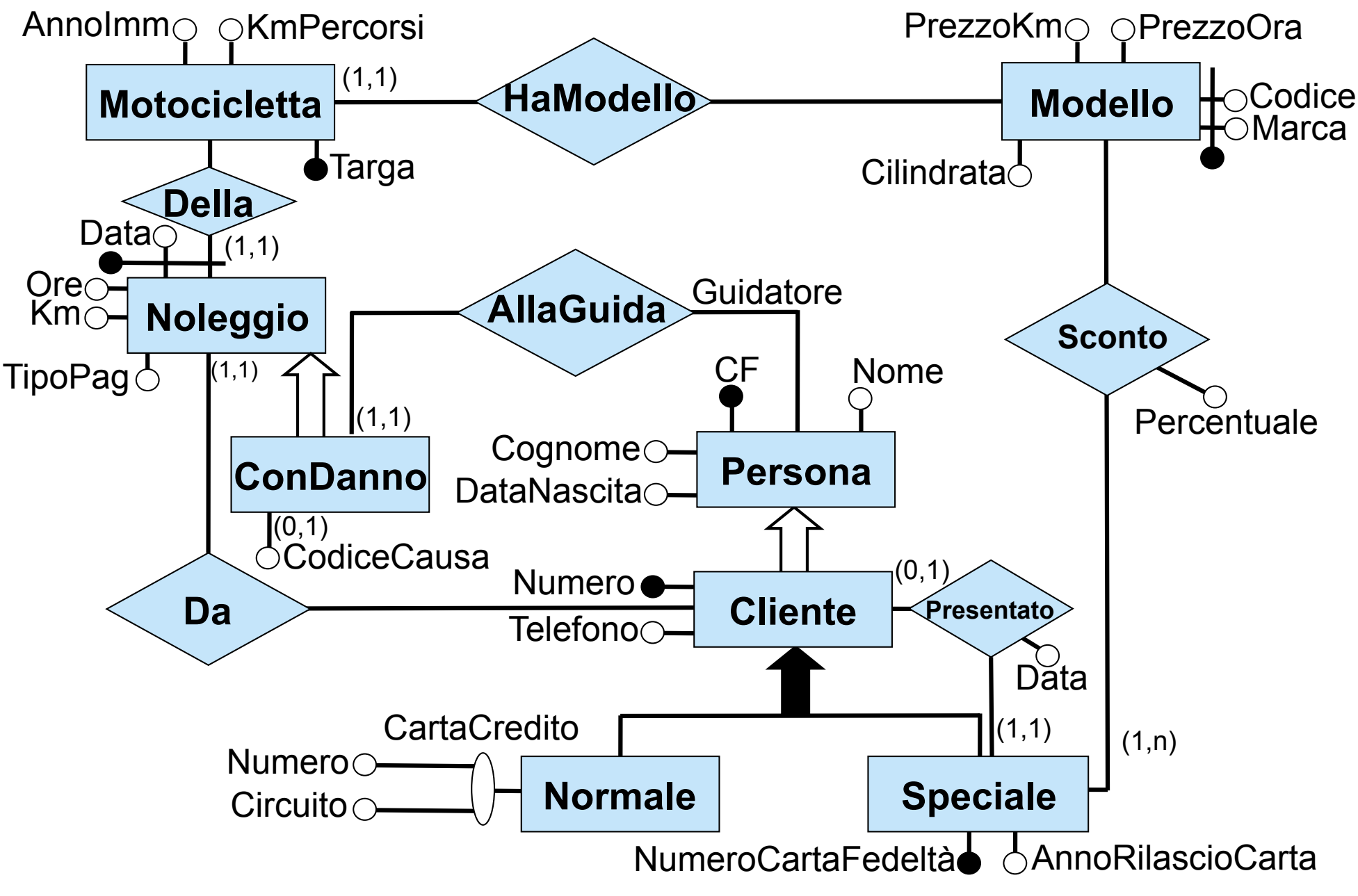
Basi di dati

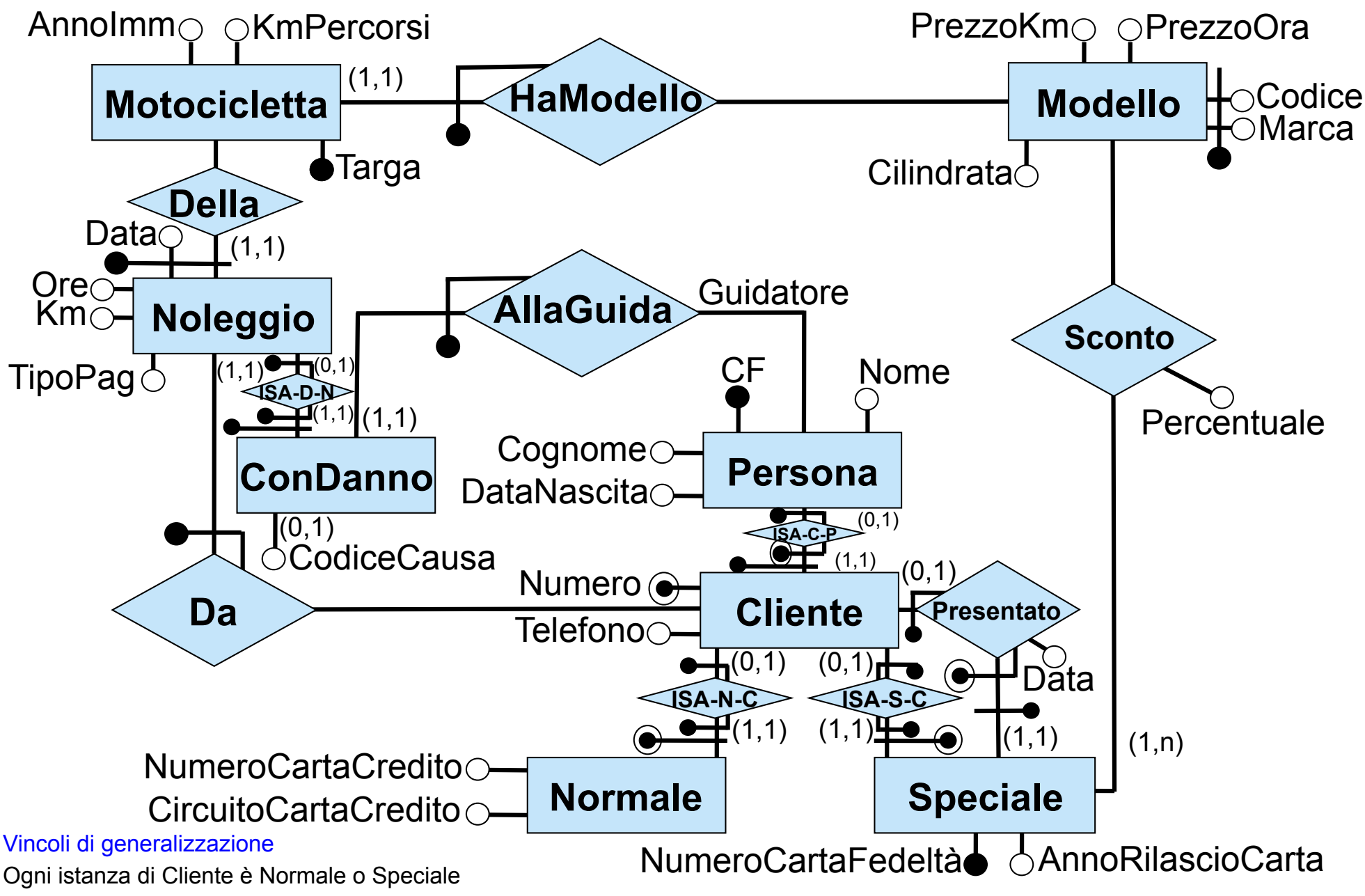
Appello del 10-01-2013
Compito A

Anno Accademico 2012/13

Problema 1 – Schema ER



Problema 2 – Schema ER ristrutturato



Vincoli di generalizzazione

Ogni istanza di Cliente è Normale o Speciale

Normale e Speciale sono entità disgiunte

Problema 2 – Schema logico dalla traduzione

Modello(<u>codice</u>, <u>marca</u>, cilindrata, prezzoKm, prezzoOra)

Motocicletta(<u>targa</u>, annoImm, kmPercorsi)

foreign key: Motocicletta[targa] ⊆ HaModello[motocicletta]

