Tutorato di Base di Dati
Lezione 6

Andrea Gasparetto

SQL: Creazione Tabelle

- CREATE TABLE Nome
  (Attributo Tipo [Vincolo {, Vincolo }]
  {, Attributo Tipo [Vincolo {, Vincolo }]}
  [, VincoloDiTabella {, VincoloDiTabella}])

- Esempio

```
CREATE TABLE Clienti (
        CodiceCliente       CHAR(3),
        Nome                CHAR(30),
        Città               CHAR(30),
        Sconto              INTEGER);
```

Tabelle Virtuali

CREATE VIEW NomeVista [(Attributo {, Attributo})]
AS EspressioneSelect

Esempio

```
CREATE VIEW OrdiniPerAgente(CodiceAgente, TotaleOrdini)
        AS SELECT CodiceAgente, SUM(Ammontare)
            FROM Ordini
            GROUP BY CodiceAgente;
```

SQL: Modifica dei dati

- **INSERT INTO** Tabella [(A1,..,An)]
  (**VALUES** (V1,..,Vn) | **AS** Select)

- **UPDATE** Tabella
  SET Attributo = Expr, …,
  Attributo = Expr
  WHERE Condizione

- **DELETE FROM** Tabella
  WHERE Condizione

INSERT

- La forma base del comando INSERT è la seguente:

```
INSERT INTO Tabella
 VALUES  (valoreA1,...,valoreAn),
         (valoreB1,...,valoreBn),
          ...
```

- dove `(valoreX1,...,valoreXn)` sono righe del tipo corrente di tabella (con gli attributi nella sequenza corretta!) e.g.

```
INSERT INTO Studenti
VALUES ('Paolo','Poli','71576', '1986', 'BL'),
       ('Giorgio','Conte','71577', '1941', 'AT');
```

INSERT

- Alternativamente si può usare la forma:

```
INSERT INTO
Tabella(colonna1,...,colonnam)
 VALUES (valoreA1,...,valoreAm),
              (valoreB1,...,valoreBm),
              ...
```

- `m` può essere < del numero di attributi `n` (le restanti colonne o prendono il valore di default o `NULL`)
- le colonne possono apparire in ordine diverso da quello in cui appaiono nella definizione di `Tabella`

INSERT

- Esempio

```
  INSERT INTO Studente (Matricola,
Nome, Cognome)
  VALUES (74324,'Gino','Bartali')
```

- Tutti i valori dichiarati **NOT NULL** e senza un valore di default dichiarato devono essere specificati

Insert e select

- E` possibile aggiungere le righe prodotte da una select ...

```
INSERT INTO Tabella AS Select
```

- Esempio: se `StNomeCognome(Nome, Cognome)` è una tabella con due campi di tipo adeguato ...

```
INSERT INTO StNomeCognome AS
            SELECT Nome, Cognome
            FROM Studenti;
```

DELETE

- La forma base del comando `DELETE` è la seguente:

```
DELETE FROM Tabella
WHERE condizione
```

- Cancella da `Tabella` le righe che soddisfano la condizione in `WHERE`: e.g.

```
DELETE FROM Esami
WHERE Voto<18;
```

- Senza la clausola `WHERE`

```
DELETE FROM Esami;
```

cancella tutte le righe (ma non la tabella)

DELETE

- La selezione delle righe da cancellare può essere basata anche su di una select. Es. Cancella gli studenti che non hanno sostenuto esami

```
DELETE FROM Studenti
WHERE Matricola NOT IN
      (SELECT Candidato FROM Esami);
```

- Strutturalmente simile alla **SELECT** (ma cancella intere righe)

UPDATE

- La forma base del comando UPDATE è:

```
UPDATE Tabella
     SET    attr1=exp1, ..,
          attrn=expn
   WHERE  condizione
```

dove `attri` ed `expi` devono avere il medesimo tipo

- Esempio:

```
UPDATE Studenti
SET    Tutor='71523'
WHERE  Matricola='76366'
       OR Matricola='76367'
```

UPDATE

- Aumenta di 1 punto il voto a tutti gli esami con voto > 23

```
UPDATE Esami
SET    Voto=Voto+1
WHERE  Voto>23 AND Voto<30;
```

- Anche in questo caso si possono usare condizioni che coinvolgono `SELECT`

Esercizi – Progettazione concettuale e logica

Una banca vuole memorizzare informazioni sui propri conti correnti. Un conto corrente, identificato da un numero, ha una data di apertura, è intestato ad una o più persone (una persona può essere cointestatario di più conti correnti). Una persona ha codice fiscale, nome, indirizzo. Ogni correntista può fare dei movimenti. Un movimento ha un codice univoco, una data, la persona che l'ha fatto, il valore (positivo o negativo) del movimento. Un movimento può essere un versamento su un conto, un prelievo da un conto, oppure un trasferimento da un conto corrente ad un altro conto corrente.

