Basi di Dati I, 22 dicembre 2021 - II intercorso

Adriano Peron - Silvio Barra

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: adrperon@unina.it

Si consideri il seguente schema relazionale che descrive automi costituiti da stati e transizioni. Gli stati e le transizioni sono etichettati da stringhe di caratteri. Uno stato può essere iniziale o finale (gli attributi iniziale e finale sono parziali). La relazione $RAGGIUNGIBILE$ mantiene traccia della raggiungibilità tra stati di un automa. Una traccia è un cammino tra due stati (da Start a End) tale che concatenando le etichette delle transizioni separandole con un trattino si ottiene la stringa $Parola$. Ad esempio, se il cammino nell'automa A è $s_1 \xrightarrow{aa} s_2 \xrightarrow{bb} s_3 \xrightarrow{cc} s_4$ si ha in $RAGGIUNGIBILE$ una riga $(A, s_1, "aa-bb-cc", s_4)$. Una parola P si dice riconosciuta dall'automa A se esiste in $RAGGIUNGIBILE$ una riga (A, s, P, s') dove s è uno stato iniziale di A s' è uno stato finale di A.

$AUTOMA(\underline{CodA}, Descrizione)$
$STATO(\underline{CodS}, Etichetta, CodA, Iniziale, Finale)$
$TRANSIZIONE(\underline{CodT}, Etichetta, StatoIn, StatoOut, CodA)$
$RAGGIUNGIBILE(CodA, Start, Parola, End)$.

Esercizio 01 *(Punti 15) Si scriva una procedura che riceve in ingresso il codice di un automa. La procedura deve ricalcolare tutti gli stati raggiungibili dell'automa. La procedura inizia eliminando da RAGGIUNGIBILE tutte le righe per l'automa. In modo iterativo inizia l'aggiunta degli stati raggiungibili a partire dagli stati iniziali dell'automa. Si supponga che vi possa essere più di uno stato iniziale per ogni automa. Gli stati raggiungibili in un passo sono quelli per cui esiste una transizione da uno stato iniziale (Start) a quello stato (End) e l'etichetta della parola è l'etichetta della transizione. Iterativamente si estende il contenuto di RAGGIUNGIBILE partendo da una riga (CodA,Start,Parola,End) considerando le transizioni che partono da End. La parola del nuovo stato raggiungibile si ottiene concatenando Parola con l'etichetta della transizione. Si osservi che la lunghezza della parola tra Start ed End coincide con il numero di caratteri separatori in Parola + 1. Uno stato raggiungibile non deve comparire più di una volta nella tabella.*

Esercizio 02 *(Punti 15) Si scriva una funzione che riceve in ingresso una stringa di caratteri (una sequenza di parolale separate da ;). Utilizzando SQL Dinamico la funzione restituisce gli automi che riconoscono tutte le parole nella stringa di input. Per lo svolgimento dell'esercizio si utilizzi la tabella RAGGIUNGIBILE. I codici degli automi sono restituiti in una stringa separati da ;.*

Basi di Dati I, Esercitazione II

Silvio Barra

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: silvio.barra@unina.it

Si consideri il seguente schema relazionale che descrive una collezione di alberi costruiti su un insieme comune di nodi:
$ALBERI(\underline{CodA}, Radice)$
$NODI(\underline{CodN}, Peso)$
$COMPNODI(CodA, CodN)$
$COMPARCHI(CodA, Padre, Figlio, Peso)$

$ALBERI$ descrive la collezione di alberi presenti ($Radice$ è un codice di nodo);
$NODI(\underline{CodN}, Peso)$, descrive la collezione di nodi condivisi su cui tutti gli alberi sono definiti ($Peso$ è un intero);
$COMPNODI$ descrive l'insieme dei nodi associati ad un albero;
$COMPARCHI$ descrive l'insieme degli archi associati ad un albero (a ciascun arco è associato un peso espresso da un valore intero. (Una foglia di un albero NON è $Padre$ in nessun arco nell'albero).