HaModello(<u>motocicletta</u>, codmodello, marcamodello)

foreign key: HaModello[motocicletta] ⊆ Motocicletta[targa]

foreign key: HaModello[codmodello,marcamodello] ⊆ Modello[codice,marca]

Noleggio(<u>motocicletta</u>, <u>data</u>, ore, km, tipopag)

foreign key: Noleggio[motocicletta,data] ⊆ Da[motocicletta,data]

Da(<u>motocicletta</u>, <u>data</u>, cliente)

foreign key: Da[motocicletta,data] ⊆ Noleggio[motocicletta,data]

foreign key: Da[cliente] ⊆ Cliente[numero]

ConDanno(<u>motocicletta</u>, <u>data</u>, codicecausa*)

foreign key: ConDanno[motocicletta,data] ⊆ Noleggio[motocicletta,data]

foreign key: ConDanno[motocicletta,data] ⊆ AllaGuida[motocicletta,data]

Cliente(<u>numero</u>, telefono)

foreign key: Cliente[numero] ⊆ ISA-C-P[cliente]

Cliente[numero] ⊆ Normale[numero] ∪ Speciale[numero]

Normale[numero] ∩ Speciale[numero] = ∅

Problema 2 – Schema logico dalla traduzione

ISA-C-P(cliente, persona)
- foreign key: ISA-C-P[cliente] ⊆ Cliente[numero]
- foreign key: ISA-C-P[persona] ⊆ Persona[CF]
- chiave: persona

AllaGuida(motocicletta, data, guidatore)
- foreign key: AllaGuida[motocicletta,danno] ⊆ ConDanno[motocicletta,data]
- foreign key: AllaGuida[guidatore] ⊆ Persona[CF]

Normale(numero, numeroCartaCredito, circuitoCartaCredito)
- foreign key: Normale[numero] ⊆ Cliente[numero]

Speciale(numero, numeroCartaFedeltà, annoRilascioCarta)
- foreign key: Speciale[numero] ⊆ Cliente[numero]
- foreign key: Speciale[numero] ⊆ Presentato[speciale]
- inclusione: Speciale[numero] ⊆ Sconto[speciale]
- chiave: numeroCartaFedeltà

Presentato(cliente, speciale, data)
- foreign key: Presentato[cliente] ⊆ Cliente[numero]
- foreign key: Presentato[speciale] ⊆ Speciale[numero]
- chiave: numero

Sconto(codmodello, marcamodello, speciale, percentuale)
- foreign key: Sconto[codmodello,marcamodello] ⊆ Modello[codice,marca]
- foreign key: Sconto[speciale] ⊆ Speciale[numero]

Problema 2 – Ristrutturazione schema logico

Il fatto che ai clienti si acceda mediante il numero identificativo è stato già considerato scegliendo "numero" come identificatore principale (poi diventato chiave primaria) di Cliente. Per rispondere alla indicazione che quando si accede ai dati relativi ai noleggi per i quali si è verificato un danno alla motocicletta e per i quali è nota la causa del danno, si vuole anche sapere chi era alla guida della motocicletta al momento del verificarsi del danno, si deve operare un accorpamento tra ConDanno e AllaGuida, ma solo per quelle tuple di ConDanno per le quali è nota la causa del danno, ovvero per le quali il valore di "codicecausa" non è nullo. È quindi necessario prima operare una decomposizione mista sulla relazione ConDanno per separare i noleggi con danno con causa nota dai noleggi con danno senza causa nota, ed una corrispondente decomposizione orizzontale di AllaGuida per separare le tuple che si riferiscono a noleggi con danno con causa nota dalle tuple che si riferiscono a noleggi con danno senza causa nota:

ConDannoSenzaCausa(moto, data)

foreign key: ConDannoSenzaCausa[motocicletta,data] ⊆ Noleggio[motocicletta,data]

foreign key: ConDannoSenzaCausa[motocicletta,data] ⊆ AllaGuidaSenzaCausa[motocicletta,data]

ConDannoConCausa(motocicletta, data, codicecausa)

foreign key: ConDannoConCausa[motocicletta,data] ⊆ Noleggio[motocicletta,data]

foreign key: ConDannoConCausa[motocicletta,data] ⊆ AllaGuidaConCausa[motocicletta,data]

