Esonero 09/09/2020

1. **Esonero SQL V1**

   Le seguenti relazioni definiscono una base di dati "MustEat" per gestire le ordinazioni e le consegne di pasti a domicilio. Gli attributi sottolineati sono le chiavi primarie delle relazioni.

   UTENTE(Codice, Nome, Email, Città, IndirizzoPreferito)

   INDIRIZZO(Utente, Indirizzo, Note)

   ORDINE(Utente, Ristorante, DataOrdine, Prezzo, IndirizzoConsegna)

   RISTORANTE(Codice, Nome, Tipo, Indirizzo, Città)

   Vincoli di integrità referenziale:
   INDIRIZZO(Utente) referenzia UTENTE(Codice),
   UTENTE(Codice, IndirizzoPreferito) referenzia INDIRIZZO(Utente, Indirizzo),
   ORDINE(Ristorante) referenzia RISTORANTE(Codice) e
   ORDINE(Utente, IndirizzoConsegna) referenzia INDIRIZZO(Utente, Indirizzo).

   Significato degli attributi: *IndirizzoPreferito* è l'indirizzo di consegna preferito per un dato utente; *Tipo* può assumere i valori *pizzeria*, *paninoteca*, *cucina etnica* e *cucina vegetariana*; *Prezzo* è il prezzo totale di un ordine. I rimanenti attributi sono autoesplicativi. Gli attributi sono tutti NOT NULL.

   Esprimere in SQL tutte e tre le seguenti interrogazioni indicando a quale query si sta rispondendo (A, B, C):

   (A - Bassa complessità) Riportare, ordinate per ordine alfabetico, le città da cui sono partiti ordini per ristoranti di cucina etnica o cucina vegetariana di Torino.

   (B - Media complessità) Indicare, per ogni utente, la massima spesa per un singolo ordine per ciascun tipo di ristorante.

   (C - Alta complessità) Trovare le pizzerie che hanno consegnato solo a utenti della propria città e i cui ordini sono in media non superiori a 30 euro.

   Soluzioni.

   A.

```
select distinct u.città CittaUtente
from utente u join ordine o on u.codice=o.utente join ristorante r on o.ristorante=r.codice
where (tipo='cucina etnica' or tipo='cucina vegetariana') and r.città='Torino'
order by CittaUtente;
```

   B.

```
select Utente, Tipo, max(Prezzo) as MaxPrezzo
from ORDINE JOIN RISTORANTE On ORDINE.Ristorante = RISTORANTE.Codice
group by Utente, Tipo;
```

   C.

```
select Ristorante, avg(Prezzo) AvgPrezzo
from ORDINE join RISTORANTE on ORDINE.Ristorante = RISTORANTE.Codice
where Ristorante
not in (
select RISTORANTE.codice
FROM RISTORANTE join ORDINE on RISTORANTE.Codice=ORDINE.Ristorante join UTENTE
on ORDINE.Utente = UTENTE.Codice
where RISTORANTE.Citta <> UTENTE.Citta
)
```

and RISTORANTE.Tipo = 'pizzeria'
group by Ristorante
having AvgPrezzo <=30;

2. **Esonero SQL V2**

Le seguenti relazioni definiscono una base di dati "MustEat" per gestire le ordinazioni e le consegne di pasti a domicilio. Gli attributi sottolineati sono le chiavi primarie delle relazioni.

UTENTE(Codice, Nome, Email, Città, IndirizzoPreferito)

INDIRIZZO(Utente, Indirizzo, Note)

ORDINE(Utente, Ristorante, DataOrdine, Prezzo, IndirizzoConsegna)

RISTORANTE(Codice, Nome, Tipo, Indirizzo, Città)

Vincoli di integrità referenziale:
INDIRIZZO(Utente) referenzia UTENTE(Codice),
UTENTE(Codice, IndirizzoPreferito) referenzia INDIRIZZO(Utente, Indirizzo),
ORDINE(Ristorante) referenzia RISTORANTE(Codice) e
ORDINE(Utente, IndirizzoConsegna) referenzia INDIRIZZO(Utente, Indirizzo).

Significato degli attributi: *IndirizzoPreferito* è l'indirizzo di consegna preferito per un dato utente; *Tipo* può assumere i valori *pizzeria*, *paninoteca*, *cucina etnica* e *cucina vegetariana*; *Prezzo* è il prezzo totale di un ordine. I rimanenti attributi sono autoesplicativi. Gli attributi sono tutti NOT NULL.

