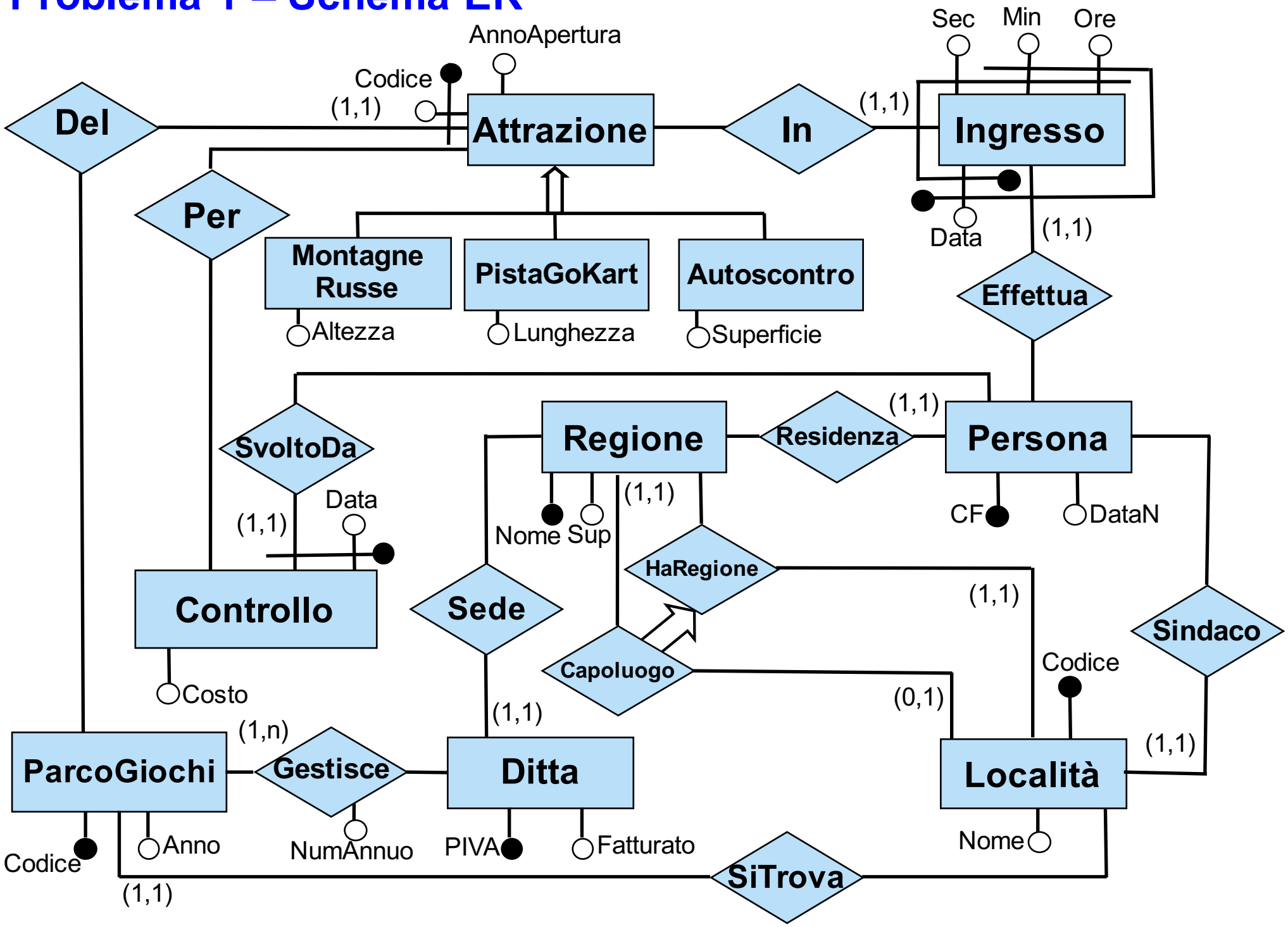
Basi di dati

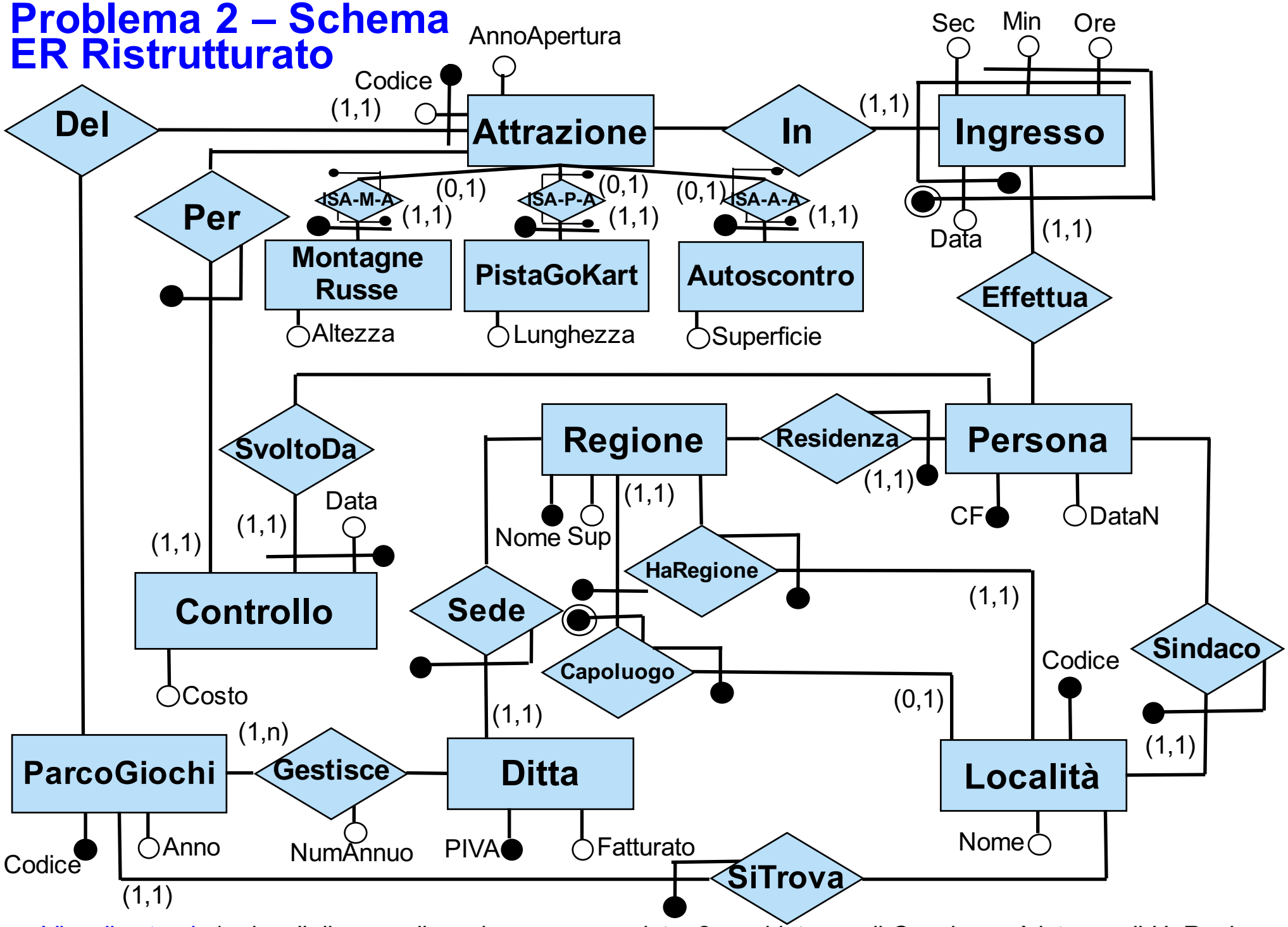
Appello del 25-01-2017
Compito B

Anno Accademico 2016/17

Problema 1 – Schema ER



Problema 2 – Schema ER Ristrutturato



Vincoli esterni: 1. vincoli di generalizzazione non completa; 2. ogni istanza di Capoluogo è istanza di HaRegione

