



SAPIENZA
UNIVERSITÀ DI ROMA

Facoltà di Ing. dell'Informazione, Informatica e Statistica
Laurea in Informatica

Basi di Dati, Modulo 2

Prof. Toni Mancini
Dipartimento di Informatica
<http://tmancini.di.uniroma1.it>

Slides B.3.2 (S.B.3.2)

Basi di dati
La fase di progettazione
Progettazione delle Funzionalità

Progettazione delle Funzionalità

Lo schema concettuale contiene una specifica per operazione di classe o di use-case in termini di:

- ▶ **segnatura**: nome dell'operazione, nome e tipo degli argomenti, tipo dell'eventuale valore di ritorno
- ▶ **precondizioni**
- ▶ **postcondizioni**.

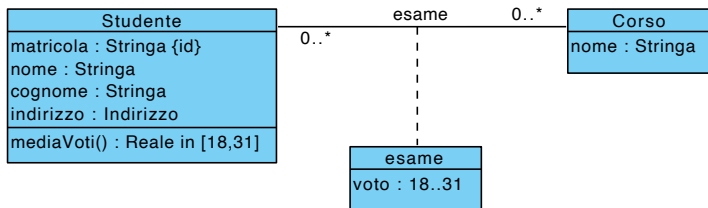
Durante la fase di progettazione, per ogni operazione, va definita una **specifica realizzativa** in termini di:

- ▶ **segnatura**: nome dell'operazione, nome e dominio degli argomenti, dominio dell'eventuale valore di ritorno (i domini sono i domini supportati dal DBMS fatti corrispondere in fase di progettazione ai tipi concettuali)
- ▶ **algoritmo in pseudo-codice** contenente eventuali comandi SQL per l'interazione con la base dati.

Progettazione delle operazioni di classe

Approccio del tutto analogo, ricordando che, a differenza delle operazioni di use-case, le operazioni di classe hanno sempre un **oggetto di invocazione**.

Esempio: schema concettuale



Progettazione delle operazioni di classe (2)

Esempio: schema concettuale (continua)

Specifica use-case **Studente**

mediaVoti() : Reale in [18,31]

precondizioni: L'istanza `this` è coinvolta in almeno una istanza dell'associazione **esame**:

$\exists c \text{ esame}(\text{this}, c).$

postcondizioni:

Modifica del Livello Estensionale dei Dati: Nessuna

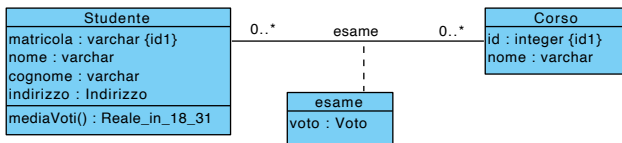
Valore di Ritorno: Detto $V = \{c, v \mid \text{esame}(\text{this}, c) \wedge \text{voto}(\text{this}, c, v)\}$,

$$\text{result} = \frac{\sum_{(c,v) \in V} v}{|V|}$$

End

Progettazione delle operazioni di classe (3)

Diagramma delle classi ristrutturato



Progettazione delle operazioni di classe (4)

Progettazione della base dati

Dominio Voto: si decide di definire un dominio utente Voto. Il dominio è basato sul dominio integer; i valori ammessi sono gli interi tra 18 e 31.

Studente (matricola:varchar)

Corso (codice:integer)

esame (studente:varchar, corso:integer, voto:Voto)

VincoloDB **foreign key:** studente references Studente(matricola)

VincoloDB **foreign key:** corso references Corso(codice)

Progettazione delle operazioni di classe (5)

Specifica realizzativa classe **Studente**

mediaVoti() : Reale_in_18_31

algoritmo:

1. Memorizza in *Q* il risultato della seguente query SQL:

```
select s.matricola , avg(e.voto) as media
from Studente s LEFT OUTER JOIN esame e
on s.matricola = e.studente
where s.matricola = PAR_1
group by s.matricola
```

dopo aver rimpiazzato 'PAR_1', con il valore dell'attributo matricola di this.

2. Se *Q* è vuoto, genera l'errore 'Lo studente non esiste'
3. Altrimenti, se *Q* è (... , NULL), genera l'errore 'Lo studente non ha superato alcun esame'
4. Altrimenti restituisci il valore dell'attributo media dell'unica ennupla di *Q*.

End

Dove implementare le operazioni di classe? (Cenni)

Operazioni di classe:

- ▶ Dipende dalla tecnologia per il back-end scelta in fase di progettazione
- ▶ Ad esempio, in caso di back-end implementati con approccio object-oriented, è possibile creare **class** associate alle classi UML del diagramma ristrutturato (tutte o solo alcune) e implementare le operazioni di classe come metodi di queste **class**
- ▶ È anche possibile implementare le operazioni di classe nel DBMS utilizzando il costrutto **CREATE FUNCTION**, al fine di renderle disponibili a tutti i back-end che accedono alla base dati. Tali FUNCTION possono essere invocate con:

```
SELECT myFunction ( argomenti ) ;
```

o utilizzate in query più complesse.