Esprimere in SQL tutte e tre le seguenti interrogazioni indicando a quale query si sta rispondendo (A, B, C):

(A - Bassa complessità) Riportare, ordinate per ordine alfabetico inverso, le città da cui sono partiti ordini per pizzerie situate in una città diversa.

(B - Media complessità) Indicare per ogni città da cui siano partiti ordini, la spesa media per ciascun tipo di ristorante, escludendo le paninoteche.

(C - Alta complessità) Tra gli utenti che hanno ordinato da almeno due ristoranti diversi dello stesso tipo, trovare l'utente che ha speso di più in un singolo ordine.

Soluzioni.

A.

```
select distinct u.città CittaUtente
from utente u join ordine o on u.codice=o.utente join ristorante r on o.ristorante=r.codice
where tipo='pizzeria' and r.città <>u.città
order by CittaUtente desc;
```

B.

```
select u.città, Tipo, avg(Prezzo) as PrezzoMedio
from UTENTE u join ORDINE o on u.codice=o.utente JOIN RISTORANTE r On o.Ristorante =
r.Codice
where r.Tipo <> 'paninoteca'
group by u.città, Tipo;
```

C.

```
with UtentiDueRist as (
select *
from UTENTE u
```

```
join ORDINE o1 on o1.Utente = u.Codice
join RISTORANTE r1 on o1.Ristorante = r1.Codice
join ORDINE o2 on o2.Utente = u.Codice
join RISTORANTE r2 on o2.Ristorante = r2.Codice
where r1.Codice > r2.Codice
and r1.Tipo = r2.Tipo)
select max(Prezzo) MaxPrezzo, o.Utente
from ORDINE o
where o.Utente in (select u.Codice from UtentiDueRist)
group by ORDINE.Utente
having MaxPrezzo >= ALL (
select Prezzo from ORDINE
where o.Utente in (select u.Codice from UtentiDueRist));
```

3. **Esonero Algebra V1**

   Le seguenti relazioni definiscono una base di dati "MustEat" per gestire le ordinazioni e le consegne di pasti a domicilio. Gli attributi sottolineati sono le chiavi primarie delle relazioni.

   UTENTE(Codice, Nome, Email, Città, IndirizzoPreferito)

   INDIRIZZO(Utente, Indirizzo, Note)

   ORDINE(Utente, Ristorante, DataOrdine, Prezzo, IndirizzoConsegna)

   RISTORANTE(Codice, Nome, Tipo, Indirizzo, Città)

   Vincoli di integrità referenziale:
   INDIRIZZO(Utente) referenzia UTENTE(Codice),
   UTENTE(Codice, IndirizzoPreferito) referenzia INDIRIZZO(Utente, Indirizzo),
   ORDINE(Ristorante) referenzia RISTORANTE(Codice) e
   ORDINE(Utente, IndirizzoConsegna) referenzia INDIRIZZO(Utente, Indirizzo).

   Significato degli attributi: *IndirizzoPreferito* è l'indirizzo di consegna preferito per un dato utente; *Tipo* può assumere i valori *pizzeria*, *paninoteca*, *cucina etnica* e *cucina vegetariana*; *Prezzo* è il costo totale di un ordine. I rimanenti attributi sono autoesplicativi. Gli attributi sono tutti NOT NULL.

   Esprimere tutte e due le seguenti interrogazioni indicando a quale query si sta rispondendo (per facilità di scrittura delle formule, si possono eseguire gli esercizi con carta e penna e fare l'upload delle foto prestando attenzione che lo svolgimento sia chiaramente leggibile):

   (A - Algebra Relazionale) Elencare i ristoranti di tipo etnico che hanno effettuato consegne a tutti gli indirizzi.

   (B - Calcolo Relazionale su tuple con dichiarazione di range) Elencare nome e tipo dei ristoranti che hanno effettuato almeno due consegne al loro stesso indirizzo.

   Soluzioni.