Problema 2 - Schema logico dopo la traduzione diretta

Attrazione(Codice, ParcoGiochi, AnnoApertura)
  foreign key: Attrazione[ParcoGiochi] ⊆ ParcoGiochi[Codice]
ParcoGiochi(Codice, Anno)
  foreign key: ParcoGiochi[Codice] ⊆ Gestisce[ParcoGiochi]
  foreign key: ParcoGiochi[Codice] ⊆ SiTrova[ParcoGiochi]
MontagneRusse(Codice, ParcoGiochi, Altezza)
  foreign key: MontagneRusse[Codice,ParcoGiochi] ⊆ Attrazione[Codice,ParcoGiochi]
  vincolo di disgiuntezza: MontagneRusse[Codice,ParcoGiochi] ∩ PistaGoKart[Codice,ParcoGiochi]=∅
  vincolo di disgiuntezza: MontagneRusse[Codice,ParcoGiochi] ∩ Autoscontro[Codice,ParcoGiochi]=∅
PistaGoKart(Codice, ParcoGiochi, Lunghezza)
  foreign key: PistaGoKart[Codice,ParcoGiochi] ⊆ Attrazione[Codice,ParcoGiochi]
  vincolo di disgiuntezza: PistaGoKart[Codice,ParcoGiochi] ∩ Autoscontro[Codice,ParcoGiochi]=∅
Autoscontro(Codice, Parcogiochi, Superficie)
  foreign key: Autoscontro[Codice,ParcoGiochi] ⊆ Attrazione[Codice,ParcoGiochi]
SiTrova(Codice, Località)
  foreign key: SiTrova[Codice] ⊆ ParcoGiochi[Codice]
  foreign key: SiTrova[Località] ⊆ Località[Nome]
Località(Codice, Nome)
  foreign key: Località[Codice] ⊆ HaRegione[Località]
  foreign key: Località[Codice] ⊆ Sindaco[Località]
Sindaco(Località, Persona)
  foreign key: Sindaco[Località] ⊆ Località[Codice]
  foreign key: Sindaco[Persona] ⊆ Persona[CF]
HaRegione(Località, Regione)
  foreign key: HaRegione[Località] ⊆ Località[Codice]
  foreign key: HaRegione[Regione] ⊆ Regione[Nome]
Regione(Nome, Sup)
  foreign key: Regione[Nome] ⊆ Capoluogo[Regione]

Problema 2 - Schema logico dopo la traduzione diretta

Capoluogo(Regione, Località)
- foreign key: Capoluogo[Regione,Località] ⊆ HaRegione[Regione,Località]
- chiave: Località

Ditta(PIVA, Fatturato)
- foreign key: Azienda[PIVA] ⊆ Sede[Azienda]

Gestisce(ParcoGiochi, Ditta, NumAnnuo)
- foreign key: Gestisce[ParcoGiochi] ⊆ ParcoGiochi[Codice]
- foreign key: Gestisce[Ditta] ⊆ Ditta[PIVA]

Sede(Ditta, Regione)
- foreign key: Sede[Ditta] ⊆ Ditta[PIVA]
- foreign key: Sede[Regione] ⊆ Regione[Nome]

Persona(CF, DataN)
- foreign key: Persona[CF] ⊆ Residenza[Persona]

Residenza(Persona, Regione)
- foreign key: Residenza[Persona] ⊆ Persona[CF]
- foreign key: Residenza [Regione] ⊆ Regione[Nome]

Ingresso(Persona, Data, Ore, Min, Sec]
- foreign key: Ingresso[Persona] ⊆ Persona[CF]

