UNIVERSITÀ DEGLI STUDI DI PADOVA

Corso di Laurea in Informatica
Esame di Basi di Dati

Seconda Prova dell'esame on-line di Maggio (data 16 Maggio)

Progettazione Concettuale (8 punti) e Logica (7 punti)

Si richiede di produrre i seguenti artefatti:

- ***Schema Concettuale Entità-Relazione,*** da inviare per email da account UniPd)
- ***Schema Logico****,* da inserire nel riquadro di cui sotto. *Indicare la chiave primaria, i vincoli di chiave esterne e gli attributi che ammettono valori nulli.*

per un'applicazione relativa ad una ***catena di piste di pattinaggio****. Nella progettazione concettuale e logica, occorre evitare di introdurre entità e tabelle non necessarie ed occorre minimizzare i valori nulli.*

Illustrare come ristrutturare il diagramma ER per essere direttamente traducibile in uno schema relazionale. ***Il diagramma ER ristrutturato deve anche essere inviato per email da account UniPd****.*

Il sistema deve memorizzare le informazioni di ogni pista della catena. Di ogni pista, è di interesse sapere (1) la città in cui si trova, (2) l'anno di apertura e (3) il fatturato. Ogni città non può avere più di una pista. Inoltre, ogni pista è gestita da una persona, però una pista potrebbe temporaneamente non avere una persona che la gestisce. Si noti che una persona non può prendere in gestione più di una pista della catena.

Il sistema memorizza un'anagrafe delle persone: di ogni persona, è di interesse sapere il codice fiscale (identificativo), la data di nascita e la città di nascita.

Quando le persone desiderano essere clienti di una pista, noleggiano un paio di pattini. Ogni paio di pattini è associato ad esattamente una pista.

Di ogni paio di pattini interessa il codice (unico all'interno della pista), il modello, la taglia. Inoltre, per ogni paio di pattini, è di interesse sapere quali aziende sono in grado di farne manutenzione. Si vuole anche memorizzare lo storico delle

manutenzioni: per ogni paio di pattini, è di interesse sapere quante volte (intero) il paio è andato in manutenzione in ogni azienda in grado di manutenere. Di ogni azienda di riparazione pattini interessa il codice identificativo e il numero di dipendenti.

Come detto, ogni paio di pattini è noleggiato da persone per usarle nella pista associata. In particolare, si vuole sapere per ogni noleggio: (1) quale paio è stato preso, (2) il giorno e l'ora del noleggio e (3) il costo complessivo del noleggio. Si noti che ogni paio di pattini può essere noleggiato più volte dalla stessa persona, ma solo in giorni diversi.

Alcuni persone sono clienti abbonati. Per tali clienti, è di interesse sapere il costo dell'abbonamento annuale (unico per cliente) e quale sconto (in percentuale) ottengono nei noleggi in virtù del loro abbonamento.

(8 punti per la modellazione concettuale + 7 punti per lo schema logica)

Notazione per lo schema logico:

