNF)

- Una relazione R con DF *F* è in BCNF se, per tutte le $X \rightarrow A$ in $F^+$
  - $A \subseteq X$ (chiamata DF banale) oppure
  - X è una superchiave di R
- R è in BCNF se le sole DF non banali che valgono su R sono vincoli di chiave
  - Non ci possono essere ridondanze in R, non essendoci altre DF diverse dai vincoli di chiave



# Relazione fra BCNF e 3NF

- Se R è in BCNF, ovviamente è in 3NF
- Se R è in 3NF, qualche ridondanza è possibile.
  - È un compromesso, usato quando la BCNF non è ottenibile
    - ad esempio, non c'è una decomposizione "buona", oppure per considerazioni relative alle prestazioni
- È sempre possibile decomporre una relazione R in una collezione di relazioni 3NF *senza-perdita* che conservi le dipendenze

# Una relazione in 3NF che non soddisfa la BCNF

**TEACH**

| Student | Course | Instructor |
|---|---|---|
| Narayan | Database | Mark |
| Smith | Database | Navathe |
| Smith | Operating Systems | Ammar |
| Smith | Theory | Schulman |
| Wallace | Database | Mark |
| Wallace | Operating Systems | Ahamad |
| Wong | Database | Omiecinski |
| Zelaya | Database | Navathe |
| Narayan | Operating Systems | Ammar |



# Determinare la forma normale di una relazione

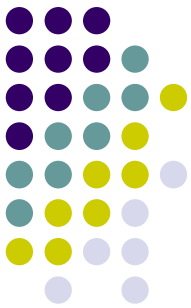
- Si inizia a verificare se la relazione soddisfa la BCNF
- Se la relazione non è in BCNF, si verifica se soddisfa la 3NF
- Se la relazione non è in 3NF, si verifica se soddisfa la 2NF
- Se la relazione non soddisfa neanche la 2NF, allora è in 1NF

# Sintesi forme normali

| Forma normale | Verifica | Rimedio (Normalizzazione) |
|---|---|---|
| **Prima (1NF)** | Le relazioni non devono contenere attributi multivalore o relazioni annidate | Creare nuove relazioni per ciascun attributo multivalore o per ciascuna relazione annidata |
| **Seconda (2NF)** | Nel caso di relazioni in cui la chiave primaria contiene più attributi, nessun attributo non-primo deve essere funzionalmente dipendente da una parte della chiave primaria | Decomporre creando una nuova relazione per ciascuna chiave parziale con i relativi attributi dipendenti. Deve rimanere anche una relazione che contiene la chiave primaria originaria e tutti gli attributi che sono dipendenti in modo completo dalla chiave |
| **Terza (3NF)** | La relazione non deve contenere attributi non-primi che siano funzionalmente dipendenti da altri attributi non-primi. | Decomporre creando una relazione che contenga gli attributi non-primi che determinano funzionalmente altri attributi non primi |



# PROPRIETÀ DELLE DECOMPOSIZIONI



# Decomposizione di uno schema di relazione

- La relazione R contenga attributi A1... An.
- Una decomposizione di R consiste nella sostituzione di R con due o più relazioni tali che
  - Ciascun nuovo schema contiene un sottoinsieme degli attributi di R (e nessun attributo che non appartiene a R)
  - Ogni attributo di R appare come attributo di una delle nuove relazioni
- Proiezione di un insieme *F* di DF
  - se R viene decomposta in X e Y, la proiezione di $F^+$ su X ($F_X$) è l'insieme di DF U → V in $F^+$ tale che U, V siano in X



# Esempio di decomposizione

- Relazione R(C,N,P,E,O,L)
  - CNPEOL ha DF C → CNPEOL e E → O
  - La seconda DF causa la violazione della 3NF
    - valori di O ripetutamente associati a valori di E.
  - Il modo più facile per risolverla è
    - creare una relazione EO per memorizzare queste associazioni
    - rimuovere O dallo schema principale cioè, decomporre CNPEOL in CNPEL e EO
- Se memorizziamo solo le proiezioni di queste tuple su CNPEL e EO, ci sono potenziali problemi cui dovremmo prestare attenzione?