ConDannoConCausa[motocicletta,data] ∩ ConDannoSenzaCausa[motocicletta,data] = ∅

AllaGuidaSenzaCausa(motocicletta, data, guidatore)

foreign key: AllaGuidaSenzaCausa[motocicletta,danno] ⊆ ConDannoSenzaCausa[motocicletta,data]

foreign key: AllaGuidaSenzaCausa[guidatore] ⊆ Persona[CF]

AllaGuidaConCausa(moto, data, guidatore)

foreign key: AllaGuidaConCausa[motocicletta,danno] ⊆ ConDannoConCausa[motocicletta,data]

foreign key: AllaGuidaConCausa[guidatore] ⊆ Persona[CF]

Problema 2 – Ristrutturazione schema logico

A questo punto possiamo operare un accorpamento tra ConDannoConCausa e AllaGuidaConCausa, eliminando la relazione AllaGuidaConCausa e trasformando la relazione ConDannoConCausa opportunamente. Le relazioni risultanti sono:

ConDannoSenzaCausa(motocicletta, data)
- foreign key: ConDannoSenzaCausa[motocicletta,data] ⊆ Noleggio[motocicletta,data]
- foreign key: ConDannoSenzaCausa[motocicletta,data] ⊆ AllaGuidaSenzaCausa[motocicletta,data]

ConDannoConCausa(motocicletta, data, codicecausa, guidatore)
- foreign key: ConDannoConCausa[motocicletta,data] ⊆ Noleggio[motocicletta,data]
- foreign key: ConDannoConCausa[guidatore] ⊆ Persona[CF]
- ConDannoConCausa[motocicletta,data] ∩ ConDannoSenzaCausa[motocicletta,data] = ∅

AllaGuidaSenzaCausa(motocicletta, data, guidatore)
- foreign key: AllaGuidaSenzaCausa[motocicletta,danno] ⊆ ConDannoSenzaCausa[motocicletta,data]
- foreign key: AllaGuidaSenzaCausa[guidatore] ⊆ Persona[CF]

Problema 2 – Schema logico finale

Modello(<u>codice</u>, <u>marca</u>, cilindrata, prezzoKm, prezzoOra)

Motocicletta(<u>targa</u>, annoImm, kmPercorsi)

foreign key: Motocicletta[targa] ⊆ HaModello[motocicletta]

HaModello(<u>motocicletta</u>, codmodello, marcamodello)

foreign key: HaModello[motocicletta] ⊆ Motocicletta[targa]

foreign key: HaModello[codmodello,marcamodello] ⊆ Modello[codice,marca]

Noleggio(<u>motocicletta</u>, <u>data</u>, ore, km, tipopag)

foreign key: Noleggio[motocicletta,data] ⊆ Da[motocicletta,data]

Da(<u>motocicletta</u>, <u>data</u>, cliente)

foreign key: Da[motocicletta,data] ⊆ Noleggio[motocicletta,data]

foreign key: Da[cliente] ⊆ Cliente[numero]

ConDannoSenzaCausa(<u>motocicletta</u>, <u>data</u>)

foreign key: ConDannoSenzaCausa[motocicletta,data] ⊆ Noleggio[motocicletta,data]

foreign key: ConDannoSenzaCausa[motocicletta,data] ⊆ AllaGuidaSenzaCausa[motocicletta,data]

ConDannoConCausa(<u>motocicletta</u>, <u>data</u>, codicecausa, guidatore)

foreign key: ConDannoConCausa[motocicletta,data] ⊆ Noleggio[motocicletta,data]

foreign key: ConDannoConCausa[guidatore] ⊆ Persona[CF]

ConDannoConCausa[motocicletta,data] ∩ ConDannoSenzaCausa[motocicletta,data] = ∅

AllaGuidaSenzaCausa(<u>motocicletta</u>, <u>data</u>, guidatore)

foreign key: AllaGuidaSenzaCausa[motocicletta,danno] ⊆ ConDannoSenzaCausa[motocicletta,data]

foreign key: AllaGuidaSenzaCausa[guidatore] ⊆ Persona[CF]

Problema 2 – Schema logico finale

Cliente(numero, telefono)

foreign key: Cliente[numero] ⊆ ISA-C-P[cliente]