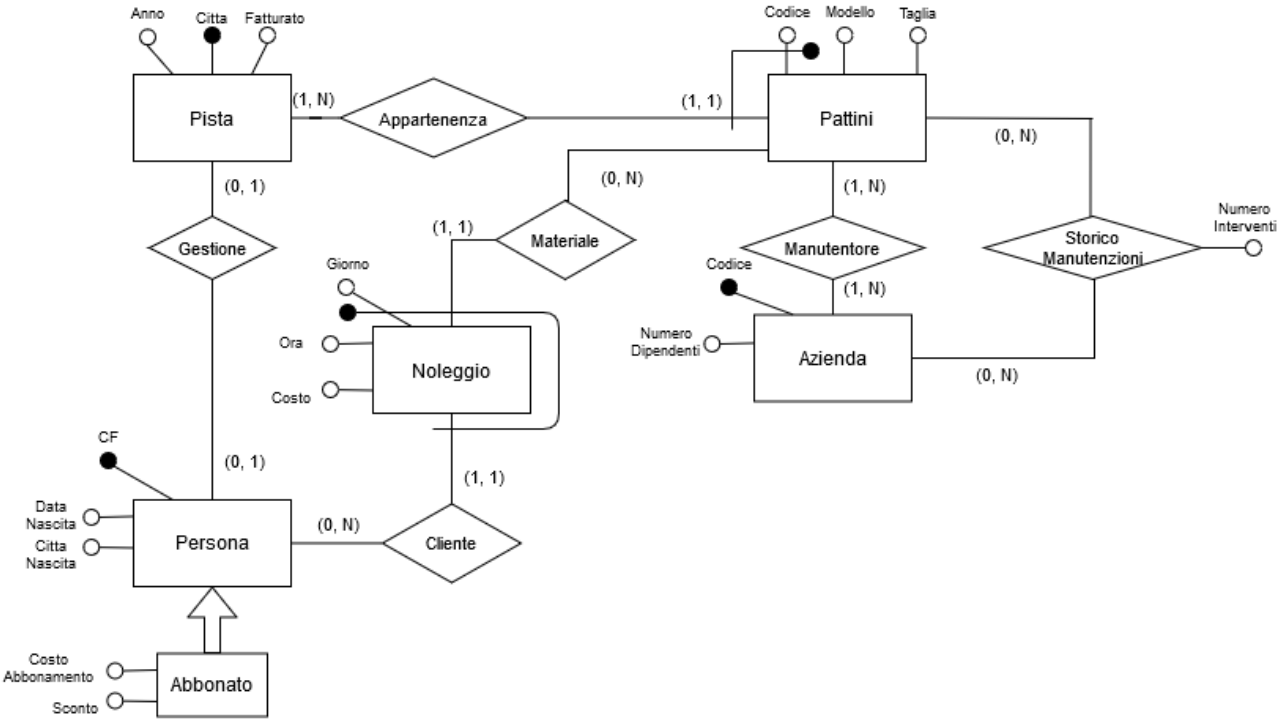
Usare il sottolineato per indicare i vincoli di chiave primaria, e il **bold** per indicare gli attributi che ammettono valori nulli.

Indicare con X.A-->Y.B per indicare che l'attributo A della tabella X è chiave esterna all'attributo B della tabella Y.

Indicare con X.(A,B) --> Y.(C,D) per indicare che gli attributi (A,B) della tabella X sono chiavi esterna all'attributo (C,D) della tabella Y.

Soluzione Progettazione Concettuale

Una soluzione è data dal seguente schema ER:

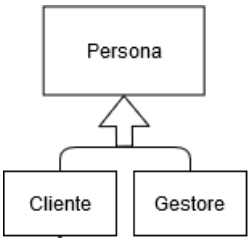


Di seguito sono alcuni punti della soluzione che meritano un approfondimento:

- L'identificatore dell'entità *Pista* è la città, poiché non è possibile avere due piste nella stessa città. L'aggiunta di un codice come identificatore è errato perché non è previsto dalle specifiche, né necessario. Inoltre, tale codice come identificatore non permettere di rappresentare il vincolo che non è possibile avere due istanze dell'entità *Pista* con lo stesso valore per l'attributo Città.
- Il codice non è sufficiente come identificatore dell'entità *Pattini* perché è unico per i pattini di una pista, ma non unico per tutti i pattini della catena.
- Si noti l'identificatore dell'entità *Noleggio*. L'identificatore modella che non è possibile che la stessa persona noleggi gli stessi pattini nello stesso giorno. Si osservi che l'ora non è parte dell'identificatore, altrimenti sarebbe possibile noleggiare i pattini ad orari diversi dello stesso giorno.
- **È errato realizzare *Noleggio* come relazione tra *Persona* e *Pattini* invece di entità come proposto sopra**. Se *Noleggio* fosse una relazione, non sarebbe possibile noleggiare gli stessi pattini più di una volta dalla parte della stessa persona: il testo indica che tale eventualità è possibile ma in giorni diversi.
- Si è considerato accettabile omettere la relazione *Manutentore*, assumendo che numero di interventi è 0 quando una azienda è registrata per la manutenzione, ma non ha effettuato ancora interventi. In tal caso, comunque,