Si dia uno schema grafico a oggetti (secondo la notazione del libro di testo) della base di dati e si trasformi nello schema relazionale mostrandone la rappresentazione grafica (anche questa secondo la notazione del libro di testo, indicando la chiave primaria e le chiavi esterne).

Passo 1 – Identificazione Entità

- Conto Corrente
- Persona/Intestatario
- Movimento

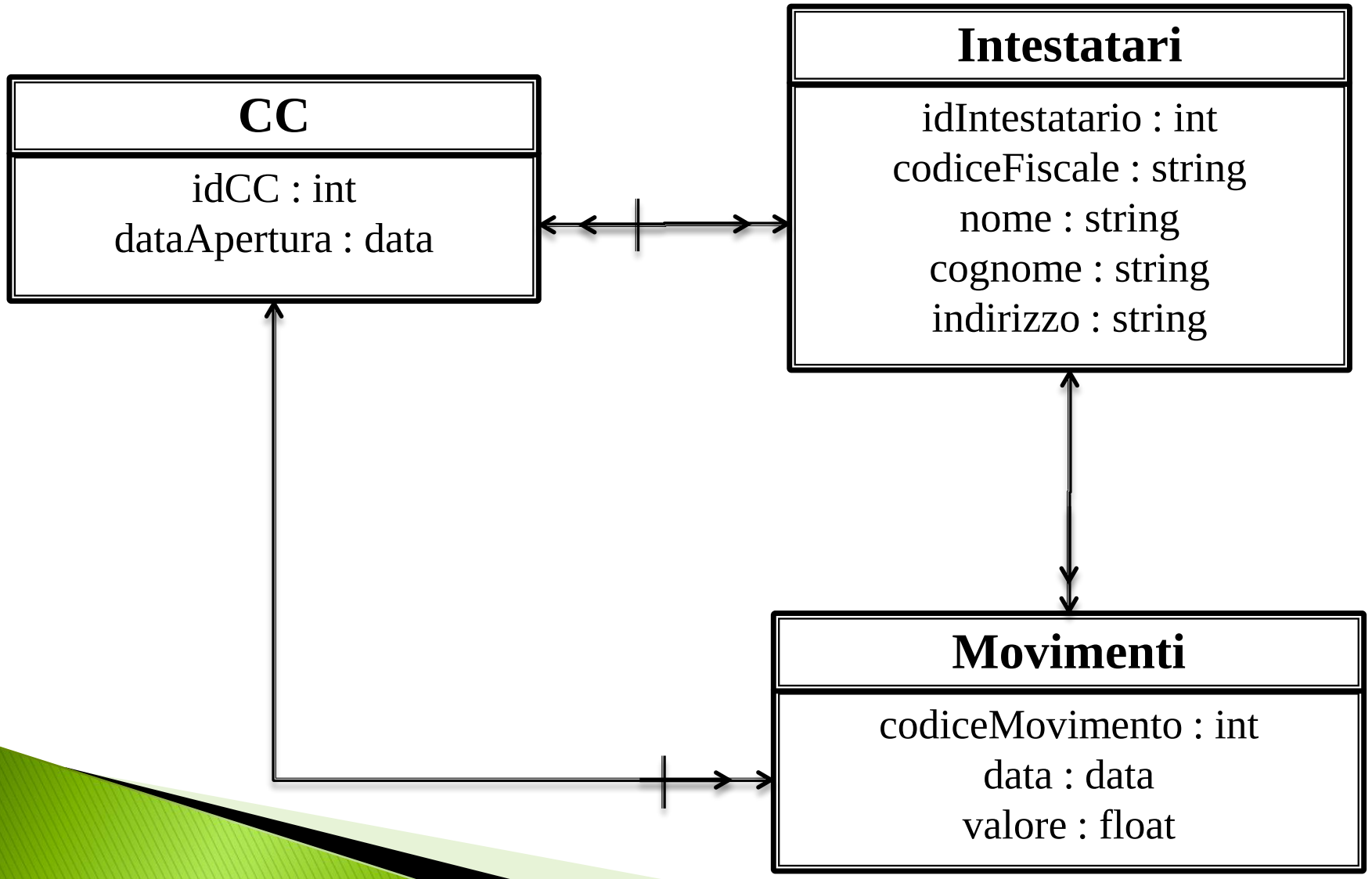
Passo 2 – Attributi richiesti

- Conto Corrente
  - Codice identificativo
  - Data di apertura
  - Intestatario/i
- Persona/Intestatario
  - Codice Fiscale
  - Nome
  - Indirizzo
- Movimento
  - Codice univoco