Dove implementare le operazioni di classe? (Cenni) (2)

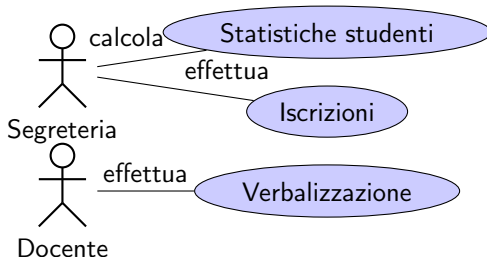
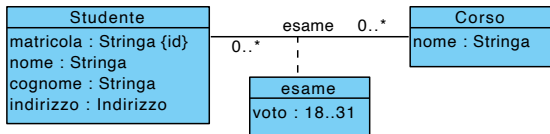
Esempio:

```
CREATE OR REPLACE FUNCTION Studente_mediaVoti(_matr varchar)
  returns Reale_in_18_31 AS $BODY$
DECLARE
  matr varchar := NULL;
  media numeric := NULL;
BEGIN
  select s.matricola , avg(e.voto) INTO matr, media
  from Studente s LEFT OUTER JOIN esame e
    on s.matricola = e.studente
  where s.matricola = _matr
  group by s.matricola;

  IF (matr IS NULL) THEN
    RAISE EXCEPTION 'Error 001 — Lo studente non esiste';
  ELSE
    IF (media IS NULL) THEN
      RAISE EXCEPTION 'Error 002 — Lo studente non ha esami';
    ELSE
      RETURN media;
    END IF;
  END IF;
END;
$BODY$ LANGUAGE 'plpgsql';
```

Progettazione degli use case

Esempio: Schema concettuale



Progettazione degli use case (2)

Schema concettuale: Specifica concettuale di use-case

Specifica use-case Verbalizzazione

verbalizzaEsame(s : Studente, c : Corso, v : 18..31)

precondizioni: L'istanza *s* non è coinvolta in alcuna istanza dell'associazione *esame* con l'istanza *c*:

$$\neg \exists c \text{ esame}(s, c).$$

postcondizioni:

Modifica del Livello Estensionale dei Dati: Il livello estensionale dei dati al termine dell'esecuzione della funzione differisce da quello di partenza come segue:

Elementi del dominio di interpretazione : invariati

Variazioni nelle ennuple di predicati: *esame*(*s*, *c*), *voto*(*s*, *c*, *v*).

Valore di Ritorno: nessuno.

End

Progettazione degli use case (3)

Specifica realizzativa di use-case

Specifica realizzativa use-case Verbalizzazione

verbalizzaEsame(s : varchar, c : integer, v : Voto)

algoritmo:

1. Esegui il seguente comando SQL:
insert into esame(studente, corso, voto) **values** (PAR_1, PAR_2, PAR_3)
dopo aver rimpiazzato 'PAR_1', 'PAR_2', 'PAR_3' con i valori dei parametri attuali, rispettivamente, s, c, v.
2. Se il comando precedente restituisce un errore di **vincolo di chiave violato**, allora genera l'errore 'Lo studente di matricola s ha già superato l'esame per il corso di codice c'.

End

Progettazione degli use case (4)

Schema concettuale: Specifica concettuale di use-case

Specifica use-case StatisticheStudenti

mediaVoti(s : Studente) : reale in [18,31]

precondizioni: L'istanza s è coinvolta in almeno un'istanza dell'associazione **esame**:

$\exists c \text{ esame}(s, c).$

postcondizioni:

Modifica del Livello Estensionale dei Dati: nessuna

Valore di Ritorno: result è tale da soddisfare la formula:

$\text{mediaVoti}_{\text{Studente}}(s, \text{result}).$

End

Progettazione degli use case (5)

Specifica realizzativa di use-case

Specifica realizzativa use-case StatisticheStudenti

mediaVoti(s : varchar) : Reale_18_31

algoritmo:

1. Memorizza in result il risultato di:

SELECT Studente_mediaVoti(PAR_1) ;

dopo aver rimpiazzato 'PAR_1' con il valore del parametro attuale s.
(La query restituisce un reale o solleva una eccezione.)

