ESERCITAZIONE: AZIENDA

Homework 24 ottobre 2002

Emanuel Weitschek
emanuel@dia.uniroma3.it

Prerequisiti (software)

- PostgreSQL

- pgAdmin

- Driver JDBC

Ambito

- Si consideri una base di dati che contiene informazioni
  - sugli **impiegati**,
  - i **progetti** e
  - le **sedi** di una azienda,
  - con le **partecipazioni** degli **impiegati** ai **progetti**
  - e le **sedi** di **svolgimento** dei **progetti** stessi;

- *ESERCIZIO: effettuare lo schema E-R*

Relazioni

- IMPIEGATI(Matricola, Cognome, Nome, Progetto), con vincolo di integrità referenziale fra Progetto e la relazione Progetti
- PROGETTI(Codice, Titolo)
- SEDI(Nome, Città, Indirizzo)
- SVOLGIMENTO(Progetto, Sede), con vincoli di integrità referenziale fra Progetto e la relazione Progetti fra Sede e la relazione Sedi

Creazione del db

- Andare su pgAdmin e creare un nuovo database chiamato “azienda”
- Eseguire i seguenti script per la generazione delle tabelle:

PROGETTI(Codice, Titolo)

```
CREATE TABLE progetti(
        codice integer NOT NULL,
        titolo character varying(64) NOT NULL,
        CONSTRAINT pk_progetti PRIMARY KEY (codice)
)
```

Creazione del db

- Andare su pgAdmin e creare un nuovo database chiamato “azienda”
- Eseguire i seguenti script per la generazione delle tabelle:

SEDI(Nome, Città, Indirizzo)

```
CREATE TABLE sedi(
        nome character varying(64) NOT NULL,
        citta character varying(64) NOT NULL,
        indirizzo character varying(64) NOT NULL,
        CONSTRAINT pk_sedi PRIMARY KEY (nome)
)
```

Creazione del db

- Eseguire i seguenti script per la generazione delle tabelle:

IMPIEGATI(Matricola, Cognome, Nome, Progetto), con vincolo di integrità referenziale fra Progetto e la relazione Progetti

```
CREATE TABLE impiegati(
        matricola integer NOT NULL,
        nome character varying(64) NOT NULL,
        cognome character varying(64) NOT NULL,
        progetto integer,
        CONSTRAINT pk_impiegati PRIMARY KEY (matricola),
        CONSTRAINT progetti_codice_fkey FOREIGN KEY
        (progetto )      REFERENCES progetti(codice) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Creazione del db

- Eseguire i seguenti script per la generazione delle tabelle:

SVOLGIMENTO(Progetto, Sede), con vincoli di integrità referenziale fra Progetto e la relazione Progetti fra Sede e la relazione Sedi

```
CREATE TABLE svolgimento(
        progetto integer NOT NULL,
        sede character varying(64) NOT NULL,
        CONSTRAINT pk_svolgimento PRIMARY KEY (progetto,sede),
        CONSTRAINT progetti_codice_fkey FOREIGN KEY
        (progetto )     REFERENCES progetti(codice) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
        CONSTRAINT sedi_nome_fkey FOREIGN KEY (sede)
        REFERENCES sedi(nome) MATCH SIMPLE ON
        UPDATE NO ACTION ON DELETE NO ACTION
)
```

Alcune istanze nel db

- **PROGETTI(Codice, Titolo)**
  1, SUPERDB
  2, ULTRADB
- **SEDI(Nome, Città, Indirizzo)**
  A, Roma, Via Ostiense 196
  B, Milano, Piazza Sempione 10
- **IMPIEGATI(Matricola, Cognome, Nome, Progetto)**
  123, Rossi, Mario, 1
  124, Verdi, Giulio,2
  125, Gialli, Chiara,2
- **SVOLGIMENTO(Progetto, Sede)**
  1,A
  2,B

Inserimento di alcune istanze nel db

- Eseguire i seguenti script per l'inserimento di alcune istanze nel db:

```
INSERT INTO progetti VALUES (1,'SUPERDB');
INSERT INTO progetti VALUES (2,'ULTRADB');

INSERT INTO sedi VALUES ('A','Roma','Via Ostiense
196');
INSERT INTO sedi VALUES ('B','Milano','Piazza Sempione
10');