# Problemi con le decomposizioni

- Ci sono tre potenziali problemi da considerare
  1. Alcune interrogazioni possono diventare costose
     - "Quanto guadagna Joe?" (salario = O*L)
  2. Dalle istanze delle relazioni decomposte, potremmo non essere in grado di ricostruire la corrispondente istanza della relazione originale!
  3. Il controllo di alcune dipendenze potrebbe richiedere un join fra le istanze delle relazioni decomposte
- Dobbiamo fare un compromesso fra la potenziale presenza di questi problemi e l'eliminazione delle ridondanze



# Decomposizioni senza-perdita

- La decomposizione di R in X e Y è *senza-perdita* rispetto a un insieme *F* di DF se, per ogni istanza *r* che soddisfa *F*

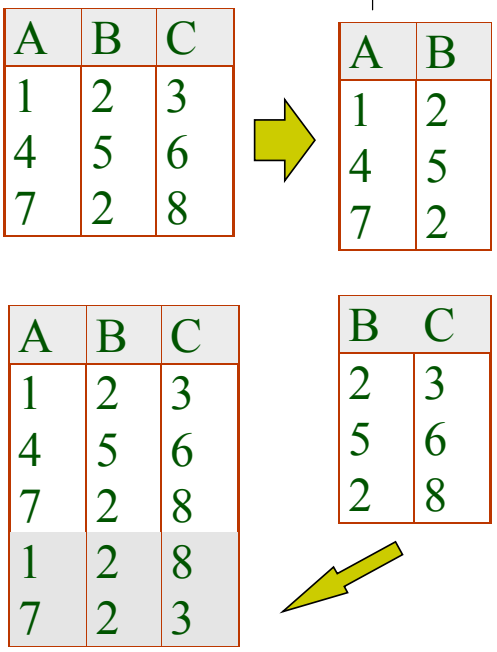
  $\pi_X(r) \bowtie \pi_Y(r) = r$
- È sempre vero che $r \subseteq \pi_X(r) \bowtie \pi_Y(r)$
  - Se l'uguaglianza è verificata la decomposizione è *senza-perdita*
- La definizione si estende facilmente a decomposizioni in 3 o più relazioni
- È essenziale che tutte le decomposizioni usate per risolvere la ridondanza siano *senza-perdita*!
  - Si evita il problema (2)

# Decomposizioni senza-perdita

- La decomposizione di R in X e Y è *senza-perdita* rispetto a *F* se e solo se $F^+$ contiene
  - $X \cap Y \rightarrow X$, oppure
  - $X \cap Y \rightarrow Y$

    gli attributi comuni formano una chiave per X o Y

- Se $U \rightarrow V$ vale su R e $U \cap V$ è vuota la decomposizione di R in UV e R – V è *senza-perdita*






# Decomposizioni che conservano le dipendenze

- Consideriamo CFGDPQV
  C è la chiave, $GP \rightarrow C$ e $FD \rightarrow P$
  - Decomposizione BCNF: CFGDQV e FDP
  - Problema: controllare $GP \rightarrow C$ richiede un join!
- Decomposizione che conserva le dipendenze (intuitiva)
  - se R è decomposta in X, Y e Z, e garantiamo che valgano le DF su X, su Y e su Z, allora devono valere anche tutte le DF che valevano su R
    - si evita il problema (3)

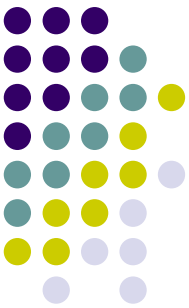
# Decomposizioni che conservano le dipendenze (segue)

- La decomposizione di R in X e Y *conserva le dipendenze* se $(F_X \cup F_Y)^+ = F^+$
- È importante considerare $F^+$, non $F$, in questa definizione
  - ABC, A → B, B → C, C → A, decomposta in AB e BC
  - Le dipendenze sono conservate? Viene conservata C → A?
- La conservazione delle dipendenze non implica la proprietà *senza-perdita*
  - ABC, A → B, decomposta in AB e BC

  e viceversa!