   A. $\pi_{Codice,IndirizzoConsegna}(ordine \bowtie_{Ristorante=Codice} \sigma_{Tipo='cucina\ etnica'}(ristorante)) \div \pi_{IndirizzoConsegna}(ordine)$

   B. $\{r.(Nome, Tipo) | r(ristorante) |$
   $\exists o(ordine)(r.Codice = o.Ristorante \wedge o.IndirizzoConsegna = r.Indirizzo)$
   $\wedge \exists o'(ordine)(o'.Ristorante = o'.Codice \wedge r.Indirizzo = o'.IndirizzoConsegna)$
   $\wedge (o'.Utente \neq o.Utente \vee o'.DataOrdine \neq o.DataOrdine))\}$

4. **Esonero Algebra V2**

   Le seguenti relazioni definiscono una base di dati "MustEat" per gestire le ordinazioni e le consegne di pasti a domicilio. Gli attributi sottolineati sono le chiavi primarie delle relazioni.

UTENTE(Codice, Nome, Email, Città, IndirizzoPreferito)

INDIRIZZO(Utente, Indirizzo, Note)

ORDINE(Utente, Ristorante, DataOrdine, Prezzo, IndirizzoConsegna)

RISTORANTE(Codice, Nome, Tipo, Indirizzo, Città)

Vincoli di integrità referenziale:
INDIRIZZO(Utente) referenzia UTENTE(Codice),
UTENTE(Codice, IndirizzoPreferito) referenzia INDIRIZZO(Utente, Indirizzo),
ORDINE(Ristorante) referenzia RISTORANTE(Codice) e
ORDINE(Utente, IndirizzoConsegna) referenzia INDIRIZZO(Utente, Indirizzo).

Significato degli attributi: *IndirizzoPreferito* è l'indirizzo di consegna preferito per un dato utente; *Tipo* può assumere i valori *pizzeria*, *paninoteca*, *cucina etnica* e *cucina vegetariana*; *Prezzo* è il costo totale di un ordine. I rimanenti attributi sono autoesplicativi. Gli attributi sono tutti NOT NULL.

Esprimere tutte e due le seguenti interrogazioni indicando a quale query si sta rispondendo (per facilità di scrittura delle formule, si possono eseguire gli esercizi con carta e penna e fare l'upload delle foto prestando attenzione che lo svolgimento sia chiaramente leggibile):

(A - Algebra Relazionale) Elencare le pizzerie che hanno effettuato consegne a tutti gli utenti.

(B - Calcolo Relazionale su tuple con dichiarazione di range) Elencare le città degli utenti che hanno effettuato ordini esclusivamente in pizzerie.

Soluzioni.

A. $\pi_{Ristorante,Utente}(ordine \bowtie_{Ristorante=Codice} \sigma_{Tipo='pizzeria'}(ristorante)) \div \rho_{Utente \leftarrow Codice}(\pi_{Codice}(utente))$

B. $\{u.Citta | u(utente) |$
$\forall o(ordine)(u.Codice = o.Utente \Rightarrow \forall r(ristorante)(o.Ristorante = r.Codice \Rightarrow r.Tipo =' pizzeria'))\}$

5. **Esonero Ottimizzazione V1**

Le seguenti relazioni definiscono una base di dati "MustEat" per gestire le ordinazioni e le consegne di pasti a domicilio. Gli attributi sottolineati sono le chiavi primarie delle relazioni.

UTENTE(Codice, Nome, Email, Città, IndirizzoPreferito)

INDIRIZZO(Utente, Indirizzo, Note)

ORDINE(Utente, Ristorante, DataOrdine, Prezzo, IndirizzoConsegna)

RISTORANTE(Codice, Nome, Tipo, Indirizzo, Città)

Vincoli di integrità referenziale:
INDIRIZZO(Utente) referenzia UTENTE(Codice),
UTENTE(Codice, IndirizzoPreferito) referenzia INDIRIZZO(Utente, Indirizzo),
ORDINE(Ristorante) referenzia RISTORANTE(Codice) e
ORDINE(Utente, IndirizzoConsegna) referenzia INDIRIZZO(Utente, Indirizzo).

Significato degli attributi: *IndirizzoPreferito* è l'indirizzo di consegna preferito per un dato utente; *Tipo* può assumere i valori *pizzeria*, *paninoteca*, *cucina etnica* e *cucina vegetariana*; *Prezzo* è il costo totale di un ordine. I rimanenti attributi sono autoesplicativi. Gli attributi sono tutti NOT NULL.