  - Data
  - Persona che ha fatto il movimento
  - Il valore (negativo o positivo)

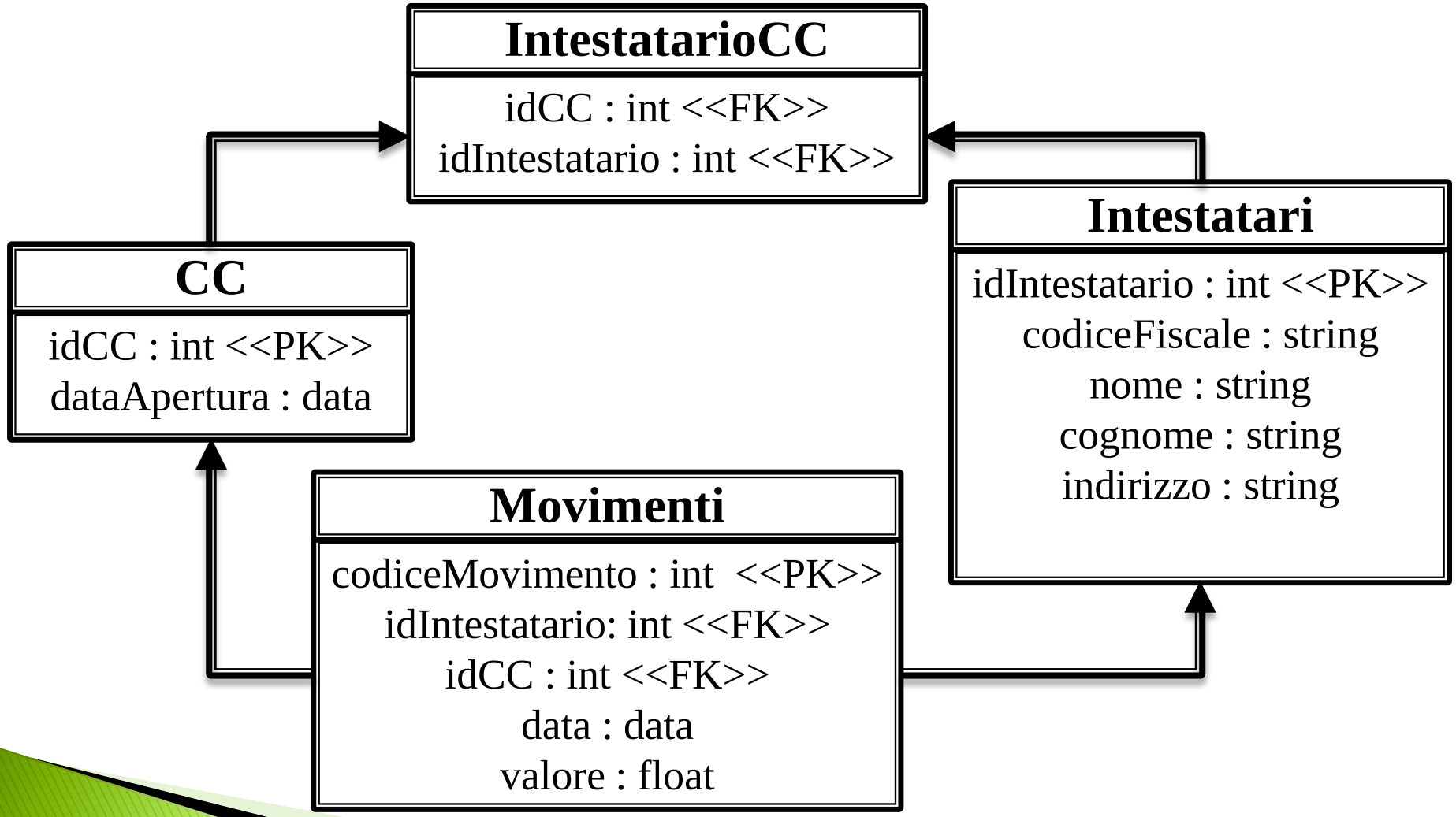
Passo 3 – Identificare informazioni non utili

- Identificare le richieste che possono essere rappresentate in maniera implicita
  - Un movimento può essere un versamento su un conto, un prelievo da un conto, oppure un trasferimento da un conto corrente ad un altro conto corrente.
- Il segno del valore ci da informazioni sul tipo di movimento
- Due movimenti possono rappresentare un trasferimento tra due conti
- **NB**: ogni scelta che viene fatta deve essere ESPLICITATA nell'esame.