Cliente[numero] ⊆ Normale[numero] ∪ Speciale[numero]

Normale[numero] ∩ Speciale[numero] = ∅

ISA-C-P(cliente, persona)

foreign key: ISA-C-P[cliente] ⊆ Cliente[numero]

foreign key: ISA-C-P[persona] ⊆ Persona[CF]

chiave: persona

Normale(numero, numeroCartaCredito, circuitoCartaCredito)

foreign key: Normale[numero] ⊆ Cliente[numero]

Speciale(numero, numeroCartaFedeltà, annoRilascioCarta)

foreign key: Speciale[numero] ⊆ Cliente[numero]

foreign key: Speciale[numero] ⊆ Presentato[speciale]

inclusione: Speciale[numero] ⊆ Sconto[speciale]

chiave: numeroCartaFedeltà

Presentato(cliente, speciale, data)

foreign key: Presentato[cliente] ⊆ Cliente[numero]

foreign key: Presentato[speciale] ⊆ Speciale[numero]

chiave: numero

Sconto(codmodello, marcamodello, speciale, percentuale)

foreign key: Sconto[codmodello,marcamodello] ⊆ Modello[codice,marca]

foreign key: Sconto[speciale] ⊆ Speciale[numero]

Problema 3

Si fa riferimento alle relazioni

Film(titolo,anno,regista,paese,spettatori)

Genere(regista,generefilm)

1. Calcolare regista e numero di spettatori dei film prodotti tra il 2000 (compreso) ed il 2010 (compreso) da Italia, Francia e Germania.

***Soluzione**: È sufficiente una banale selezione sulla relazione Film.*

```
select Film.regista, Film.spettatori
from  Film
where Film.anno >= 2000 and Film.anno <= 2010 and
      (Film.paese = 'Italia' or Film.paese = 'Francia' or
       Film.paese = 'Germania')
```

Problema 3

2. Per ogni genere, calcolare il numero complessivo di spettatori che hanno assistito a film diretti da registi specializzati in quel genere.

***Soluzione**: Si calcola il join tra Film e Genere sull'attributo regista, e poi si raggruppano (con "group by") le tuple sulla base dell'attributo generefilm. Per ogni gruppo si calcola poi la somma dei valori che compaiono nell'attributo spettatori.*

```
select Genere.generefilm, sum(Film.spettatori)
from  Film, Genere
where Film.regista = Genere.regista
group by Genere.generefilm
```

Problema 3

3. Calcolare gli anni in cui è stato prodotto almeno un film che ha avuto più di 1000 spettatori, ed in cui non sono stati prodotti film diretti da registi specializzati nel genere “horror”.

***Soluzione**: Si calcolano con una sottoquery S gli anni in cui sono stati prodotti film diretti da registi specializzati nel genere “horror”, e poi si include nel risultato della query complessiva ogni anno che in cui è stato prodotto almeno un film che ha avuto più di 1000 spettatori e che non compare nel risultato della sottoquery S.*

```
select f1.anno
from Film f1
where f1.spettatori > 1000 and
      f1.anno not in (select f2.anno
                      from Film f2, Genere g
                      where f2.regista = g.regista and
                            g.generefilm = ‘horror’)
```

Problema 4

Un vincolo di integrità è una condizione che si esprime a livello di schema della base di dati e che si intende debba essere soddisfatta da tutte le istanze della base di dati.

I vincoli presenti nello schema S sono:

1. Cardinalità minima 1 e massima 1 di E rispetto ad R1 nel ruolo E
2. Cardinalità minima 1 e massima 1 di E rispetto ad R2 nel ruolo E
3. Vincolo di identificazione per E dato dai ruoli E in R1 ed E in R2
4. Vincolo di cardinalità minima 1 e massima 1 dell'attributo V per l'entità E

Una istanza I di S che non soddisfa tutti i vincoli è, ad esempio, una qualunque in cui si ha (non importa mostrare le istanze di A,B,R1 ed R2 in I):

Istanze(I,E) = { *e* }

Istanze(I,V) = { }

Qualunque siano le istanze di A,B,R1 ed R2 in I, l'istanza I non soddisfa il vincolo di cardinalità minimo 1 dell'attributo V per l'entità E, poiché per l'istanza *e* dell'entità E non è definito alcun valore per l'attributo V.