Data la seguente query sulla base di dati MustEat: $\sigma_{Email='dtrump@whitehouse.gov' \wedge ristorante.Citta='WashingtonD.C.'}((ut$
$ordine)$
$\bowtie_{Ristorante=ristorante.Codice} ristorante)$

disegnare gli alberi sintattici prima e dopo l'ottimizzazione logica. Per semplicità non considerare le varianti di parsificazione date dalla proprietà associativa dei join.

Inoltre calcolare il numero di tuple "mosse" prima e dopo l'ottimizzazione logica.

Si svolgano i calcoli sapendo che:

CARD(ordine) = 1 000 000

CARD(utente) = VAL(Utente,ordine) = 50 000

CARD(ristorante) = VAL(Ristorante,ordine) = 5 000

VAL(Email,utente) = 50 000

VAL(Citta,ristorante) = 100

Soluzioni.

$\sigma_{Email='dtrump@whitehouse.gov' \wedge ristorante.Citta='WashingtonD.C.'}((utente \bowtie_{utente.Codice=Utente} ordine) \bowtie_{Ristorante=ristorante.Codice} ristorante)$

Prima dell'ottimizzazione:

- Costo $r_1 = utente \bowtie_{utente.Codice=Utente} ordine$: 50 000 × 1 000 000 = 5×10^{10}.
- Cardinalità di $|r_1| = |utente \bowtie_{utente.Codice=Utente} ordine| = CARD(ordine)$ = 1 000 000 (equijoin attraverso la chiave esterna)
- Costo join $r_2 = r_1 \bowtie_{Ristorante=ristorante.Codice} ristorante$ = 1 000 000 × 5 000 = 5×10^9
- Cardinalità di $|r_2| = |r_1 \bowtie_{Ristorante=ristorante.Codice} ristorante| =$ $= \min\{\frac{1}{VAL(Ristorante,r1)}, \frac{1}{VAL(Codice,ristorante)}\} \times CARD(r1) \times CARD(ristorante) = \min\{\frac{1}{5\ 000}, \frac{1}{5\ 000}\} \times$ 1 000 000 × 5 000 = 10^6
- Costo della selezione: $|r_2| = 10^6$
- Costo totale = $5 \times 10^{10} + 5 \times 10^9 + 10^6 \approx 5 \times 10^{10}$

Dopo l'ottimizzazione:

$(\sigma_{Email='dtrump@whitehouse.gov'}(utente) \bowtie_{utente.Codice=Utente} ordine) \bowtie_{Ristorante=ristorante.Codice} \sigma_{ristorante.Citta='WashingtonD.C.'}(ristorante)$

- Costo $\sigma_{Email='dtrump@whitehouse.gov'}(utente) = 5 \times 10^5$
- Tuple prodotte dalla selezione $|\sigma_1| = |\sigma_{Email='dtrump@whitehouse.gov'}(utente)| = \frac{1}{VAL(Email,utente)} \times CARD(utente) = 1$
- Costo $\sigma_{Citta='WashingtonD.C.'}(ristorante) = 5 \times 10^3$
- Tuple prodotte dalla selezione $|\sigma_2| = |\sigma_{Citta='WashingtonD.C.'}(ristorante)| =$ $= \frac{1}{VAL(Citta,ristorante)} \times CARD(ristorante) = \frac{1}{100} \times 5\ 000 = 50$
- Costo $r_1 = \sigma_1 \bowtie_{utente.Codice=Utente} ordine$: 1 × 1 000 000 = 10^6.
- Cardinalità di $|r_1| =$ $= \min\{\frac{1}{\min\{VAL(Codice,utente), CARD(\sigma_1)\}}, \frac{1}{VAL(Utente,ordine)}\} \times CARD(\sigma_1) \times CARD(ordine) =$ $= \min\{\frac{1}{\min\{50\ 000,1\}}, \frac{1}{50\ 000}\} \times 1 \times 1\ 000\ 000 = \min\{\frac{1}{1}, \frac{1}{50\ 000}\} \times 1 \times 1\ 000\ 000 = 20$
- Costo join $r_2 = r_1 \bowtie_{Ristorante=ristorante.Codice} \sigma_2 = 20 \times 50 = 10^3$
- Costo totale = $5 \times 10^5 + 5 \times 10^3 + 10^6 + 10^3 \approx 10^6$

6. **Esonero Ottimizzazione V2**

Le seguenti relazioni definiscono una base di dati "MustEat" per gestire le ordinazioni e le consegne di pasti a domicilio. Gli attributi sottolineati sono le chiavi primarie delle relazioni.