INSERT INTO impiegati VALUES (123,'Rossi','Mario',1);
INSERT INTO impiegati VALUES (124,'Verdi','Giulio',2);
INSERT INTO impiegati VALUES (125,'Gialli','Chiara',2);

INSERT INTO svolgimento VALUES (1,'A');
INSERT INTO svolgimento VALUES (2,'B');
```

Query 1.1

- L’interrogazione SQL che trova i progetti che si svolgono presso sedi situate a Roma, mostrando, per ciascuno, il codice e il titolo del progetto e il nome della sede:

```
select codice, titolo, nome
from progetti, svolgimento, sedi
where codice = progetto
and sede = nome
and citta = 'Roma'
```

Query 1.2

- L’interrogazione SQL che conta, per ciascun progetto, gli impiegati che lavorano ad esso
- Se interessano solo i codici dei progetti:

```
select progetto, count (*)
from impiegati
group by progetto
```

Query 1.2

- L’interrogazione SQL che conta, per ciascun progetto, gli impiegati che lavorano ad esso
- Se invece interessano anche i titoli:

```
select progetto, titolo, count (*)
from impiegati, progetti
where progetto = codice
group by progetto, titolo
```

Query 1.2

- L’interrogazione SQL che conta, per ciascun progetto, gli impiegati che lavorano ad esso
- Se ci sono progetti cui non lavora nessun impiegato:

```
select codice, titolo, count(matricola)
from progetti left join impiegati
on codice = progetto
group by codice
```

Query 1.2

- L’interrogazione SQL che conta, per ciascun progetto, gli impiegati che lavorano ad esso
- Se ci sono progetti cui non lavora nessun impiegato, ma il sistema non supporta il join esterno:

```
select codice, titolo, count(matricola)
from progetti join impiegati on codice = progetto
group by codice, titolo
union select codice, titolo, 0
from progetti
where codice not in (select progetto
from impiegati)
```

Query 1.3

- L'interrogazione SQL che trova tutti i progetti (con codice e titolo) e, per ciascuno di essi, gli impiegati coinvolti (mostrando matricola e cognome)

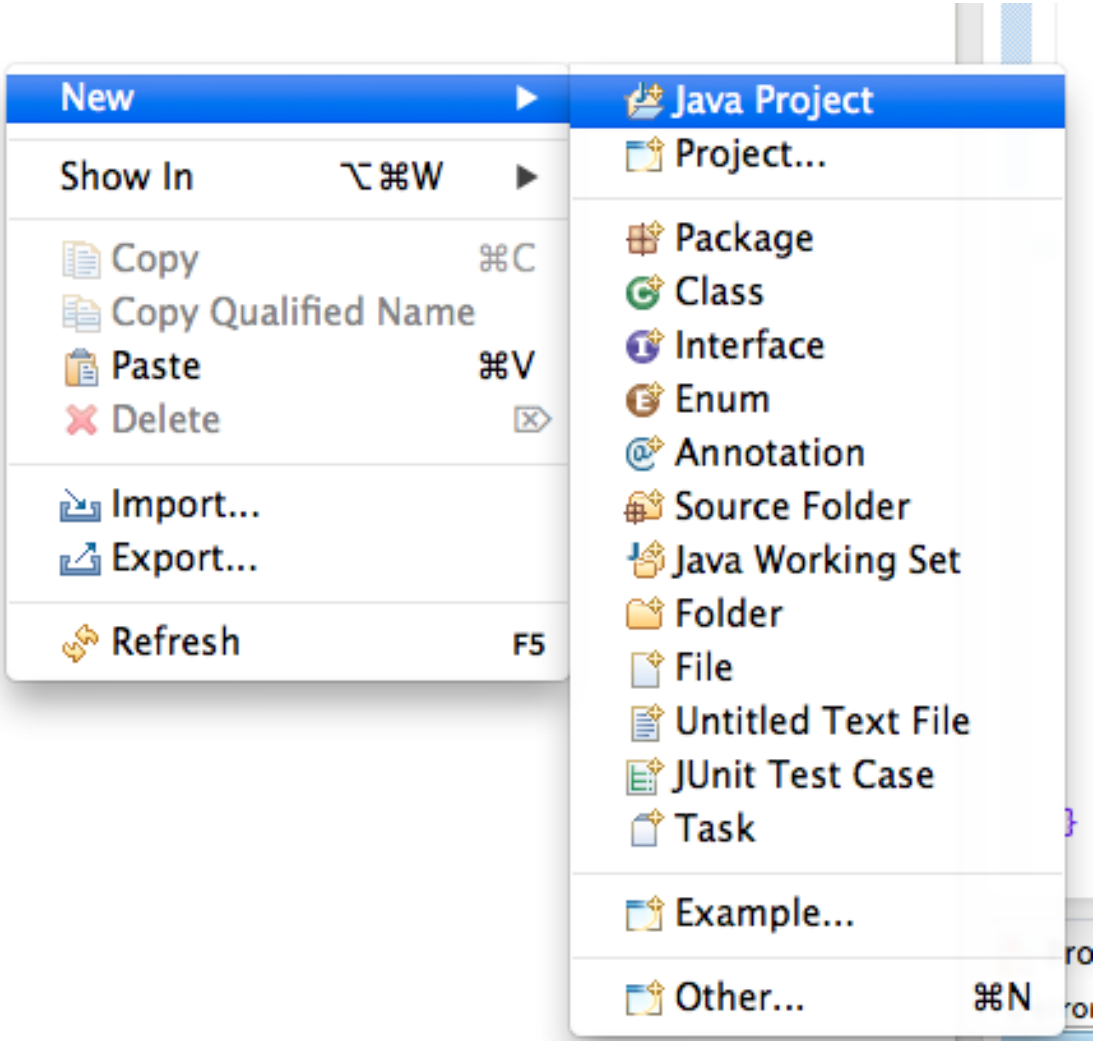
```
select Codice, Titolo, Matricola, Cognome
from Progetti left join Impiegati
on Codice = Progetto
order by Codice
```

oppure