## NORMALIZZAZIONE



# Decomposizione in BCNF

- Se X → Y (Y singolo attributo) viola la BCNF, si decompone R in R – Y e XY
  - Applicazioni ripetute producono una collezione di relazioni in BCNF
  - Decomposizione senza-perdita
- Ad esempio
  CFGDPQV, chiave C, GP → C, FD → P, G → F
  - Per eliminare FD → P, decomponiamo in FDP, CFGDQV
  - Per eliminare G → F, decomponiamo CFGDQV in GF e CGDQV
- L'ordine in cui si analizzano le dipendenze che violano la BCNF può portare a decomposizioni diverse



# BCNF e conservazione delle dipendenze

- Potrebbe non esserci una decomposizione in BCNF che conserva le dipendenze
  - Ad esempio, CSZ, CS → Z, Z →C
  - Non si può decomporre conservando la prima DF; non in BCNF
- Analogamente, la decomposizione di CFGDPQV in FDP, GF e CGDQV non conserva le dipendenze (rispetto alle DF GP → C, FD → P e G → F)
  - È una decomposizione senza-perdita
  - In questo caso, aggiungere GPC alla collezione di relazioni dà una decomposizione che conserva le dipendenze
    - Le tuple GPC sono memorizzate solo per controllare la DF! (Ridondanza!)

# Decomposizione in 3NF

- L'algoritmo per la decomposizione senza-perdita in BCNF può essere usato per ottenere una decomposizione senza-perdita in 3NF
- Garantire la conservazione delle dipendenze
  - Se X → Y non viene conservata, aggiungere la relazione XY
  - Ma XY potrebbe violare la 3NF!
    Ad esempio, se si aggiunge CGP per *conservare* GP → C, che succede se abbiamo anche G → C?
- Necessità di un approccio differente
  Invece di *F*, l'insieme di DF dato, usiamo una copertura minima di *F*



# Copertura di un insieme di DF

- Un insieme *F* di dipendenze funzionali copre un altro insieme *E* di dipendenze funzionali, se ogni DF in *E* è presente anche in $F^+$
  - Si dice anche che *E* è coperto da *F*
- In altre parole, gli insiemi di DF *E* e *F* sono equivalenti

# Copertura minima per un insieme di DF

- Copertura minima *G* per un insieme *F* di DF
  - $F^+ = G^+$
  - La parte destra di ogni DF in *G* è un singolo attributo
  - Se modifichiamo *G* cancellando una DF o cancellando attributi da una DF in *G*, la chiusura cambia
- Intuitivamente, ogni DF in G è necessaria, e "quanto più piccola possibile" allo scopo di ottenere la stessa chiusura di *F*
- Ad esempio
  A → B, ABCD → E, EF → GH, ACDF → EG
  ha la seguente copertura minima
  A → B, ACD → E, EF → G e EF → H



# Algoritmo per ottenere la copertura minima

1. Porre le DF in forma standard
   - cioè con un attributo singolo nella parte destra
2. Minimizzare la parte sinistra di ogni DF
3. Cancellare le DF ridondanti

- L'ordine in cui si esaminano le DF può portare a coperture minime diverse

# Copertura minima
# minimizzazione parte sinistra DF

Per ogni DF $X \rightarrow A$ in $F$

Per ogni attributo $B \subset X$

if $\{F - (X \rightarrow A)\} \cup \{(X - \{B\}) \rightarrow A\}$ è equivalente a $F$

then sostituisci $X \rightarrow A$ con $(X - \{B\}) \rightarrow A$ in $F$



# Copertura minima
# cancellazione DF ridondanti

Per ogni dipendenza funzionale $X \rightarrow A$ in $F$

if $\{F - (X \rightarrow A)\}$ è equivalente a $F$

then rimuovi $(X \rightarrow A)$ da $F$

# Esempio

Data $F$ = {B → A, D → A, AB → D}

1. le dipendenze sono già in forma canonica
2. per aumento da B → A abbiamo B → AB
   per transitività abbiamo B → D
   possiamo sostituire AB → D con B → D
3. per transitività B → D e D → A implicano B → A,
   che è dunque ridondante