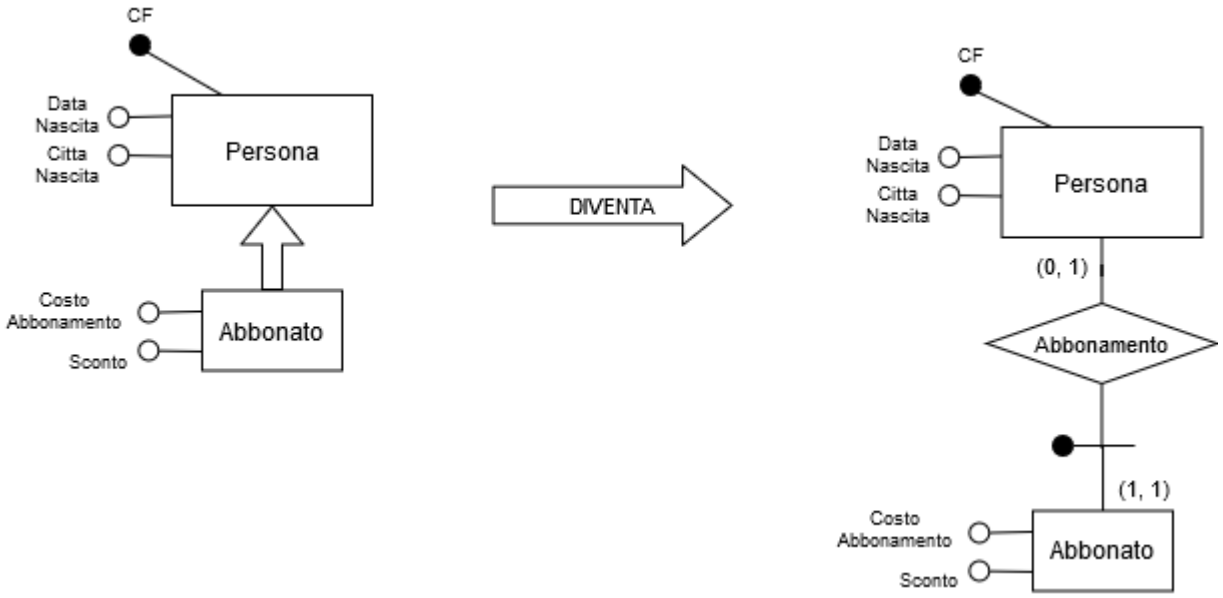
la cardinalità della relazione *Storico Manutentori* deve essere (1,N) dalla parte dell'azienda: ogni azienda è inserita se e solo se c'è almeno un paio di pattini che può essere manutenuta da essa.

L'ultima osservazione riguarda la possibilità qui a fianco (con *Persona* che generalizza le entità *Cliente* e *Gestore*). Tale soluzione è errata a meno che viene esplicitamente detto che la generalizzazione persona-cliente-gestore è sovrapposta. Se non viene detto che è *sovrapposta* viene considerata il default, che è *esclusiva*: vorrebbe dire che una persona può essere solamente cliente oppure gestore. Si noti che il fatto che la generalizzazione sia totale (freccia piena) o parziale (freccia vuota) non cambia nulla.



Soluzione "Progettazione Logica"

Il primo passaggio è la ristrutturazione dello ER per rimuovere la generalizzazione:



Si è deciso in questo caso di sostituire la generalizzazione con la relazione *Abbonamento*. La soluzione alternativa sarebbe stata quella di inglobale l'entità *Abbonato* all'interno di *Persona*. Tuttavia, per tutti coloro che non sono abbonati, gli attributi *Costo Abbonamento* e *Sconto* sarebbero dovuti essere nulli. Come indicato nel testo, si vuole minimizzare il numero di valori nulli.