Esercizio 01 *Si scriva una interrogazione in algebra relazionale che fornisca se valutata il codice degli alberi in cui ogni nodo (tranne le foglie) ha esattamente due discendenti.*

Esercizio 02 *Si scriva una interrogazione in SQL che fornisca coppie di codici di alberi tali che il secondo elemento della coppia è un sottoalbero del primo elemento della coppia (ogni nodo e ogni arco del sottoalbero deve essere nodo e arco, rispettivamente, del primo albero).*

Esercizio 03 *Si esprima nel modo più adeguato il seguente insieme di vincoli:*

- *Ogni nodo di un albero ha al più un padre;*
- *La radice di un albero non ha padre;*
- *I nodi padre e figlio di un arco associato ad una albero devono essere nodi dell'albero;*
- *Quando viene rimosso un nodo da un grafo (cancellazione dalla tabella $COMPNODI$) devono essere rimossi tutti gli archi e tutti i nodi del sottoalbero radicato nel nodo (discendenti);*

Basi di Dati e Sistemi Informativi I, 18-10-22

Silvio Barra, Mara Sangiovanni

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: silvio.barra@unina.it

Si consideri il seguente schema relazionale:

IMPIEGATO(*Matricola*, *Cognome*, *Stipendio*, **Dipartimento**)
DIPARTIMENTO(*Codice*, *Nome*, *Sede*, **Manager**)
PROGETTO(*Sigla*, *Nome*, *Bilancio*, **Responsabile**)
PARTECIPAZIONE(**Progetto**, **Impiegato**)

Esercizio 01 *Algebra Relazionale e Query SQL per le seguenti interrogazioni.*

1. Trovare matricola e cognome degli impiegati che guadagnano piu' di 50 milioni.
2. Trovare cognome e stipendio degli impiegati che lavorano a Roma.
3. Trovare cognome degli impiegati e nome del dipartimento in cui lavorano.
4. Trovare cognome degli impiegati che sono direttori di dipartimento.
5. Trovare i nomi dei progetti e i cognomi dei responsabili.
6. Trovare i nomi dei progetti con bilancio maggiore di 100K e i cognomi degli impiegati che lavorano su di essi.
7. Trovare il cognome degli impiegati che guadagnano piu' del loro direttore di dipartimento.
8. Trovare cognome dei direttori di dipartimento e dei responsabili di progetto.
9. Trovare nomi dei dipartimenti in cui lavorano impiegati che guadagnano piu' di 60K.
10. Trovare nomi dei dipartimenti in cui tutti gli impiegati guadagnano piu' di 60K.
11. Trovare cognome degli impiegati di stipendio massimo.
12. Trovare matricola e cognome degli impiegati che non lavorano a nessun progetto.
13. Trovare matricola e cognome degli impiegati che lavorano a piu' di un progetto.
14. Trovare matricola e cognome degli impiegati che lavorano a un solo progetto.
15. Trovare il cognome dei direttori che lavorano nello stesso dipartimento di cui sono direttori.

Basi di Dati e Sistemi Informativi I, 22 dicembre 2022 - PROVA A

Silvio Barra, Mara Sangiovanni

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: silvio.barra@unina.it, mara.sangiovanni@unina.it

Si consideri il seguente schema relazionale che descrive un tabellone nel corso di una partita ($codP$ è il codice della partita). Una partita ha una sequenza di mosse ($OrdMossa$ indica l'ordine della mossa). La tabella TAB contiene per ogni partita e mossa della partita una riga per ogni casella del tabellone (individuata dalla coppia X e Y di coordinate orizzontale e verticale). Tale riga indica il contenuto della casella ($Pezzo$) ad una specifica mossa della partita (se la casella è vuota allora $Pezzo$ è NULL). Quindi, per ogni partita e mossa si ha una descrizione completa del contenuto del tabellone. Nella tabella $MOSSA$ viene indicato lo spostamento di un pezzo da una casella ad un'altra casella. La tabella $ELIMINATO$ contengono i pezzi eliminati nelle varie mosse. Nella tabella $PEZZI$ viene memorizzata l'informazione sull'assegnazione dei pezzi ai giocatori nelle partite.