UTENTE(Codice, Nome, Email, Città, IndirizzoPreferito)

INDIRIZZO(Utente, Indirizzo, Note)

ORDINE(Utente, Ristorante, DataOrdine, Prezzo, IndirizzoConsegna)

RISTORANTE(Codice, Nome, Tipo, Indirizzo, Città)

Vincoli di integrità referenziale:
INDIRIZZO(Utente) referenzia UTENTE(Codice),
UTENTE(Codice, IndirizzoPreferito) referenzia INDIRIZZO(Utente, Indirizzo),
ORDINE(Ristorante) referenzia RISTORANTE(Codice) e
ORDINE(Utente, IndirizzoConsegna) referenzia INDIRIZZO(Utente, Indirizzo).

Significato degli attributi: *IndirizzoPreferito* è l'indirizzo di consegna preferito per un dato utente; *Tipo* può assumere i valori *pizzeria*, *paninoteca*, *cucina etnica* e *cucina vegetariana*; *Prezzo* è il costo totale di un ordine. I rimanenti attributi sono autoesplicativi. Gli attributi sono tutti NOT NULL.

Data la seguente query sulla base di dati MustEat:

$\sigma_{Email='stanislawlem@solaris.org' \wedge ristorante.Citta='Cracovia'}((utente \bowtie_{utente.Codice=Utente} ordine) \bowtie_{Ristorante=ristorante.Codice} ristorante)$

disegnare gli alberi sintattici prima e dopo l'ottimizzazione logica. Per semplicità non considerare le varianti di parsificazione date dalla proprietà associativa dei join.

Inoltre calcolare il numero di tuple "mosse" prima e dopo l'ottimizzazione logica.

Si svolgano i calcoli sapendo che:

CARD(ordine) = 1 000 000

CARD(utente) = VAL(Utente,ordine) = 50 000

CARD(ristorante) = VAL(Ristorante,ordine) = 5 000

VAL(Email,utente) = 50 000

VAL(Citta,ristorante) = 100

Soluzioni.

$\sigma_{Email='stanislawlem@solaris.org' \wedge ristorante.Citta='Cracovia'}((utente \bowtie_{utente.Codice=Utente} ordine) \bowtie_{Ristorante=ristorante.Codice} ristorante)$

Prima dell'ottimizzazione:

- Costo $r_1 = utente \bowtie_{utente.Codice=Utente} ordine$: 50 000 × 1 000 000 = 5×10^{10}.
- Cardinalità di $|r_1| = |utente \bowtie_{utente.Codice=Utente} ordine| = CARD(ordine) = 1\ 000\ 000$ (equijoin attraverso la chiave esterna)
- Costo join $r_2 = r_1 \bowtie_{Ristorante=ristorante.Codice} ristorante = 1\ 000\ 000 \times 5\ 000 = 5 \times 10^9$
- Cardinalità di $|r_2| = |r_1 \bowtie_{Ristorante=ristorante.Codice} ristorante| =$
  $= \min\{\frac{1}{VAL(Ristorante,r1)}, \frac{1}{VAL(Codice,ristorante)}\} \times CARD(r1) \times CARD(ristorante) = \min\{\frac{1}{5\ 000}, \frac{1}{5\ 000}\} \times 1\ 000\ 000 \times 5\ 000 = 10^6$
- Costo della selezione: $|r_2| = 10^6$
- Costo totale = $5 \times 10^{10} + 5 \times 10^9 + 10^6 \approx 5 \times 10^{10}$

Dopo l'ottimizzazione:

$(\sigma_{Email='stanislawlem@solaris.org'}(utente) \bowtie_{utente.Codice=Utente} ordine) \bowtie_{Ristorante=ristorante.Codice} \sigma_{ristorante.Citta='Cracovia'}(ristorante)$

- Costo $\sigma_{Email='stanislawlem@solaris.org'}(utente) = 5 \times 10^5$
- Tuple prodotte dalla selezione $|\sigma_1| = |\sigma_{Email='stanislawlem@solaris.org'}(utente)| = \frac{1}{VAL(Email,utente)} \times CARD(utente) = 1$