Alla luce di quanto sopra, lo schema logico è il seguente:

PERSONA(CF, DataNascita, CittaNascita)

PISTA(Citta, Fatturato, Anno, **Gestore**)

- Chiave Esterna: PISTA.Gestore→Persona.CF
- Chiave Aggiuntiva: Gestore[1]

PATTINI(Codice, CittaPista, Modello, Taglia)

- Chiave Esterna: PATTINI.CittaPista→PISTA.Citta

NOLEGGIO(CodicePattini, CittaPista, Persona, Giorno, Ora, Costo)

- Chiave Esterna: NOLEGGIO.(CodicePattini, CittaPista)→PATTINI.(Codice,CittaPista)
- Chiave Esterna: NOLEGGIO.Persona→Persona.CF

AZIENDA(Codice, NumeroDipendenti)

MANUTENTORE(CodicePattini, CittaPista, CodiceAzienda)

- Chiave Esterna: MANUTENTORE.(CodicePattini, CittaPista)→PATTINI.(Codice,CittaPista)
- Chiave Esterna: MANUTENTORE.CodiceAzienda→AZIENDA.Codice

STORICO(CodicePattini, CittaPista, CodiceAzienda, NumeroInterventi)

- Chiave Esterna:[2] STORICO.(CodicePattini, CittaPista)→PATTINI.(Codice,CittaPista)
- Chiave Esterna:[2] STORICO.CodiceAzienda→AZIENDA.Codice

ABBONATO(CF, Costo, Sconto)

- Chiave Esterna: ABBONATO.CF→PERSONA.CF

Ad essere rigorosi nel minimizzare i valori nulli, occorre:

- avere la relazione *PISTA(Citta, Fatturato, Anno)*,
- introdurre una relazione *GESTIONE(CittaPista,Gestore)* dove GESTIONE.CittaPista→PISTA.Citta e GESTIONE.Gestore→PERSONA.CF

Questo garantisce assenza di NULL: una tupla (citPist,CF) esiste in GESTIONE se la pista nella città citPist è gestita dalla persona con codice fiscale CF. Similmente alla prima soluzione occorre aggiungere che *Gestore* è anche chiave (non primaria).

[1] Questa chiave aggiuntiva è necessaria per assicurare che ci sia al più un solo gestore. Tuttavia non era richiesta aggiungere altra chiave oltre a quella primaria. Quindi, se non è stata inserita non è considerato errore.
[2] Volendo arrivare ad una soluzione ancora migliore, si può osservare che le tuple (pattini,pista,azienda) in storico sono un sottoinsieme di quelli in manutentore:
STORICO.(CodicePattini, CittaPista, CodiceAzienda) → MANUTENTORE.(CodicePattini, CittaPista, CodiceAzienda)

Assegnamento del Punteggio

Parte del Dominio di Modellazione	Modellazione Concettuale				Modellazione Logica (tabelle)			
	Punti per la modellazione	Errori Commessi			Punti per la modellazione	Errori Commessi		
		1	2	3+		1	2	3+
Modellazione delle Piste e del Gestore *(Entità Pista e Relazione Gestione)*	1 punto	-0.5 punti	-1 punto	-1 punto	0.5 punti	-0.25 punti	-0.5 punti	-0.5 punti
Modellazione dei Pattini e della loro Appartenenza a Piste *(Pattini e Appartenenza)*	1.5 punti	-0.5 punti	-1 punto	-1.5 punto	1.5 punti	-0.5 punti	-1 punto	-1.5 punti
Modellazione dei Noleggi *(Noleggio, Materiale, Cliente)* ***(L'errore di considerare "Noleggio" come relazione corrisponde di fatto a molteplici errori con una penalizzazione totali di punti: 1.5)***	1.5 punti	-0.75 punti	-1.25 punti	-1.5 punti	1.5 punti	-0.5 punti	-1 punto	-1.5 punti
Modellazione delle Manutenzioni *(Manutentore, Storico e Azienda)*	2 punti	-0.75 punti	-1.5 punti	-2 punti	1.25 punti	-0.5 punti	-1 punto	-1.25 punti
Modellazione della Persona e dell'Abbonato ***(L'errore sulla generalizzazione nello ER concettuale è grave e corrisponde a una penalizzazione totali di punti: 2)***	2 punti	-0.75 punti	-1.5 punti	-2 punti	1.25 punti	-0.5 punti	-1 punto	-1.25 punti
Conversione allo ER "ristrutturato" ***(L'errore nella conversione della generalizzazione è grave e corrisponde a una penalizzazione totali di punti: 1)***	Non applicabile				1	0.5	0	0