$PEZZI(codP, Giocatore, Pezzo)$
$TAB(CodP, OrdMossa, Pezzo, X, Y)$
$MOSSA(\underline{CodP, OrdMossa}, Giocatore, daX, daY, aX, aY, Pezzo)$
$ELIMINATO(CodP, OrdMossa, Pezzo)$

Esercizio 01 *(Punti 10) Si scriva un trigger che viene attivato all'inserimento di una mossa in una partita. L'effetto del trigger è il seguente:*

1. *Viene eliminato il pezzo che si trova (alla mossa precedente) eventualmente nella casella di destinazione della mossa;*

2. *Vengono inserite righe per ogni casella del tabellone (le stesse caselle presenti alla mossa precedente) per mantenere la situazione corrente del tabellone (le caselle che non sono sorgente e destinazione della mossa mantengono la stessa situazione e il pezzo interessato dalla mossa si sposta dalla casella sorgente a quella della destinazione).*

Esercizio 02 *(Punti 8) Si scriva una funzione SQL che riceve in ingresso un codice di partita ed un pezzo. La funzione restituisce una stringa di caratteri contenente il cammino fatto dal pezzo nel tabellone nel corso della partita. Il cammino viene espresso come sequenza delle coordinate X,Y delle caselle separate da virgole.*
Esempio Stringa in output: X1,Y1;X2,Y2;.....

Esercizio 03 *(Punti 15) Si scriva una funzione SQL che riceve in ingresso una lista di codici di partite separate dal carattere separatore '@' e utilizzando SQL DINAMICO restituisce una lista di pezzi che sono stati eliminati in* ***tutte*** *le partite passate per parametro.*

[redacted]

Basi di Dati I, 19 Gennaio 2023

Mara Sangiovanni, Silvio Barra

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: mara.sangiovanni@unina.it, silvio.barra@unina.it

PRIMA PARTE: Esercizi 1, 2, 3, 4 e 5
SECONDA PARTE: Esercizi 6, 7, e 8
APPELLO: Tutti

Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire i prestiti in una biblioteca. Di un *LIBRO* (descrizione del libro) possono esserci più copie fisiche, memorizzate nella tabella *ESEMPLARE*. L'attributo booleano *Prestito* indica se l'esemplare è disponibile per il prestito. L'attributo booleano *Consultazione* indica se l'esemplare è disponibile per la consultazione in loco. Il prestito riguarda esclusivamente le copie fisiche; le prenotazioni sono registrate nella tabella *PRENOTAZIONE*: ogni prenotazione è relativa ad un libro (non alla sua copia fisica). Quando un esemplare del libro è disponibile si può effettuare il prestito. Ogni *UTENTE* ha un *PROFILO* che regola la durata dei prestiti e il massimo numero di esemplari che possono essere richiesti in prestito. I prestiti sono registrati nella tabella *PRESTITO*: ogni prestito ha una data di effettuazione, una data di scadenza, una data di restituzione e una data di sollecito (rispettivamente *DataE*, *DataS* , *DataR* e *DataSol*). La data di restituzione e la data di sollecito sono settate a *NULL* se il libro non è stato ancora restituito e se il prestito non è ancora scaduto, rispettivamente.

LIBRO(*ISBN*, *Titolo*, *Editore*, *Anno*, *Descrizione*)
ESEMPLARE(*ISBN*, *CodiceBarre*, *Collocazione*, *Prestito*, *Consultazione*)
UTENTE(*CF*, *CodProfilo*, *Nome*, *Cognome*, *DataN*)
PROFILO(*CodProfilo*, *MaxDurata*, *MaxPrestito*)
PRESTITO(*CodPrestito*, *CodiceBarre*, *Utente*, *DataE*, *DataS*, *DataR*, *Sollecito*)
PRENOTAZIONE(*CodPrenotazione*, *ISBN*, *Utente*, *Data*)

