UNIVERSITÀ DEGLI STUDI DI MILANO

CONCORSO PUBBLICO, PER ESAMI, PER IL RECLUTAMENTO DI N. 1 UNITÀ DI PERSONALE AFFERENTE ALL'AREA DEI FUNZIONARI - SETTORE TECNICO-INFORMATICO, CON RAPPORTO DI LAVORO SUBORDINATO A TEMPO INDETERMINATO PRESSO L'UNIVERSITÀ DEGLI STUDI DI MILANO - DIREZIONE ICT - CODICE 22497

La Commissione giudicatrice della selezione, nominata con Determina Direttoriale n.18085 del 31/10/2024, composta da:

Dott.ssa Marialuisa De Francesco	Presidente
Dott. Pasquale Ficara	Componente
Dott. Roberto Sironi	Componente
Sig.ra Suellen Sfragaro	Segretaria

comunica le tracce relative alla prima prova.

TRACCIA n. 1

I/Le candidati/e illustrino le fasi necessarie allo sviluppo di una applicazione web che prevede la gestione totalmente asincrona di tutte le funzionalità.

Si faccia riferimento in particolare alla fase di test e debug.

Usare carattere ARIAL 14, interlinea 1,5, è possibile utilizzare elenchi puntati con trattino.

TRACCIA n. 2

I/Le candidati/e i descrivano come progetterebbero un database relazionale per gestire un sistema di monitoraggio delle richieste di assistenza degli utenti in un contesto operativo di un grande Ateneo.

Usare carattere ARIAL 14, interlinea 1,5, è possibile utilizzare elenchi puntati con trattino.

TRACCIA n. 3

Si deve sviluppare un modulo software web-based per gestire i feedback degli utenti sui servizi digitali di un grande Ateneo. I candidati illustrino come procederebbero nell'analisi dei requisiti e nella pianificazione del testing e nel debugging.

Inoltre, si documenti il processo di sviluppo e le modalità di rilascio e aggiornamento periodico del software.

Usare carattere ARIAL 14, interlinea 1,5, è possibile utilizzare elenchi puntati con trattino.

La Commissione comunica le tracce relative alla seconda prova.

TRACCIA n. 1

Si riporti il risultato di tutte le singole attività in un unico foglio Word di risposta.

1. Si consideri il seguente schema relazionale:

```
STUDENTI (Matricola, NomeS, CognomeS, CittàRes, Sesso, NumTelefono)
CORSI (CodCorso, NomeC, NumCrediti, MatricolaDocente)
ESAMI_SUPERATI (Matricola, CodCorso, Voto)
```

Usando lo schema dati fornito, si risponda alle seguenti domande:

a. Scrivere la query SQL che estrae la matricola degli studenti che hanno superato l'esame di informatica con 30 o quello di fisica con 25;

b. Definire una vista permanente che elenchi il nome e il cognome degli studenti che hanno preso il voto più alto all'esame che ha codice "F3X-A"

2. Si consideri

la relazione **prestito (id, utente, libro, consentito, durata, penale)**

e il vincolo `CHECK(consentito = TRUE AND (durata < 2 OR NOT penale <= 100))`.

Quale fra le seguenti operazioni di inserimento va a buon fine?