Esercizio Transazioni (5 punti)

Data lo schedule S = r1(x)w2(x)r3(x)w1(u)w3(v)r3(y)r2(y)w3(u)w4(t)w3(t) sapendo che S è conflict-serializzabile:

1. Mostrare/Spiegare se S è (o non è) view-serializzabile;
2. se S view-serializzabile, mostrare uno schedule seriale T che è view-equivalente a S, mostrando perché S and T sono view-equivalenti.

Soluzione

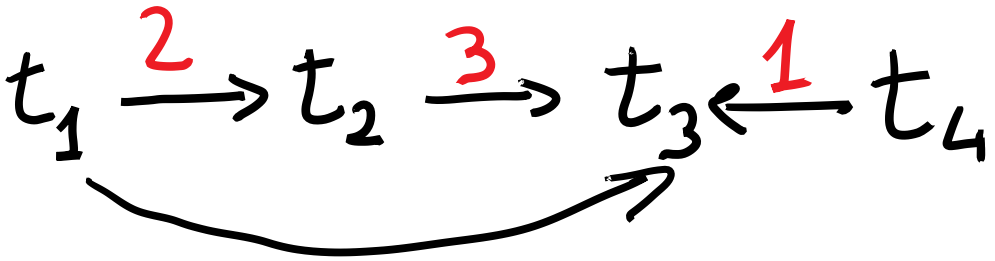
Parte 1

Siccome S è conflict-serializzabile, è anche view-serializzabile.

Parte 2

Se S è conflict-serializzabile, è possibile costruire un grafo dei conflitti di S che è aciclico. Il percorso nel grafo che tocca tutte le transazioni, fornisce un ordine delle transazioni. Scrivendo le operazioni nell'ordine delle transazioni e rispettando l'ordine all'interno delle transazioni, si ottiene uno schedule seriale che è conflict-equivalente e quindi anche view-equivalente.

Il seguente è il grafo dei conflitti:



Quindi, è possibile serializzare le transizioni nel seguente ordine: t1, t2, t4, t3.[3] Questo corrisponde allo schedule seriale T = r1(x) w1(u) w2(x) r2(z) w4(t) r3(x) w3(v) r3(z) w3(u) w3(t), che è conflict-equivalente e quindi anche view-equivalente.

Si poteva ovviamente anche lavorare direttamente sulla view-equivalenza, guardando alla relazione leggi-da e alle scritture finali. In S, *r3(x) legge da w2(x)*, mentre r1(x), r2(x) e r3(z) non leggono da scritture di transazioni in S. Le scritture finali sono w2(x), w3(t), w3(u) e w3(v). È facile verificare che lo schedule T ha una sola leggi-da come S, cioè r3(x) legge da w2(x), e le stesse scritture finali.

[3] In realtà, va bene qualsiasi ordine in cui appaia la sequenza <t1, t2, t3> e t4 sia prima di t3.
Dunque anche: <t4, t1, t2, t3> oppure <t1,t4,t2,t3>.

Assegnamento di punteggio

- **1 punto:** la parte 1 (CSR implica VSR), che è banale;

Per la parte 2 da aggiungere alla parte 1:

- **2.5 punti:** Una soluzione corretta con una spiegazione corretta ma vaga (per es. “T ha la stessa relazione leggi-da e le stesse scritture finali” senza indicare quali)
- **5 punti:** Una spiegazione dettagliata (per es. spiegando rispetto al grafo dei conflitti o mostrando la relazione leggi-da ed