- Costo $\sigma_{Citta='Cracovia'}(ristorante) = 5 \times 10^3$
- Tuple prodotte dalla selezione $|\sigma_2| = |\sigma_{Citta='Cracovia'}(ristorante)| =$
  $= \frac{1}{VAL(Citta,ristorante)} \times CARD(ristorante) = \frac{1}{100} \times 5\ 000 = 50$
- Costo $r_1 = \sigma_1 \bowtie_{utente.Codice=Utente} ordine$: $1 \times 1\ 000\ 000 = 10^6$.
- Cardinalità di $|r_1| =$
  $= \min\{\frac{1}{\min\{VAL(Codice,utente),CARD(\sigma_1)\}}, \frac{1}{VAL(Utente,ordine)}\} \times CARD(\sigma_1) \times CARD(ordine) =$
  $= \min\{\frac{1}{\min\{50\ 000,1\}}, \frac{1}{50\ 000}\} \times 1 \times 1\ 000\ 000 = \min\{\frac{1}{1}, \frac{1}{50\ 000}\} \times 1 \times 1\ 000\ 000 = 20$
- Costo join $r_2 = r_1 \bowtie_{Ristorante=ristorante.Codice} \sigma_2 = 20 \times 50 = 10^3$
- Costo totale $= 5 \times 10^5 + 5 \times 10^3 + 10^6 + 10^3 \approx 10^6$

7. **Esonero Teoria 3NF V1**

   Dati:

   $R(A, B, C, D, E, F, G)$ e

   $F = \{ADG \to BC, DG \to EF, EDG \to A, AE \to G\}$

   dire, motivando la risposta, se R è in 3FN e se non lo è decomporla in relazioni in 3FN esplicitando tutti i passaggi. La relazione normalizzata in 3FN è BCNF? Perché?

   Soluzione.

   Per prima cosa, occorre individuare la o le chiavi della relazione R. Ogni chiave deve contenere D, perché D non compare a destra in nessuna dipendenza funzionale (e quindi deve fare parte della chiave), ma $D^+ = \{D\}$, e quindi non è chiave; $\underline{DG^+ = \{D, G|E, F|A|B, C\}}$ è in effetti una chiave. Un'altra chiave è ADE.

   La relazione è già in 3NF, infatti tutte le dipendenze funzionali sono del tipo superchiave, tranne l'ultima che è del tipo attributi primi.

   Il risultato non è BCNF perché la relazione ha una dipendenza funzionale di tipo non ammesso dalla BCNF (attributi primi).

8. **Esonero Teoria 3NF V2**

   Dati:

   $R(A, B, C, D, E, F, G, H)$ e

   $F = \{DFG \to BE, DG \to AC, CDG \to F, FC \to D\}$

   dire, motivando la risposta, se R è in 3FN e se non lo è decomporla in relazioni in 3FN esplicitando tutti i passaggi. La relazione normalizzata in 3FN è BCNF? Perché?

   Soluzione.

   Per prima cosa, occorre individuare la o le chiavi della relazione R. Ogni chiave deve contenere GH, perché G e H non compaiono a destra in nessuna dipendenza funzionale (e quindi devono fare parte della chiave), ma $GH^+ = \{G, H\}$, e quindi non è chiave. Proviamo ad aggiungere a GH gli altri attributi di R che sono a sinistra e troviamo due chiavi candidate: $DHG^+ = \{D, H, G|C, A|F|B, E\}$ e $GHFC^+ = \{G, H, F, C|D|B, E|A\}$.

   La relazione non è in 3NF perché ad esempio la prima d.f. non è banale, non è di tipo superchiave e non è attributi primi.

   Troviamo la copertura minimale dalla forma canonica $F' = \{FDG \to B, FDG \to E, DG \to C, DG \to A, CDG \to F, FC \to D\}$.

   A F B B C E D G E C F A G D H H

F è estraneo sia in $FDG \to B$ che in $FDG \to E$, infatti da DG posso determinare C e da DGC posso determinare F. C è estraneo in $CDG \to F$ perché C si può determinare da $DG \to C$. Quindi abbiamo $F' = \{DG \to B, DG \to E, DG \to C, DG \to A, DG \to F, FC \to D\}$.

Non ci sono d.f. ridondanti e possiamo scomporre in 3NF:

R1(D,G,F,B,E,C,A)

R'(F,C,D)

Gli attributi di R' sono un sottoinsieme di quelli di R1 e accorpiamo la relazione conservando la d.f. R1(D,G,A,B,C,E,F) con $AE \to G$.