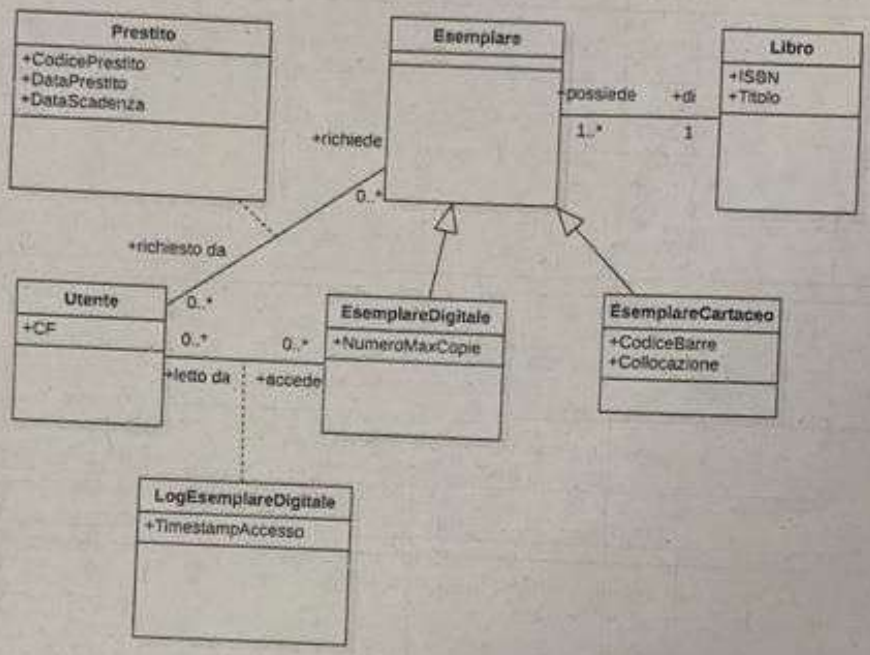
- **Esercizio 01** *(Punti 8) Si scriva una espressione in algebra relazionale che, se valutata, restituisca il nome degli utenti che non hanno mai restituito i prestiti in ritardo.*

- **Esercizio 02** *(Punti 8) Si scriva una interrogazione in SQL che, se valutata, fornisce il TITOLO del libro che ha avuto nell'anno 2022 il maggior numero di prestiti (si intende per numero di prestiti di un libro il numero di prestiti di tutti i suoi esemplari).*

- **Esercizio 03** *(Punti 7) Si implementino nel modo più adeguato i seguenti vincoli:*

  1. *Un utente non può avere più prestiti in corso (esemplari non ancora restituiti) di quanto previsto dal suo profilo.*
  2. *Se è presente un sollecito la restituzione è stata fatta dopo la data di scadenza.*
  3. *Un utente non può avere più di un profilo associato*

Esercizio 04 *(Punti 8)*



Si consideri la bozza di Class Diagram in figura.

- *Effettuare la ristrutturazione e motivarne il processo;*
- *Definire lo schema logico;*

Esercizio 05 *(Punti 7) Si implementino i seguenti trigger*

- *il primo trigger viene attivato quando viene inserita la data di sollecito per un prestito. L'effetto del trigger è che tutte le prenotazioni di quell'utente vengo automaticamente cancellate.*
- *il secondo trigger viene attivato ad ogni richiesta di prestito. L'effetto è che bisogna controllare se l'utente ha una prenotazione per quel libro. In tal caso la prenotazione corrispondente va rimossa.*

Esercizio 06 *(Punti 10) Si scriva una procedura che per ogni utente e per ogni anno estragga il numero dei prestiti e il numero delle prenotazioni. Si presupponga l'esistenza di una tabella* $STATISTICHE$*, inizialmente vuota, con il seguente schema:*
$STATISTICHE(CF, Anno, NumPrestiti, NumPren)$

Esercizio 07 *(Punti 14) Si scriva una funzione che riceve in ingresso una stringa contenente delle parole separate tra loro dal simbolo* $-$*. Si scriva una interrogazione in SQL dinamico che recupera i libri in cui* $TUTTE$ *le parole della stringa compaiono nella descrizione del Libro. La funzione restituisce una stringa contenente i titoli dei libri recuperati separati dal simbolo* ;.

APPELLO 27_04_2023.pdf

Basi di Dati, 27 aprile 2023

Mara Sangiovanni, Silvio Barra

DIETI, Corso di Laurea in Informatica, Università di Napoli "Federico II", Italy
E-mail: mara.sangiovanni@unina.it, silvio.barra@unina.it

Tempo: 2,5 ore.
Si consideri il seguente schema relazionale che descrive un frammento della base di dati per gestire una collezione di lavori scientifici (articoli) pubblicati su rivista. La tabella *ARTICOLI* contiene tutti articoli: *DOI* è l'identificativo dell'articolo; *ISSN* è l'identificativo della rivista. Gli autori dei lavori sono identificati dall'attributo *ORCID*. Nella tabella *SCRIVE* si codifica l'associazione tra articoli e gli autori; l'attributo *Ordine* indica la posizione dell'autore nella lista degli autori di un lavoro. Nella tabella *BIBLIOGRAFIA* viene tenuta traccia delle citazioni tra articoli: *DOI* è l'identificativo del lavoro che cita un articolo in bibliografia e *DOICitato* è l'identificativo del lavoro citato. La tabella *KEYWORD* tiene traccia delle parole chiave (*Parola*) associate a un articolo.