Schema ad oggetti



Schema Relazionale



Esercizio 1

Persona(CF, Nome, Cognome, Età)
Città(Nome, NumeroAbitanti)
HaAbitato(CFPersona, NomeCittà, Da_Anno, A_Anno)
HaLavorato(CFPersona, NomeCittà, P_IVA_Ditta, Da_Anno, A_Anno)
Ditta(P_IVA, NumeroImpiegati, CapitaleSociale)
HaAvutoSedeIn(P_IVA_Ditta, NomeCittà, Da_Anno, A_Anno)

- Trovare le partite IVA delle ditte che hanno (o hanno avuto) sedi in tutte le città

```
SELECT P_IVA_Ditta
FROM HaAvutoSedeIn
GROUP BY P_IVA_Ditta
HAVING COUNT(DISTINCT NomeCittà) =
        (SELECT COUNT(*)
        FROM Città)
```

Esercizio 1 – Continua

- Per ogni ditta trovare il numero degli impiegati che vi hanno lavorato, risiedendo nella stessa città dove aveva sede la ditta.

```
SELECT HaLavorato.P_IVA_Ditta, COUNT(DISTINCT
HaLavorato.CFPersona)
FROM HaLavorato
        JOIN HaAvutoSedeIn ON HaLavorato.P_IVA_Ditta = HaAvutoSedeIn.P_IVA_Ditta
        JOIN HaAbitato ON HaAbitato.CFPersona = HaLavorato.CFPersona
WHERE HaAbitato.CFPersona = HaLavorato.CFPersona
                AND ((HaAbitato.Da_Anno <= HaAvutoSedeIn.A_Anno
                AND HaAbitato.A_Anno >= HaAvutoSedeIn.A_Anno)
                                        OR
                (HaAvutoSedeIn.Da_Anno <= HaAbitato.A_Anno
                AND HaAvutoSedeIn.A_Anno >= HaAbitato.A_Anno))
GROUP BY HaLavorato.P_IVA_Ditta
```

Esercizio 2

INSEGNAMENTI(Codice,Denominazione)
STUDENTI(Matricola,Cognome,Nome)
ESAMI(Studente,Corso,Data,Voto)

- Trovare la media dei voti riportati agli esami per ciascun insegnamento (indicando codice, denominazione e voto medio)

```
SELECT Codice, Denominazione, AVG(Voto)
FROM Insegnamenti JOIN Esami ON Codice = Corso
GROUP BY Codice, Denominazione
```

Esercizio 2 – Continua

INSEGNAMENTI(Codice,Denominazione)
STUDENTI(Matricola,Cognome,Nome)
ESAMI(Studente,Corso,Data,Voto)

- Trovare gli studenti (mostrando il numero di matricola) che hanno superato almeno due esami dopo il 1/1/2000

```
SELECT Studente
FROM Esami
WHERE Data > '01-Jan-2000'
GROUP BY Studente
HAVING COUNT(*) >= 2
```

Esercizio 2 – Continua

INSEGNAMENTI(Codice,Denominazione)
STUDENTI(Matricola,Cognome,Nome)
ESAMI(Studente,Corso,Data,Voto)

- Trovare denominazione, data e voto per gli esami superati da Mario Rossi

```
SELECT Denominazione, Data, Voto
FROM Esami, Studenti, Insegnamenti
WHERE Studente = Matricola
        AND Codice = Corso
        AND Nome = “Mario”
        AND Cognome = “Rossi”
```

$$\pi_{Denominazione,Data,Voto}(Insegnamenti \bowtie_{Codice=Corso} Esami \bowtie_{Studente=Matricola} \sigma_{Cognome='Pestalozzi' \wedge Nome='Bartolomeo'}(Studenti))$$