2. Se la query ha lanciato una eccezione, rilancia l'errore e termina.
3. Altrimenti, restituisci result.

End

Progettazione degli use case (6)

Schema concettuale: Specifica concettuale di use-case

Specifica use-case StatisticheStudenti (continua)

numMedioEsami() : reale ≥ 0

precondizioni: Il livello estensionale dei dati definisce almeno una istanza di classe **Studente**:
 $\exists s \text{ Studente}(s)$.

postcondizioni:

Modifica del Livello Estensionale dei Dati: nessuna

Valore di Ritorno: **result** è pari al numero di istanze di associazione **esame** definite nel livello estensionale diviso per il numero di istanze di entità **Studente**. Formalmente, siano:

$$E = \{(s, c) \mid \text{esame}(s, c)\} \quad \text{e} \quad S = \{s \mid \text{Studente}(s)\}$$

gli insiemi, rispettivamente, di tutte le coppie (s, c) istanze dell'associazione **esame** e di tutte le istanze dell'entità **Studente**. Si ha: $\text{result} = \frac{|E|}{|S|}$.

End

Progettazione degli use case (7)

Specifica realizzativa di use-case

Specifica realizzativa use-case StatisticheStudenti (continua)

numMedioEsami() : RealeNonNeg

algoritmo:

1. Esegui la seguente query SQL e memorizzane il risultato nella variabile Q:

```
select e.numEsami/s.numStudenti as numMedioEsami
from (select count(*) as numEsami from esame) e,
      (select count(*) as numStudenti from Studente) s
```

(La query restituisce un reale o un'eccezione di 'divisione per zero'.)

2. Se il risultato della query è un'eccezione di 'divisione per zero', generare l'errore 'Non esistono studenti' e termina.
3. Altrimenti, restituisci Q.

End

Nota: Il dominio **RealeNonNeg** è stato scelto, all'inizio della Fase di Progettazione, come corrispondente del dominio concettuale **reale** ≥ 0 .

Dove implementare le operazioni di use-case? (Cenni)

Operazioni di use-case:

- ▶ Dipende dalla tecnologia per il back-end scelta in fase di progettazione
- ▶ In generale, nel livello più esterno del back-end, di modo che siano accessibili agli utenti del sistema (attori del diagramma concettuale)
- ▶ Esempio: in applicazioni web RESTful, sono procedure accessibili tramite gli endpoint REST
- ▶ Autenticazione utenti tramite, ad es., token OAuth2
- ▶ L'architettura può essere più complessa e prevedere più livelli (multi-layered applications)

Dove implementare le operazioni di use-case? (Cenni) (2)

Esempio:

(back-end web RESTful in Flask, <https://flask.palletsprojects.com>)

```
from flask import ...
from flask_restful import Resource ...
import psycopg2 # Driver Python per PostgreSQL
...

class MediaVoti(Resource):
    # Invocato dal front-end mediante chiamata HTTPS di tipo GET, ad es.:
    # GET https://app.site/statistichestudenti/mediavoti/<matricola>
    def get(self, matricola: str) -> "tuple[dict, int]":
        try:
            with get_db() as db:
                with db.cursor() as cur:
                    cur.execute(
                        "SELECT Studente_mediaVoti(%s) as media", (matricola,)
                    ) # puo' sollevare eccezioni
                    result = cur.fetchone() # leggi la prima ennupla restituita
                    return # crea oggetto JSON da restituire
                    {
                        "matricola": matricola,
                        "media": result["media"]
                    }, 200 # HTTP OK
        except psycopg2.DatabaseError as err:
            if (codice associato ad err == 001):
```

Dove implementare le operazioni di use-case? (Cenni) (3)

```
        abort(codice HTTP, description("Lo studente non esiste"))
    elif ...
except Exception:
    abort(500, description="Internal Server Error")
...
```

Dove implementare le operazioni di use-case? (Cenni) (4)

```
from flask import ...
from flask_restful import Resource ...
import psycopg2 # Driver Python per PostgreSQL
...

class VerbalizzaEsame(Resource):
    # Invocato dal front-end mediante chiamata HTTPS di tipo POST, ad es.:
    # POST https://app.site/verbalizzazione/verbalizzaesame
    # con body JSON { studente:str, corso:int, voto:int }
    def post(self) -> "tuple[dict, int]":
        (studente, corso, voto) = leggi il body della richiesta POST
        try:
            with get_db() as db:
                with db.cursor() as cur:
                    cur.execute(
                        "INSERT INTO esame(studente, corso, voto) values (%s, %s, %s)"
                        ", (studente, corso, voto)"
                    ) # puo' sollevare eccezioni
                    ...
            return 201 # HTTP CREATED
        except psycopg2.DatabaseError as err:
            if (codice associato ad err == ...):
                abort(codice HTTP, description("Lo studente non esiste"))
            elif ...
        except Exception:
            abort(500, description="Internal Server Error")
        ...
```

Dove implementare le operazioni di use-case? (Cenni) (5)