Aggiungiamo una relazione R2(D,G,H) contenente una chiave di R.

Il risultato non è BCNF perché R1 ha la dipendenza funzionale $FC \to D$ non banale e non di tipo superchiave.

9. **Esonero Teoria Armstrong V1**

   Dimostrare la seguente proprietà usando le regole di Armstrong:

   Date le dipendenze funzionali $f_1 = AB \to C$, $f_2 = D \to E$ e $f_3 = CE \to Z$, se B è un attributo estraneo in f_1 allora $AD \to Z$.

   Soluzione.

   Dato che B è estraneo in f_1, allora vale $A \to C$. Per il teorema del prodotto, se $A \to C$ e $D \to E$ allora $AD \to CE$. Infine, per transitività, dato che $AD \to CE$ e $CE \to Z$, vale $AD \to Z$ [C.V.D.]

10. **Esonero Teoria Armstrong V2**

    Dimostrare la seguente proprietà usando le regole di Armstrong: A C D E Z X Date le dipendenze funzionali $f_1 = CB \to A$, $f_2 = E \to D$ e $f_3 = AD \to X$, se B è un attributo estraneo in f_1 allora $CE \to X$.

    Soluzione.

    Dato che B è estraneo in f_1, allora vale $C \to A$. Per il teorema dell'unione, se $C \to A$ e $E \to D$ allora $CE \to AD$. Infine, per transitività, dato che $CE \to AD$ e $AD \to X$, vale $CE \to X$ [C.V.D.]

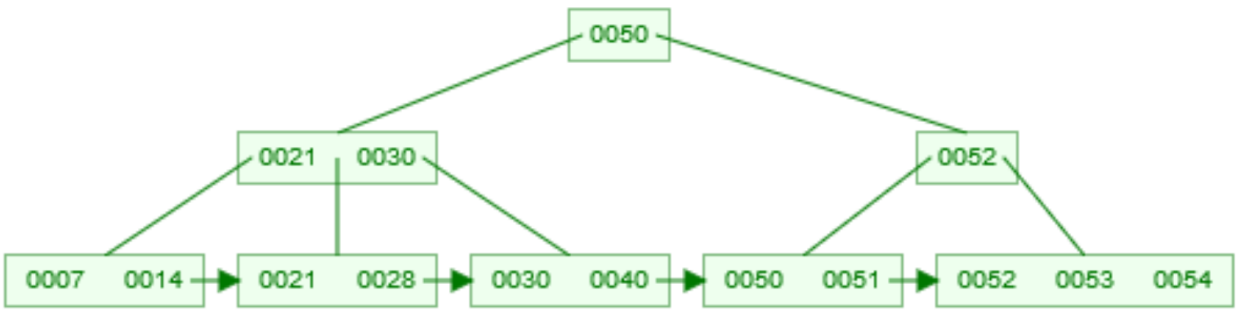
11. **Esonero Teoria B+-Tree V1**

    Rappresentare due possibili B+-tree contenenti le chiavi $(7, 14, 21, 28, 30, 40, 50, 51, 52, 53, 54)$ per i casi con m=4 e m=6. Non si richiede di simulare le singole operazioni di inserimento, ma di mostrare un possibile B+-tree con le caratteristiche indicate.

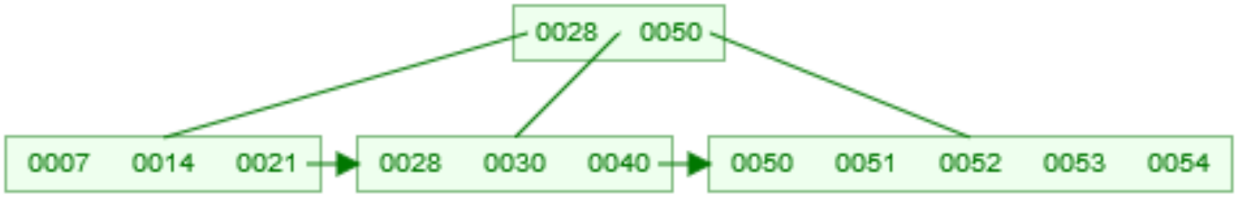
    Soluzione.