AUTORE(<u>*ORCID*</u>, *Nome*, *Cognome*, *Affiliazione*)
SCRIVE(<u>*DOI*, *ORCID*</u>, *Ordine*)
ARTICOLO(<u>*DOI*</u>, *Titolo*, *Anno*, *pagI*, *pagF*, *NomeRivista*, *ISSN*, *Numero*, *Fascicolo*)
BIBLIOGRAFIA(<u>*DOI*, *DOICitato*</u>)
KEYWORDS(<u>*DOI*, *Parola*</u>)

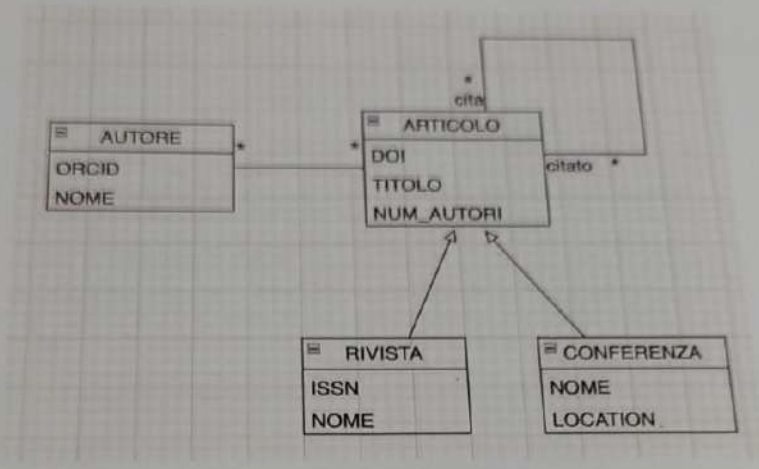
Esercizio 01 *(Punti 6) Si scriva una interrogazione in algebra relazionale che, se valutata, restituisca ORCID, nome e cognome degli autori che non hanno MAI scritto articoli in collaborazione (sempre autori unici dei loro articoli).*

Esercizio 02 *(Punti 9) Si scriva in SQL una interrogazione che restituisca coppie di identificativi di articoli (DOI1, DOI2) che rispettano il seguente criterio: le parole chiave associate a DOI2 sono TUTTE associate anche a DOI1.*

Esercizio 03 *(Punti 5) Si implementino nel modo più opportuno i seguenti vincoli:*

1. *Due autori di uno stesso articolo devono avere valori di ordine diverso;*
2. *Con riferimento alla tabella BIBLIOGRAFIA, l'anno di pubblicazione del lavoro identificato da DOI deve essere uguale o successivo all'anno di pubblicazione del lavoro identificato da DoiCitato;*
3. *Un articolo non deve citare più di una volta un altro articolo.*

Esercizio 04 *(Punti 6) Si consideri la bozza di Class Diagram in figura.*



- *Effettuare la ristrutturazione e motivarne il processo;*
- *Disegnare lo schema concettuale ristrutturato;*
- *Definire lo schema logico.*

Esercizio 05 *(Punti 9) Si scriva una funzione PLSQL che riceve l'ORCID di un autore e restituisca il suo indice di Hirsch (h-index). L'h-index di un autore è* **n** *se è autore di almeno* **n** *articoli, ognuno dei quali ha ricevuto almeno* **n** *citazioni.*
(SUGGERIMENTO: ordinare gli articoli per numero decrescente di citazioni).

Basi di Dati, 14 giugno 2023

Mara Sangiovanni, Silvio Barra

DIETI, Corso di Laurea in Informatica, Università di Napoli 'Federico II', Italy
E-mail: mara.sangiovanni@unina.it, silvio.barra@unina.it

Tempo: 2,5 ore.

Si consideri il seguente schema relazionale che descrive alcuni dati anagrafici: $PERSONA$, fornisce informazioni anagrafiche quali il Codice fiscale (chiave) data di nascita e data di morte (attributo parziale nullo per persone vive); $RESIDENZA$ descrive lo storico delle residenze di ogni persona. Una persona può avere molte residenze, di cui una sola corrente (la residenza corrente è quella che ha NULL associato a $DataF$) ed alcune concluse (La data di fine residenza è presente). $GENITORI$ associa ad ogni persona padre e madre (entrambi gli attributi posso essere parziali). $FAMIGLIA$ descrive i nuclei famigliari e $CFCapo$ è l'identificativo del capo famiglia. $COMPFAMIGLIA$ indica la composizione della famiglia.