```
select Codice, Titolo, Matricola, Cognome
from Progetti, Impiegati
where Codice = Progetto
order by Codice
```

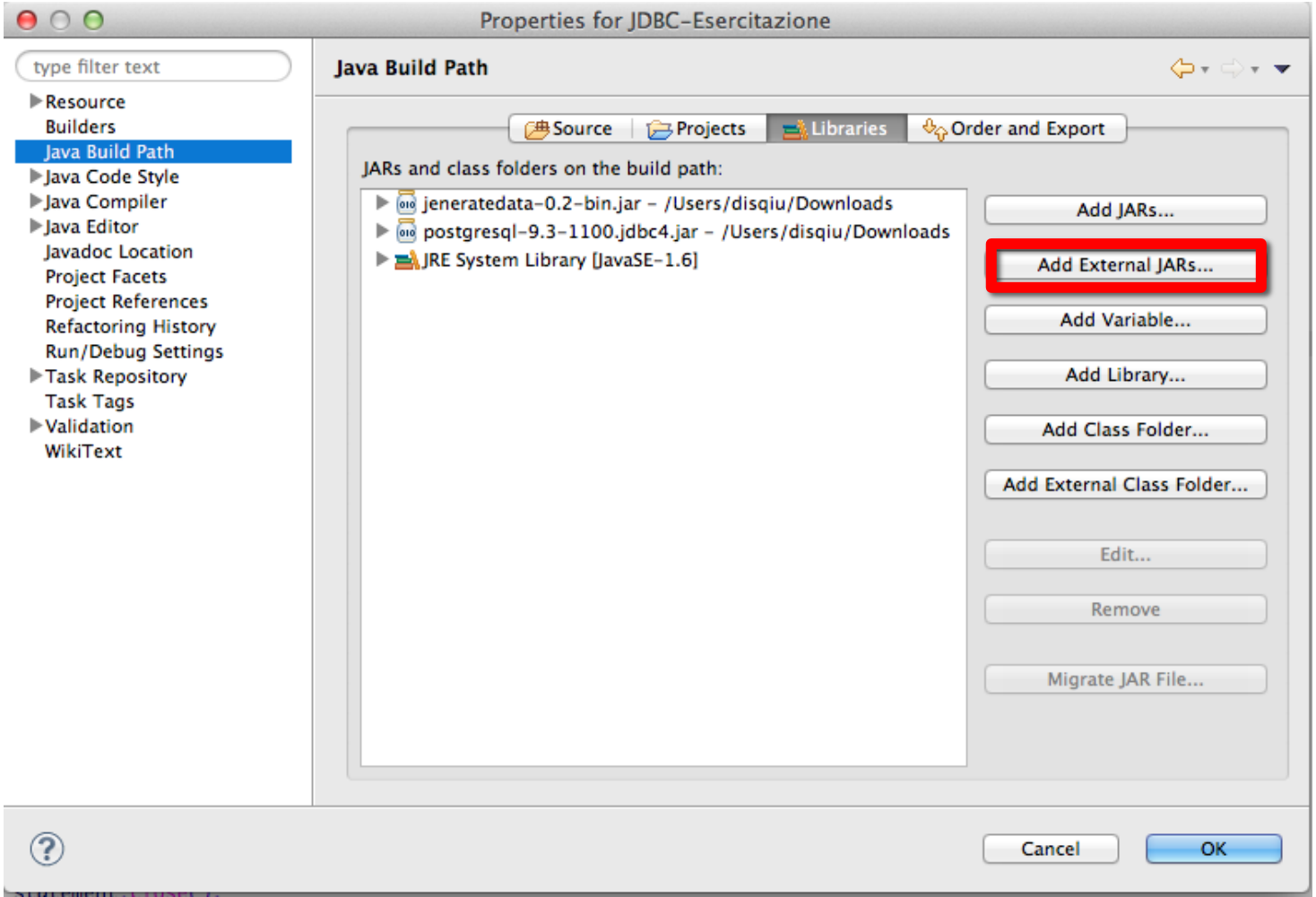
Query 1.3 in JDBC

Creazione nuovo progetto



Import jar esterni

Progetto > Properties > Java Build Path > Add External Jars



Cosa c’è da importare?

- postgresql-9.3-1100.*
  - Driver JDBC per postgres
  - postgresql-9.3-1100.jdbc3 => versione java 1.5 o maggiore*
  - postgresql-9.3-1100.jdbc4 => versione java 1.6 o maggiore*
  - postgresql-9.3-1100.jdbc41 => versione java 1.7 o maggiore*

- jeneratedata-0.2-bin.jar
  - Generazione pseudo randomica di valori
  - Stringhe, Nomi, Date etc

* Versione java: java –version

Struttura del codice

Main:

```
public static void main(String a[]) throws SQLException {
        Connection con = driverEConnessione();
        ResultSet result = interrogazioneProgettiImpiegati(con);
        boolean fineResult = false;
        boolean inizio = true;
        while (ciSonoAncoraProgetti(inizio, fineResult, result)) {
            fineResult = stampaProgettoEImpiegati(result);
            inizio = false;
        }
    }
```

Struttura del codice

Main:

```
public static void main(String a[]) throws SQLException {
        Connection con = driverEConnessione();
        ResultSet result = interrogazioneProgettiImpiegati(con);
        boolean fineResult = false;
        boolean inizio = true;
        while (ciSonoAncoraProgetti(inizio, fineResult, result)) {
            fineResult = stampaProgettoEImpiegati(result);
            inizio = false;
        }
    }
}
```

Connessione al db

Struttura del codice

Connessione al db:

```
static Connection driverEConnessione() {
    // inserire driver
        try {
            Class.forName("org.postgresql.Driver");
        } catch (Exception e) {
            System.out.println("Driver non OK");
        }
        Connection con = null;
        try {
            // inserire URL, username e password
            String url = "jdbc:postgresql://localhost/azienda";
            String username = "postgres";
            String pwd = "emanuel";
            con = DriverManager.getConnection(url, username, pwd);
        } catch (SQLException e) {
            System.out.println("Errore nella connessione");
            System.out.println(e.getErrorCode() + " " + e.getSQLState() + e.getMessage());
        }
        return con;
```

Struttura del codice

Main:

```
public static void main(String a[]) throws SQLException {
        Connection con = driverEConnessione();
        ResultSet result = interrogazioneProgettiImpiegati(con);
        boolean fineResult = false;
        boolean inizio = true;
        while (ciSonoAncoraProgetti(inizio, fineResult, result)) {
            fineResult = stampaProgettoEImpiegati(result);
            inizio = false;
        }
    }
}
```

Interrogazione che trova tutti i progetti e, per ciascuno di essi, gli impiegati coinvolti

Struttura del codice

Interrogazione che trova tutti i progetti e, per ciascuno di essi, gli impiegati coinvolti