In(PersIngr, DataIngr, OreIngr, MinIngr,CodAttr,ParcoAttr)
- foreign key: In[PersIngr,DataIngr,OreIngr,MinIngr] ⊆ Ingresso[Persona,Data,Ore,Min]
- foreign key: In[CodAttr,ParcoAttr] ⊆ Attrazione[Codice,ParcoGiochi]
- vincolo esterno: nella relazione che è il risultato dell'equijoin tra In e Ingresso con condizione (PersIngr=Persona and DataIngr=Data and OreIngr=Ore and MinIngr=Min), gli attributi CodAttr, ParcoAttr, Data, Ore, Min e Sec formano una chiave

Problema 2 - Schema logico dopo la ristrutturazione

- *Indicazione 1*: Accorpamento di Regione e Capoluogo (sfruttando l'accoppiamento forte)
- *Indicazione 2*: Accorpamento di Attrazione con MontagneRusse, PistaGoKart, Autoscontro, con relativa introduzione dell'attributo "Tipo" (i cui valori possibili sono esattamente quattro, 1 corrispondente a MontagneRusse, 2 a PistaGoKart e 3 a Autoscontro e 4 ad altro), e con indicazioni di possibili valori nulli negli attributi Altezza, Lunghezza e Superficie; successiva decomposizione per ricreare le relazioni MontagneRusse e PistaKoGart, riscorporandole da Attrazione, al fine di evitare valori nulli.

Attrazione(Codice, ParcoGiochi, AnnoApertura, Tipo, Superficie*)
foreign key: Attrazione [ParcoGiochi] $\subseteq$ ParcoGiochi[Codice]
vincolo di dominio: Tipo=1 OR Tipo=2 OR Tipo=3 OR Tipo=4
vincolo di tupla: Tipo=3 SE E SOLO SE Superficie<>NULL
vincolo esterno: $PROJ_{Codice,ParcoGiochi}SEL_{Tipo=1}$(Attrazione) $\subseteq$ MontagneRusse[Codice,ParcoGiochi]
vincolo esterno: $PROJ_{Codice,Parcogiochi}SEL_{Tipo=2}$(Attrazione) $\subseteq$ PistaGoKart[Codice,ParcoGiochi]

ParcoGiochi(Codice, Anno)
foreign key: ParcoGiochi[Codice] $\subseteq$ Gestisce[ParcoGiochi]

MontagneRusse(Codice, ParcoGiochi, Altezza)
foreign key: MontagneRusse[Codice,ParcoGiochi] $\subseteq$ $PROJ_{Codice,ParcoGiochi}SEL_{Tipo=1}$(Attrazione)

PistaGoKart(Codice, ParcoGiochi, Lunghezza)
foreign key: PistaGoKart[Codice,ParcoGiochi] $\subseteq$ $PROJ_{Codice,ParcoGiochi}SEL_{Tipo=2}$(Attrazione)

SiTrova(Codice, Località)
foreign key: SiTrova[Codice] $\subseteq$ ParcoGiochi[Codice]
foreign key: SiTrova[Località] $\subseteq$ Località[Nome]

Località(Codice, Nome)
foreign key: Località[Codice] $\subseteq$ HaRegione[Località]
foreign key: Località[Codice] $\subseteq$ Sindaco[Località]

Sindaco(Località, Persona)
foreign key: Sindaco[Località] $\subseteq$ Località[Nome]
foreign key: Sindaco[Persona] $\subseteq$ Persona[CF]

HaRegione(Località, Regione)
foreign key: HaRegione[Località] $\subseteq$ Località[Codice]
foreign key: HaRegione[Regione] $\subseteq$ Regione[Nome]

Problema 2 - Schema logico dopo la ristrutturazione

Regione(Nome, Sup, Capoluogo)
- foreign key: Regione[Capoluogo] ⊆ Località[Nome]
- foreign key: Regione[Nome,Capoluogo] ⊆ HaRegione[Regione,Località]
- chiave: Capoluogo

Ditta(PIVA,Fatturato)
- foreign key: Azienda[PIVA] ⊆ Sede[Azienda]