    A. Caso m=4. Ogni nodo ha al massimo $m - 1 = 3$ chiavi, la radice ha almeno 1 chiave, gli altri nodi hanno almeno $\lceil \frac{m}{2} - 1 \rceil = 1$ chiave. Le foglie devono contenere tutte le chiavi e essere linkate.

    Un possibile B+-tree è in figura.



Caso m=6. Ogni nodo ha al massimo $m-1=5$ chiavi, la radice ha almeno 1 chiave, gli altri nodi hanno almeno $\lceil \frac{m}{2}-1 \rceil = 2$ chiavi.



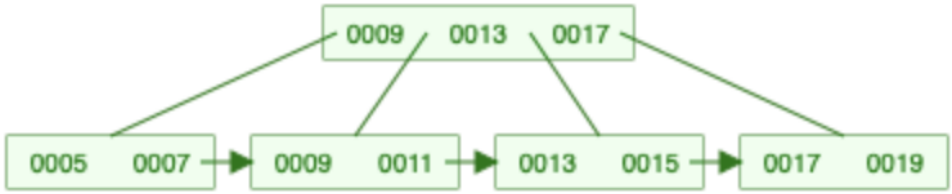
12. **Esonero Teoria B+-Tree V2**

Rappresentare due possibili B+-tree contenenti le chiavi $(5, 7, 9, 11, 13, 15, 17, 19)$ per i casi con m=4 e m=6. Non si richiede di simulare le singole operazioni di inserimento, ma di mostrare un possibile B+-tree con le caratteristiche indicate.

Soluzione.

A. Caso m=4. Ogni nodo ha al massimo $m-1=3$ chiavi, la radice ha almeno 1 chiave, gli altri nodi hanno almeno $\lceil \frac{m}{2}-1 \rceil = 1$ chiave. Le foglie devono contenere tutte le chiavi e essere linkate.

Un possibile B+-tree è in figura.



Caso m=6. Ogni nodo ha al massimo $m-1=5$ chiavi, la radice ha almeno 1 chiave, gli altri nodi hanno almeno $\lceil \frac{m}{2}-1 \rceil = 2$ chiavi.

13. **Esonero Teoria 2PL V1**

Si consideri un file di log L con il seguente contenuto in seguito a un crash:

$\langle T1, START \rangle$;

$\langle T2, START \rangle$;

$\langle T1, BS(t1[A], 15), AS(t1[A], 20) \rangle$;

$\langle T2, BS(t2[B], 13), AS(t2[B], 15) \rangle$;

$\langle T3, START \rangle$;

$\langle T1, COMMIT \rangle$;

$\langle T3, BS(t3[C], 13), AS(t3[C], 15) \rangle$;

$\langle T3, COMMIT \rangle$

crash!

Ipotizzando che, al riavvio del DBMS in seguito al crash, il contenuto dell'area primaria sia il seguente:

— A — B — C
t1 — 15 — 4 — 1
t2 — 9 — 15 — 11
t3 — 21 — 4 — 15

indicare il contenuto della nuova area primaria.

Soluzione.

Le transazioni T1 e T3 vanno in commit prima del crash. Vanno quindi rifatte e $LC = \{T1, T3\}$. La transazione T2 va in crash prima del commit (o abort), di conseguenza $LA = \{T2\}$.

La nuova area primaria sarà:

	A	B	C
t1	**20**	4	1
t2	9	**13**	11
t3	21	4	15

14. **Esonero Teoria 2PL V2**

Si consideri un file di log L con il seguente contenuto in seguito a un crash:

$\langle T1, START \rangle$;

$\langle T1, BS(t1[A], 1), AS(t1[A], 3) \rangle$;

$\langle T2, START \rangle$;

$\langle T2, BS(t2[B], 10), AS(t2[B], 2) \rangle$;

$\langle T2, ABORT \rangle$

$\langle T3, START \rangle$;

$\langle T3, BS(t3[C], 13), AS(t3[C], 17) \rangle$;

$\langle T1, COMMIT \rangle$;

$\langle T3, COMMIT \rangle$

crash!

Indicare il contenuto delle liste LA e LC per il ripristino.

Soluzione.

Le transazioni T1 e T3 vanno in commit prima del crash. Vanno quindi rifatte e $LC = \{T1, T3\}$. La transazione T2 va in abort prima del crash, quindi non va disfatta. Di conseguenza $LA = \{\emptyset\}$.