A. INSERT INTO prestito VALUES (220, 'Rossi', 'B01', TRUE, 5, 200);

B. INSERT INTO prestito VALUES (240, 'Rossi', 'B02', FALSE, 1, 50);

C. INSERT INTO prestito VALUES (260, 'Rossi', 'B03', TRUE, 5, 50);

D. Nessuna delle precedenti

3. Il candidato, dopo aver attentamente analizzato tutti i passaggi della seguente applicazione ETL in JAVA, spieghi l'obiettivo finale e commenti il codice di ogni passaggio:

```
import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.File;
import java.sql.*;

public class ETLProcess {
    private static final String DB_URL = "jdbc:mysql://localhost:3306/testdb";
    private static final String USER = "username";
    private static final String PASSWORD = "password";

    public static void main(String[] args) {
        try {
            File xmlFile = new File("ordini.xml");
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);

            doc.getDocumentElement().normalize();
            NodeList ordineList = doc.getElementsByTagName("ordine");

            try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASSWORD);
                 PreparedStatement pstmt = conn.prepareStatement(
                     "INSERT INTO ordini (id, cliente, totale) VALUES (?, ?, ?)")) {

                for (int i = 0; i < ordineList.getLength(); i++) {
                    Node node = ordineList.item(i);
                    if (node.getNodeType() == Node.ELEMENT_NODE) {
                        Element element = (Element) node;

                        int id =
Integer.parseInt(element.getElementsByTagName("id").item(0).getTextContent());
                        String cliente =
element.getElementsByTagName("cliente").item(0).getTextContent();
```

```
                        int quantita =
Integer.parseInt(element.getElementsByTagName("quantita").item(0).getTextContent());
                        double prezzo =
Double.parseDouble(element.getElementsByTagName("prezzo").item(0).getTextContent());
                        double totale = quantita * prezzo;

                        pstmt.setInt(1, id);
                        pstmt.setString(2, cliente);
                        pstmt.setDouble(3, totale);
                        pstmt.executeUpdate();
                    }
                }

                System.out.println("Dati caricati con successo");

            } catch (SQLException e) {
                System.err.println("Errore SQL: " + e.getMessage());
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

TRACCIA n. 2

Si riporti il risultato di tutte le singole attività in un unico foglio Word di risposta.

1. Si consideri il seguente schema relazionale:

```
STUDENTI (Matricola, NomeS, CognomeS, CittàRes, Sesso, NumTelefono)
CORSI (CodCorso, NomeC, NumCrediti, MatricolaDocente)
ESAMI_SUPERATI (Matricola, CodCorso, Voto)
```

Usando lo schema dati fornito, si risponda alle seguenti domande:

a. Scrivere la query SQL che estrae la matricola degli studenti che hanno superato l'esame di informatica con 30 e quello di fisica con 25;

b. Definire una vista permanente che elenchi la matricola e la media voti degli studenti che hanno superato almeno 10 esami

2. Si consideri la seguente tabella ORARIO:

TRENO	***DATA***	***ORA***	***DESTINAZIONE***	***DURATA***
AV01	12.07.12	10:00	ROMA	4
AV01	17.07.12	10:00	ROMA	4
AV02	13.07.12	12:00	BOLOGNA	1
AV02	13.07.12	18:00	NAPOLI	5
AV03	13.07.12	16:00	BOLOGNA	1

Indicare il risultato corretto dell'interrogazione:

```
SELECT treno, data, SUM(durata) / COUNT(*)
    FROM orario
GROUP BY treno, data
HAVING COUNT(*) > 1
ORDER BY data.
```

A. <AV01, 12.07.12, 4>, <AV02, 13.07.12, 3>, <AV03, 13.07.12, 1>, <AV01, 17.07.12, 4>

B. <AV01, 12.07.12, 4>, <AV02, 13.07.12, 6>, <AV03, 13.07.12, 1>, <AV01, 17.07.12, 4>

C. <AV02, 13.07.12, 3>

D. Nessuna delle precedenti

---

3. Il candidato, dopo aver attentamente analizzato il codice sorgente della seguente applicazione ETL in JAVA, spieghi l'obiettivo finale e commenti il codice di ogni passaggio.

```
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONArray;
import org.json.JSONObject;

public class ETLProcess {

    private static final String API_URL = "https://jsonplaceholder.typicode.com/users";
    private static final String DB_URL = "jdbc:postgresql://localhost:5432/testdb";
    private static final String USER = "username";
    private static final String PASSWORD = "password";
    private static final Logger LOGGER = Logger.getLogger(ETLProcess.class.getName());

    public static void main(String[] args) {
        HttpClient client = HttpClient.newHttpClient();

        try {
            HttpRequest request = HttpRequest.newBuilder()
                    .uri(URI.create(API_URL))
                    .GET()
                    .build();

            HttpResponse<String> response = client.send(request,
                                                    HttpResponse.BodyHandlers.ofString());

            if (response.statusCode() == 200) {
                LOGGER.info("Dati estratti con successo dall'API");

                JSONArray users = new JSONArray(response.body());
                LOGGER.info("Inizio");

                try (Connection conn = DriverManager.getConnection(DB_URL, USER, PASSWORD);
```

```
                    PreparedStatement pstmt = conn.prepareStatement(
                      "INSERT INTO utenti (id, nome, username, email) VALUES (?, ?, ?, ?)"))
                  {
                    conn.setAutoCommit(false);

                    for (int i = 0; i < users.length(); i++) {
                        JSONObject user = users.getJSONObject(i);

                        int id = user.getInt("id");
                        String nome = user.getString("name").toUpperCase();
                        String username = user.getString("username");
                        String email = user.getString("email");

                        pstmt.setInt(1, id);
                        pstmt.setString(2, nome);
                        pstmt.setString(3, username);
                        pstmt.setString(4, email);
                        pstmt.addBatch();
                    }

                    pstmt.executeBatch();
                    conn.commit();
                    LOGGER.info("Procedura completata con successo");