```
static ResultSet interrogazioneProgettiImpiegati(Connection con) {
    ResultSet result = null;
    try {
        Statement query = con.createStatement();
        String queryString
                = "select Codice, Titolo, Matricola, Cognome "
                + "from Impiegati right join Progetti  "
                + "  on Progetto = Codice "
                + "order by Codice";
        result = query.executeQuery(queryString);
    } catch (SQLException e) {
        System.out.println("Errore nella interrogazione");
        System.out.println(e.getErrorCode() + " " + e.getSQLState() + e.getMessage());
    }
    return result;
}
```

Struttura del codice

Main:

```
public static void main(String a[]) throws SQLException {
        Connection con = driverEConnessione();
        ResultSet result = interrogazioneProgettiImpiegati(con);
        boolean fineResult = false;
        boolean inizio = true;
        while (ciSonoAncoraProgetti(inizio, fineResult, result)) {
            fineResult = stampaProgettoEImpiegati(result);
            inizio = false;
        }
    }
}
```

Metodo di supporto per verificare se ci sono ancora progetti

Struttura del codice

Metodo di supporto per verificare se ci sono ancora progetti

```
static boolean ciSonoAncoraProgetti(boolean inizio, boolean fineResultSet, ResultSet
result) {
        boolean risposta = false;
        try {
            if (inizio) {
                risposta = result.next();
            } else if (fineResultSet) {
                risposta = false;
            } else {
                risposta = true;
            }
        } catch (SQLException e) {
            System.out.println("Errore nella scansione del risultato");
            System.out.println(e.getErrorCode() + " " + e.getSQLState() + e.getMessage());
        }
        return risposta;
    }
```

Struttura del codice

Main:

```
public static void main(String a[]) throws SQLException {
        Connection con = driverEConnessione();
        ResultSet result = interrogazioneProgettiImpiegati(con);
        boolean fineResult = false;
        boolean inizio = true;
        while (ciSonoAncoraProgetti(inizio, fineResult, result)) {
            fineResult = stampaProgettoEImpiegati(result);
            inizio = false;
        }
    }
}
```

Metodo di supporto per la stampa

Struttura del codice

Metodo di supporto per la stampa

```
static boolean stampaProgettoEImpiegati(ResultSet result) {
        int codiceProgetto, matricola, codiceNuovo;
        String titolo, cognome;
        boolean fineResultSet = false;
        try {
            codiceProgetto = result.getInt("Codice");
            titolo = result.getString("Titolo");
            System.out.println("\nProgetto:" + codiceProgetto + " " + titolo);
            matricola = result.getInt("Matricola");
            cognome = result.getString("Cognome");
            if (result.wasNull()) {
                System.out.println("Nessun impiegato");
                if (!result.next()) {
                    fineResultSet = true;
                }
```

Struttura del codice

Metodo di supporto per la stampa

```
else {
        System.out.println("Impiegati:");
        System.out.println("  " + matricola + " " + cognome);
        boolean fineProgetto = false;
        while (!(fineResultSet) && !(fineProgetto)) {
            if (!result.next()) {
                fineResultSet = true;
            } else {
                codiceNuovo = result.getInt("Codice");
                matricola = result.getInt("Matricola");
                cognome = result.getString("Cognome");
                if (codiceNuovo == codiceProgetto) {
                    System.out.println("  " + matricola + " " + cognome);
                } else {
                    fineProgetto = true;
                }
            }
        }
    }
} catch (SQLException e) {
    System.out.println("Errore nella scansione di result");
    System.out.println(e.getErrorCode() + " " + e.getSQLState()
            + "\n" + e.getMessage());
}
return fineResultSet;   }
```

Query 1.4

- Estendere la risposta al quesito precedente mostrando anche, per ciascun progetto, la lista delle sedi di svolgimento, costruendo quindi un prospetto come il seguente:

```
CodiceProgetto TitoloProgetto
   MatricolaImpiegato CognomeImpiegato
   MatricolaImpiegato CognomeImpiegato
     ... ...
NomeSede Citta
NomeSede Citta
... ...
CodiceProgetto TitoloProgetto
  ... ...
```

Query 1.4

- Possibile soluzione:

```
select Codice, Nome, Citta
from Progetti join Svolgimento on Codice = Progetto
join Sedi on Sede = Nome
order by Codice
```

oppure