Esercizio 3

Tesi(Studente, Titolo, Data, Voto, Relatore, Correlatore*)
Docenti(Matricola, Cognome, Nome, Dipartimento)
Studenti(Matricola, Cognome, Nome)
Dipartimenti(Sigla, Nome, Area)

- Trovare nome, cognome e matricola degli studenti che hanno avuto nel 2011 sia il relatore che, se esiste, il correlatore di un dipartimento di area ingegneristica.

```
SELECT Studenti.Matricola, Studenti.Nome, Studenti.Cognome
FROM Studenti
JOIN Tesi ON Studenti.Matricola = Studente
JOIN Docenti ON Relatore = Docenti.Matricola
JOIN Dipartimenti on Dipartimento = Sigla
WHERE Data BETWEEN '01-jan-2011' AND '31-dec-2011'
AND Area = 'Ingegneria'
AND (Correlatore IS NULL
OR Correlatore IN (SELECT Matricola
                   FROM Docenti JOIN Dipartimenti ON Dipartimento = Sigla
                   WHERE Area = 'Ingegneria'))
```

Esercizio 3 – Continua

Tesi(Studente, Titolo, Data, Voto, Relatore, Correlatore*)
Docenti(Matricola, Cognome, Nome, Dipartimento)
Studenti(Matricola, Cognome, Nome)
Dipartimenti(Sigla, Nome, Area)

- Trovare nome e cognome dei docenti tali che gli studenti di cui sono stati relatori unici (cioè senza correlatore) hanno preso tutti più di 100 alla tesi.

```
SELECT Nome, Cognome
FROM Docenti
WHERE Matricola NOT IN  (SELECT Matricola
                         FROM Docenti JOIN Tesi ON Matricola = Relatore
                         WHERE Correlatore IS NULL AND voto < 100)
```

Esercizio 4

Persone(CodiceFiscale,Cognome,Nome,DataDiNascita)
Dipendenti(CodiceFiscale,Filiale,Qualifica)
Qualifiche(Codice,Descrizione)
Filiali(Codice,Città,Direttore)
Agenzie (Numero,Filiale,Indirizzo,Reggente)
ContiCorrenti (Numero,Agenzia,Filiale)
Titolarita-CC (Conto,Titolare)

- Trovare le partite IVA delle ditte che hanno (o hanno avuto) sedi in tutte le città

```
SELECT P_IVA_Ditta
FROM HaAvutoSedeIn
GROUP BY P_IVA_Ditta
HAVING COUNT(DISTINCT NomeCittà) = (SELECT COUNT(*)
                                   FROM Città)
```

Esercizio 4 – Continua

- Per ogni ditta trovare il numero degli impiegati che vi hanno lavorato, risiedendo nella stessa città dove aveva sede la ditta.

```
SELECT HaLavorato.P_IVA_Ditta, COUNT(DISTINCT HaLavorato.CFPersona)
FROM HaLavorato
   JOIN HaAvutoSedeIn ON HaLavorato.P_IVA_Ditta = HaAvutoSedeIn.P_IVA_Ditta
  JOIN HaAbitato ON HaAbitato.CFPersona = HaLavorato.CFPersona
WHERE HaAbitato.CFPersona = HaLavorato.CFPersona
          AND ((HaAbitato.Da_Anno <= HaAvutoSedeIn.A_Anno
          AND HaAbitato.A_Anno >= HaAvutoSedeIn.A_Anno)
          OR (HaAvutoSedeIn.Da_Anno <= HaAbitato.A_Anno
               AND HaAvutoSedeIn.A_Anno >= HaAbitato.A_Anno))
GROUP BY HaLavorato.P_IVA_Ditta
```

Esercizio 4 – Continua

Persone(CodiceFiscale,Cognome,Nome,DataDiNascita)
Dipendenti(CodiceFiscale,Filiale,Qualifica)
Qualifiche(Codice,Descrizione)
Filiali(Codice,Città,Direttore)
Agenzie (Numero,Filiale,Indirizzo,Reggente)
ContiCorrenti (Numero,Agenzia,Filiale)
Titolarita-CC (Conto,Titolare)

- Per ogni agenzia, mostrare numero, codice della filiale, città e cognome del reggente.

```
SELECT numero, codice, città, cognome
FROM Agenzie
        JOIN Filiali ON filiali = codice
        JOIN Persone ON reggente = codicefiscale
```

$$\pi_{numero,codice,citta,cognome}(agenzie \bowtie_{filiale=codice} filiali \bowtie_{reggente=codicefiscale} persone)$$