La copertura minima di $F$ è quindi

{B → D, D → A}



# Decomposizione in 3NF

## senza perdita e con conservazione delle dipendenze

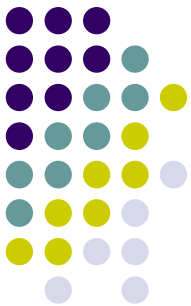
- Dati
  - una relazione $R$ con un insieme $F$ dei DF che sia una copertura minima
  - una decomposizione senza perdita di R, $R_1$, $R_2$,…,$R_n$, ciascuna $R_i$ in 3NF
  - l'insieme di dipendenze di $F$ non conservato, $N$
- Per ciascuna DF $X \rightarrow A$ in $N$, aggiungiamo alla decomposizione la schema di relazione $XA$
- Ottimizzazione: se abbiamo diversi schemi $X \rightarrow A_1$, $X \rightarrow A_2$,… possiamo sostituirli con $X \rightarrow A_1A_2A_3\ldots A_n$

# Sintesi 3NF

- Approccio alternativo al precedente
  - Dato uno schema *R* e una copertura minimale *F* di DF
  - si costruisce la decomposizione R aggiungendo uno schema *XA* per tutte le DF $X \rightarrow A_i$ in *F* (A = $\{A_1, A_2, \ldots, A_n\}$)
  - La decomposizione non è senza perdita se nessuna relazione contiene una chiave di R. Allora si crea un ulteriore schema con attributi che formano una chiave di *R*
  - Si eliminano eventuali relazioni ridondanti



# RAFFINAMENTO DI SCHEMI ER



# Schemi ER e DF

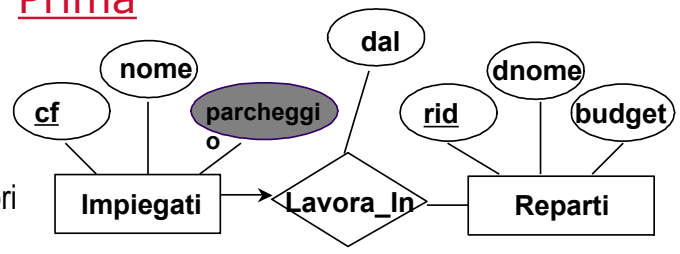
- In uno schema ER possiamo evidenziare solo vincoli di chiave
  - Le DF non sono rappresentabili
    - Vincoli su entità
    - Vincoli su relazioni
- La conoscenza delle DF consente di raffinare gli schemi ER
  - Identificare gli attributi delle entità
  - Identificare le entità



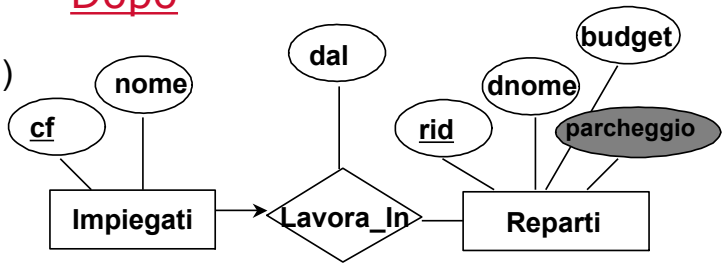
# Raffinamento di uno schema ER
## Esempio

- Prima traduzione dello schema ER
  *Lavoratori*(C, N, P, R, D)
  *Reparto*(R, M, B)
  - Parcheggi associati ai lavoratori
- Supponiamo che a tutti i lavoratori di un reparto sia assegnato lo stesso parcheggio: R → P
  Ridondanza risolta da
  *Lavoratori2*(C, N, R, D)
  *Parcheggi_Rep*(R, P)
- Miglioramento
  Lavoratori2(C, N, R, D)
  Reparti(R, M, B, P)

Prima




Dopo



# Riepilogo

- Se una relazione è in BCNF, è priva di ridondanze. Quindi, cercare di garantire che tutte le relazioni siano in BCNF è una buona euristica.
- Se una relazione non è in BCNF, possiamo provare a decomporla in una collezione di relazioni BCNF
  - Dobbiamo considerare se *tutte le DF sono conservate*.
  - Se una decomposizione in BCNF senza-perdita che conserva le dipendenze non è possibile (o non adatta, date le interrogazioni tipiche), dovremmo considerare una decomposizione in 3NF
  - Le decomposizioni dovrebbero essere eseguite e/o riesaminate tenendo in conto i requisiti sulle prestazioni