```
select Codice, Nome, Citta
from Progetti, Svolgimento, Sedi
where Codice = Progetto and Sede = Nome
order by Codice
```

Query 1.4 in JDBC: struttura del codice

Main:

```
public static void main(String a[]) throws SQLException {
    Connection con = driverEConnessione();
    ResultSet result = interrogazioneProgettiImpiegati(con);
    ResultSet resultSedi = interrogazioneProgettiSedi(con);
    boolean fineResult = false;
    boolean inizio = true;
    while (ciSonoAncoraProgetti(inizio, fineResult, result)) {
        int codiceProgetto = progettoCorrente(result);
        fineResult = stampaProgettoEImpiegati(result);
        inizio = false;
        stampaSedi(codiceProgetto, resultSedi);

    }
}
```

Struttura del codice

Main:

```
public static void main(String a[]) throws SQLException {
    Connection con = driverEConnessione();
    ResultSet result = interrogazioneProgettiImpiegati(con);
    ResultSet resultSedi = interrogazioneProgettiSedi(con);
    boolean fineResult = false;
    boolean inizio = true;
    while (ciSonoAncoraProgetti(inizio, fineResult, result)) {
        int codiceProgetto = progettoCorrente(result);
        fineResult = stampaProgettoEImpiegati(result);
        inizio = false;
        stampaSedi(codiceProgetto, resultSedi);

    }
}
```

Struttura del codice

La lista delle sedi di svolgimento dei progetti:

```
static ResultSet interrogazioneProgettiSedi(Connection con) {
    ResultSet resultSedi = null;
    try {
        Statement querySedi = con.createStatement(
                ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_UPDATABLE);
        String queryStringSedi
                = "select Codice, Nome, Citta "
                + "from  Progetti join Svolgimento on Codice = Progetto  "
                + "  join Sedi on Sede = Nome  "
                + "order by Codice";
        resultSedi = querySedi.executeQuery(queryStringSedi);
    } catch (SQLException e) {
        System.out.println("Errore nella interrogazione");
        System.out.println(e.getErrorCode() + " " + e.getSQLState() + e.getMessage());
    }
    return resultSedi;
}
```

Struttura del codice

Main:

```
public static void main(String a[]) throws SQLException {
    Connection con = driverEConnessione();
    ResultSet result = interrogazioneProgettiImpiegati(con);
    ResultSet resultSedi = interrogazioneProgettiSedi(con);
    boolean fineResult = false;
    boolean inizio = true;
    while (ciSonoAncoraProgetti(inizio, fineResult, result)) {
        int codiceProgetto = progettoCorrente(result);
        fineResult = stampaProgettoEImpiegati(result);
        inizio = false;
        stampaSedi(codiceProgetto, resultSedi);

    }
}
```

Struttura del codice

Metodo che restituisce il progetto corrente:

```
static int progettoCorrente(ResultSet result) {
    int codice = 0;
    try {
        codice = result.getInt("Codice");
    } catch (SQLException e) {
        System.out.println("Errore nella scansione del risultato");
        System.out.println(e.getErrorCode() + " " + e.getSQLState() +
          e.getMessage());
    }

    return codice;
}
```

Struttura del codice

Main:

```
public static void main(String a[]) throws SQLException {
    Connection con = driverEConnessione();
    ResultSet result = interrogazioneProgettiImpiegati(con);
    ResultSet resultSedi = interrogazioneProgettiSedi(con);
    boolean fineResult = false;
    boolean inizio = true;
    while (ciSonoAncoraProgetti(inizio, fineResult, result)) {
        int codiceProgetto = progettoCorrente(result);
        fineResult = stampaProgettoEImpiegati(result);
        inizio = false;
        stampaSedi(codiceProgetto, resultSedi);

    }
}
```

Struttura del codice

Metodo per la stampa delle sedi:

```
static void stampaSedi(int codiceProgetto, ResultSet resultSedi) {
        try {
            if (resultSedi.next()) {
                int codiceQuerySedi = resultSedi.getInt("Codice");
                boolean fineResultSet = false;
                boolean fineProgetto = false;
                if (codiceQuerySedi == codiceProgetto) {
                    System.out.println("Sedi:");
                } else {
                    System.out.println("Nessuna sede");
                }
                while (!(fineResultSet) && !(fineProgetto)) {
                    if (codiceQuerySedi == codiceProgetto) {
                        String nomeSede = resultSedi.getString("Nome");
                        String citta = resultSedi.getString("Citta");
                        System.out.println("  " + nomeSede + " " + citta);
                        if (resultSedi.next()) {
                            codiceQuerySedi = resultSedi.getInt("Codice");
                        } else {
                            fineResultSet = true;
                        }
                    }
```

Struttura del codice

Metodo per la stampa delle sedi:

```
            else {
                    fineProgetto = true;
                    resultSedi.previous();
                }
             };
        } else {
            System.out.println("Nessuna sede");
        }
    } catch (SQLException e) {
        System.out.println("Errore nella scansione di resultSedi");
        System.out.println(e.getErrorCode() + " " + e.getSQLState()
                + "\n" + e.getMessage());
    };
}
```

Domanda 2

Con riferimento ad una relazione

SOCI(Codice, Cognome, Nome, Categoria, Età),

scrivere le interrogazioni SQL che calcolano l'età media dei soci di ciascuna categoria, nei due casi seguenti:

1. se l'età non è nota si usa per essa il valore nullo;
2. se l'età non è nota si usa per essa il valore 0.

Domanda 2

SOCI(Codice, Cognome, Nome, Categoria, Età),

- età media dei soci di ciascuna categoria, se l'età non è nota si usa per essa il valore nullo

```
select Categoria, AVG(Eta) AS EtaMedia
from Soci
group by Categoria
```

Domanda 2

SOCI(Codice, Cognome, Nome, Categoria, Età),

- età media dei soci di ciascuna categoria, se l'età non è nota si usa per essa il valore 0

```
select Categoria, AVG(Eta) AS EtaMedia
from Soci
where Eta <> 0
group by Categoria
```

Domanda 3

Mostrare come in SQL si possa formulare, senza usare il costrutto except, un'interrogazione che calcola la differenza di due relazioni R e S definite entrambe sugli attributi A e B.

```
select *
from R
where not exists (select *
                  from S
                  where S.A = R.A
                  and S.B = R.B)
```

Domanda 4

Considerare le relazioni

DIPARTIMENTI(Codice, Direttore)

IMPIEGATI(Matricola,Nome, Stipendio, Direttore)

e le due interrogazioni seguenti, specificare se e in quali casi esse possono produrre risultati diversi:

Domanda 4

DIPARTIMENTI(Codice, Direttore)
IMPIEGATI(Matricola,Nome, Stipendio, Direttore)

```
SELECT AVG(Stipendio)
FROM Impiegati
WHERE DIRETTORE IN (SELECT Direttore
                    FROM Dipartimenti)
```

Cosa calcola la precedente interrogazione?

Calcola la media degli stipendi degli impiegati il cui direttore dirige un dipartimento

Domanda 4

DIPARTIMENTI(Codice, Direttore)
IMPIEGATI(Matricola,Nome, Stipendio, Direttore)

```
SELECT AVG(Stipendio)
FROM Impiegati I, Dipartimenti D
WHERE I.Direttore = D.Direttore
```

Cosa calcola la precedente interrogazione?

Calcola la media dei valori degli stipendi per le ennuple ottenute concatenando quelle degli impiegati con quelle dei dipartimenti con uguale direttore: un impiegato il cui direttore dirige più dipartimenti compare più volte, modificando potenzialmente la media

Domanda 4

- In altre parole, le due interrogazioni calcolano la media dei valori di stipendio nelle ennuple ottenute in risposta alle due interrogazioni seguenti.
- Nella seconda, un impiegato il cui direttore dirige più dipartimenti compare più volte e quindi il suo stipendio “pesa di più” nel calcolo della media.

Domanda 4

DIPARTIMENTI(Codice, Direttore)
IMPIEGATI(Matricola,Nome, Stipendio, Direttore)

```
SELECT *
FROM Impiegati
WHERE DIRETTORE IN (SELECT Direttore
                    FROM Dipartimenti)
```

Domanda 4

DIPARTIMENTI(Codice, Direttore)
IMPIEGATI(Matricola,Nome, Stipendio, Direttore)

```
SELECT *
FROM Impiegati I, Dipartimenti D
WHERE I.Direttore = D.Direttore
```

Domanda 4

- Quindi i risultati delle due interrogazioni possono differire quando vi sono direttori che dirigono più dipartimenti.