Gestisce(ParcoGiochi, Ditta, NumAnnuo)
- foreign key: Gestisce[ParcoGiochi] ⊆ ParcoGiochi[Codice]
- foreign key: Gestisce[Ditta] ⊆ Ditta[PIVA]

Sede(Ditta, Regione)
- foreign key: Sede[Ditta] ⊆ Ditta[PIVA]
- foreign key: Sede[Regione] ⊆ Regione[Nome]

Persona(CF, DataN)
- foreign key: Persona[CF] ⊆ Residenza[Persona]

Residenza(Persona, Regione)
- foreign key: Residenza[Persona] ⊆ Persona[CF]
- foreign key: Residenza [Regione] ⊆ Regione[Nome]

Ingresso(Persona, Data, Ore, Min, Sec]
- foreign key: Ingresso[Persona] ⊆ Persona[CF]

In(PersIngr, DataIngr, OreIngr, MinIngr,CodAttr,ParcoAttr)
- foreign key: In[PersIngr,DataIngr,OreIngr,MinIngr] ⊆ Passaggio[Persona,Data,Ore,Min]
- foreign key: In[CodAttr,ParcoAttr] ⊆ Attrazione[Codice,ParcoGiochi]
- vincolo esterno: nella relazione che è il risultato dell'equijoin tra In e Ingresso con condizione (PersIngr=Persona and DataIngr=Data and OreIngr=Ore and MinIngr=Min), gli attributi CodAttr, ParcoAttr, Data, Ore, Min e Sec formano una chiave

Problema 3: soluzione

1. La risposta è positiva: esiste una base di dati T1 tale che la valutazione della query 1 su T1 dà un risultato non vuoto uguale alla valutazione della query 2 su T1. Infatti, basta considerare come T1 la base di dati che ha le seguenti tuple:

   <a1,b1> in R, <b1,c1> in P, <d1,e1> in Q.

   È facile verificare che la valutazione di entrambe le query sulla base di dati T1 dà come risultato: { <a1,c1> }.

2. Se ci focalizziamo su basi di dati in cui Q non è vuota, allora la risposta è negativa, come si può verificare analizzando cosa fanno le due query. La query 2 calcola il join tra R e la relazione ottenuta da P eliminando quelle tuple in cui il valore di C compare nell'attributo D di Q. La query 2 può essere descritta così: la query calcola il prodotto cartesiano tra R, P e Q, poi seleziona le tuple che soddisfano la clausola where e poi fa la proiezione su A e C. In altre parole, del prodotto cartesiano iniziale rimangono quelle tuple che contengono il join tra R e le tuple di P ottenute eliminando quelle tuple in cui il valore di C compare nell'attributo D di Q, e di tale tuple si calcola la proiezione. Relativamente ad una base di dati in cui Q non è vuota, le due query calcolano lo stesso risultato.

   Nota: se consideriamo anche basi di dati con la relazione Q vuota, la risposta è positiva (infatti la query 1 può calcolare un risultato non vuoto, mentre la query 2 fornisce un risultato vuoto). Chi non ha tenuto conto di questa possibilità non sarà comunque penalizzato.

Problema 4: soluzione

Si omettono le definizioni di vincolo di integrità e di vincolo di cardinalità.

Esiste una istanza I dello schema S in cui l'insieme delle istanze della entità E non è vuoto. Costruiamo I in questo modo: mettiamo e1 in Istanze(I,E), ed introduciamo la tupla <E:e1,F;f1> in Istanze(I,R), ponendo f in Istanze(I,F). A questo punto f1, essendo istanza di F, deve partecipare 3 volte a R, e quindi introduciamo anche le tuple <E:e2,F:f1> e <E:e3,F:f1> in Istanze(I,R). Siccome poi ogni istanza di F deve partecipare 2 volte a Q, inseriamo le tuple <E:e2,F:f1> e <E:e3,F:f1> anche in Istanze(I,Q). Si verifica facilmente che l'I risultante soddisfa tutti i vincoli dello schema concettuale e quindi è una istanza di S in cui l'insieme delle istanze di E non è vuoto.