$PERSONA(\underline{CF}, Nome, Cognome, DataN, DataM)$
$RESIDENZA(\underline{CF, DataI}, Via, Num, Citta, DataF)$
$GENITORI(\underline{CF}, CFMadre, CFPadre)$
$FAMIGLIA(\underline{codFamiglia}, CFCapo)$
$COMPFAMIGLIA(\underline{codFamiglia, codMembro})$

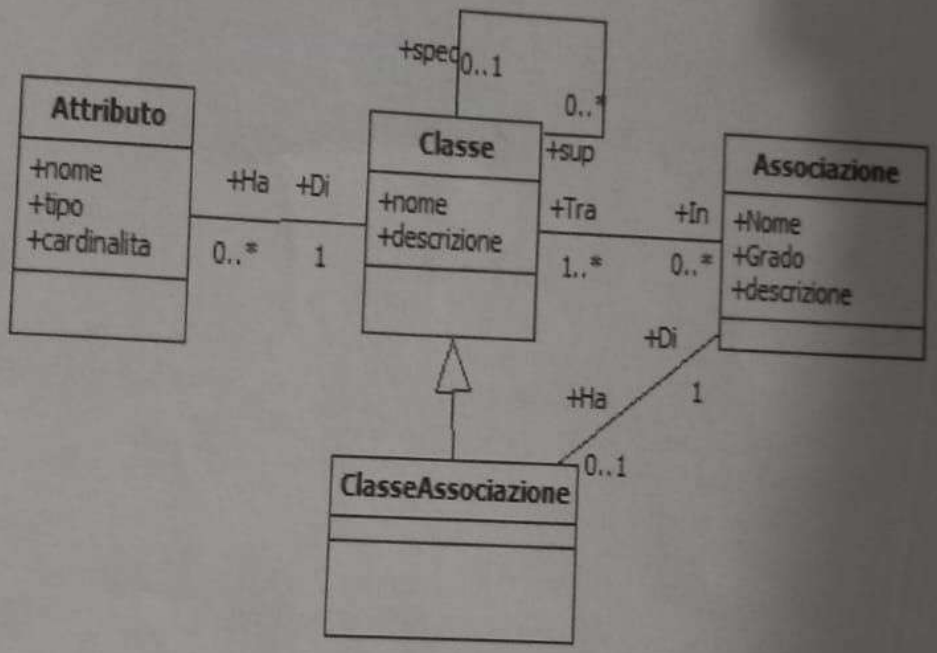
Esercizio 01 *(Punti 7) Si scriva una interrogazione in algebra relazionale che, se valutata, fornisca il Nome ed il Cognome di coloro che, in ogni anno, hanno cambiato residenza il più alto numero di volte.*

Esercizio 02 *(Punti 7) Si scriva una interrogazione in SQL che fornisca Nome e Cognome delle persone che sono state residenti per tutta la vita ESCLUSIVAMENTE nella stessa città.*

Esercizio 03 *(Punti 6) Si implementino nel modo più opportuno i seguenti vincoli (o trigger), motivando opportunamente le scelte fatte:*

1. *Quando viene cancellata una persona deve essere cancellata anche ogni associazione di parentela che la riguarda;*
2. *Ogni persona ha al più un padre ed una madre;*
3. *Non si può essere residenti nello stesso tempo in due luoghi diversi;*
4. *Quando viene fissata la data di morte viene anche chiusa in quella data la residenza;*

Esercizio 04 *(Punti 6) Si consideri la bozza di Class Diagram in figura.*



- *Effettuare la ristrutturazione e motivarne il processo;*
- *Disegnare lo schema concettuale ristrutturato;*
- *Definire lo schema logico.*

Esercizio 05 *(Punti 8) Usando SQL dinamico si scriva una funzione che riceve in ingresso una stringa contenente codici fiscali separati da ";" Per OGNUNO DEI CODICI FISCALI IN INPUT la funzione restituisce una stringa così fatta: Codice fiscale, Nome, Cognome ed età di tutti componenti del gruppo familiare ordinati per età DECRESCENTE (i valori sono separati da spazi, ogni componente da ",", i gruppi familiari da ";").*