                } catch (SQLException e) {
                    LOGGER.log(Level.SEVERE, "Errore: ", e);
                }

            } else {
                LOGGER.severe("Errore: " + response.statusCode());
            }

        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Errore durante il processo ETL: ", e);
        }
    }
}
```

TRACCIA 3

Si riporti il risultato di tutte le singole attività in un unico foglio Word di risposta.

1. Si consideri il seguente schema relazionale:

   STUDENTI (Matricola, NomeS, CognomeS, CittàRes, Sesso, NumTelefono)

   CORSI (CodCorso, NomeC, NumCrediti, MatricolaDocente)

   ESAMI_SUPERATI (Matricola, CodCorso, Voto)

   Usando lo schema dati fornito, si risponda alle seguenti domande:

   a. Scrivere la query SQL che estrae la matricola degli studenti che non hanno mai preso 30;
   b. Definire una vista permanente che elenchi il codice corso e il nome corso per cui almeno uno studente ha preso 30

2. Siano date le due tabelle seguenti

clienti		
id	**cognome**	**nome**
1	Bianchi	Aldo
2	Rossi	Aldo
3	Verdi	Aldo

acquisti			
id	**id_cliente**	**articolo**	**prezzo**
1	1	A1	11
2	1	A2	12
3	3	A3	13

ed il costrutto SQL

```
SELECT c.nome, c.cognome
FROM clienti AS c
LEFT OUTER JOIN acquisti AS a ON a.id_cliente = c.id;
```

Indicare il numero di righe prodotte come risultato:

A. 2
B. 3
C. 4

E rappresentare il risultato che ottenuto.

3. Il candidato, dopo aver attentamente analizzato il codice sorgente della seguente applicazione ETL in JAVA, spieghi l'obiettivo finale e commenti il codice di ogni passaggio.

```
import java.io.FileWriter;
import java.io.IOException;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONArray;
import org.json.JSONObject;

public class ETLProcess {

    private static final String REST_ENDPOINT = "https://api.example.com/studenti";
    private static final String CSV_FILE_PATH = "studenti.csv";
    private static final Logger LOGGER = Logger.getLogger(ETLProcess.class.getName());
    private static final HttpClient CLIENT = HttpClient.newHttpClient();
```

```
    public static void main(String[] args) {
        try {

            JSONArray studentiArray = fetchData();

            if (studentiArray != null) {

                writeData(studentiArray);
                LOGGER.info("Dati presenti: " + CSV_FILE_PATH);
            } else {
                LOGGER.warning("Nessun dato presente.”);
            }

        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Errore durante il processo ETL", e);
        }
    }

      private static JSONArray fetchData() {
        try {
            HttpRequest request = HttpRequest.newBuilder()
                    .uri(URI.create(REST_ENDPOINT))
                    .header("Accept", "application/json")
                    .GET()
                    .build();


            HttpResponse<String> response = CLIENT.send(request,
HttpResponse.BodyHandlers.ofString());


            if (response.statusCode() == 200) {

                return new JSONArray(response.body());
            } else {
                LOGGER.warning("Errore: codice di stato " + response.statusCode());
                return null;
            }

        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Errore", e);
            return null;
        }
    }
    private static void writeData(JSONArray studentiArray) {
        try (FileWriter csvWriter = new FileWriter(CSV_FILE_PATH)) {

            csvWriter.append("id,nome,cognome,email\n");


            for (int i = 0; i < studentiArray.length(); i++) {
                JSONObject studente = studentiArray.getJSONObject(i);


                int id = studente.optInt("id", -1);
                String nome = studente.optString("nome", "").toUpperCase();
                String cognome = studente.optString("cognome", "").toUpperCase();
                String email = studente.optString("email", "");


                if (id != -1 && !nome.isEmpty() && !cognome.isEmpty()) {
                    csvWriter.append(id + "," + nome + "," + cognome + "," + email + "\n");
                } else {
```

```
                    LOGGER.warning("Riga ignorata: " + studente);
                }
            }

            csvWriter.flush();

        } catch (IOException e) {
            LOGGER.log(Level.SEVERE, "Errore: ”, e);
        }
    }
}
```

Milano, 11 novembre 2024

La Commissione

Dott.ssa Marialuisa De Francesco - Presidente

Dott. Pasquale Ficara - Componente

Dott. Roberto Sironi - Componente

Sig.ra Suellen Sfragaro - Segretaria