|  |  |  |
| --- | --- | --- |
| C:\Users\Administrateur\Downloads\logo fr .jpg |  | C:\Users\Administrateur\Downloads\logo ar.jpg |
| **Office de la Formation Professionnelle et de la Promotion du Travail** |
| **Direction de la Recherche et de l’Ingénierie de la Formation****Division Conception des Examens** |

**Examen National de Fin d’année**

**Session de Septembre 2020**

**Examen de Fin de Formation (Epreuve de Synthèse)**

**Eléments de réponses**

|  |  |  |  |
| --- | --- | --- | --- |
| **Filière** | **Techniques de Développement Informatique** | **Variante** |  |
|  |  |  |  |
| **Niveau** | **TS** | **Durée** | **4 H** |  |  | **Barème**  | **/100** |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

**Consignes et Conseils:**

* Ce **document** contient des éléments de correction à adapter selon la technologie utilisée : SQL Server ou Oracle, Java ou C #, JavaScript ou jQuery , etc ...
* **Les dossiers 1, 2 et 3 sont OBLIGATOIRES.**
* **Le dossiers 4 et le dossier 5 sont au C H O I X .**

**Détail du Barème :**

|  |  |  |
| --- | --- | --- |
| **N° Des Dossiers** | **Travaux à réaliser** | **Barème** |
| **Partie Théorique** |
| **Dossier 1** | Base de données (les bases) | **30 points** |
| **Dossier 2**  | Programmation Orientée Objet | **10 points** |
|  |  |  |
| ***Total*** | **/40 points** |
| **Partie Pratique** |
| **Dossier 3** | Base de données  (requête /procédures/Triggers) | **30 points** |
| **Dossier 4** | Développement Web coté client | **30 points** |
| **Dossier 5** | Développement client-serveur | **30 points** |
|  |  |  |
| ***Total*** | **/60 points** |
| **Total Général**  | **/100 points** |

**Partie Théorique**

**Dossier 1 : (30 pts)**

***Questions 1 et 2 : Voir cours.***

1. Contraintes d’intégrité référentielle :

La colonne idV de la table Participant - La colonne idP de la table Blague - La colonne idBlague de la table Vote.

1. Non : violation contrainte clé étrangère.
2. Non : violation contrainte clé primaire.
3. Oui : la colonne idP de la table Blague est seulement une clé étrangère.
4. Proposer un modèle conceptuel de données (MCD) correspondant au schéma relationnel ci-dessus.

Participant

idP

nom

prenom

civilite

tel

Ville

idV

libelleV

habite a

participer

concerner

Vote

idsession

date\_vote

Blague

idBlague

titre

contenu

nombrevote

**Dossier 2 : (10 pts)**

**Questions 1) , 2) , 3) et 4) :**

class Participant

 {

 private int idP ;

 private string nom;

 private string prenom;

 private string tel;

 public Participant()

 { }

 public Participant(int id, string nom , string pren, string tel)

 {

 if (!tel.All(char.IsDigit) || (tel.Length != 10) || (!tel.StartsWith("0")) )

 throw new Exception("N° de tél invalide");

 idP = id;

 this.nom = nom;

 prenom = pren;

 this.tel = tel;

 }

 public int IdP

 {

 get

 {

 return idP;

 }

 set

 {

 idP = value;

 }

 }

 public string Nom

 {

 get

 {

 return nom;

 }

 set

 {

 nom = value;

 }

 }

 public string Prenom

 {

 get

 {

 return prenom;

 }

 set

 {

 prenom = value;

 }

 }

 public string Tel

 {

 get

 {

 return tel;

 }

 set

 {

 if (!value.All(char.IsDigit) || (value.Length != 10) || (!value.StartsWith("0")))

 throw new Exception("N° de tél invalide");

 tel = value;

 }

 }

 public string nomComplet()

 {

 string s = prenom.Substring(0, 1).ToUpper() + "." + nom;

 return s;

 }

 }

**Partie Pratique**

**Dossier 3 : (30 pts)**

1. Ecrire une requête pour supprimer toutes les blagues ayant moins de 3 votes :

*Delete from blague where idblague in (Select idblague from vote group by idblague having count(\*)<3) ;*

1. Ecrire une requête qui affiche le nom et prénom de tous les participants de « Fes » :

*Select nom, prenom from participant join ville using(idv) where libellev=’Fes’ ;*

1. Ecrire une requête qui affiche le nombre de blagues par participant. On afficher le nom et prénom ainsi que le nombre de blagues. On retient seulement ceux ayant plus de 2 blagues :

*Select nom, prenom, count(\*) nbrblague from blague join participant using(idp)*

*Group by nom, prenom*

*Having count(\*)>2 ;*

1. Ecrire une requête qui affiche le nom, prénom et le nombre de vote pour chaque blague des trois participants ayant obtenu les trois premières places dans le classement. N’utiliser pas le champ *nombrevote*, recalculer le :

*Select \**

*From ( Select nom, prenom, count(\*) nbrvotes from blague join participant using(idp) join vote using (idblague)*

*Group by nom, prenom*

*Order by nbrvotes )*

*Where rownum <=3;*

1. Ecrire une procédure stockée permettant de renvoyer à l’aide d’un paramètre de sortie le nombre de vote obtenu par un participant dont l’id est passé en paramètre :

*Create or replace procedure getNbrVotes(p\_idp IN number, nbrvote OUT number)*

*Is*

*Begin*

*Select count(\*) int nbrvotes from blague join participant using(idp) join vote using (idblague) where idp=p\_idp ;*

*End ;*

**/**

1. Ecrire un trigger permettant d’incrémenter le champ *nombrevote* de la table Blague suite à chaque nouveau vote :

*Create or replace trigger upd\_nbrvote*

*After insert on vote*

*For each row*

*Begin*

*Update blague set nombrevote=nombrevote+1 where idblague= :new.idb ;*

*End ;*

***/***

**Dossier 4 : (30 pts)**

1. Proposer un schéma XSD :

<?xml version="1.0" encoding="utf-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

 <xs:element name="FestivaleRire">

 <xs:complexType>

 <xs:sequence>

 <xs:element name="participant" maxOccurs="unbounded" >

 <xs:complexType>

 <xs:sequence>

 <xs:element name="nom" type="xs:string" />

 <xs:element name="prenom" type="xs:string" />

 <xs:element name="civilite" type="xs:string" />

 <xs:element name="tel" type="xs:integer" />

 <xs:element name="ville">

 <xs:complexType>

 <xs:sequence>

 <xs:element name="idv" type="xs:integer" />

 <xs:element name="libellev" type="xs:string" />

 </xs:sequence>

 </xs:complexType>

 </xs:element>

 <xs:element maxOccurs="unbounded" name="blague" maxOccurs="unbounded" >

 <xs:complexType>

 <xs:sequence>

 <xs:element name="titre" type="xs:string" />

 <xs:element name="contenu" type="xs:string" />

 <xs:element name="nbrvote" type="xs:integer" />

 </xs:sequence>

 <xs:attribute name="idb" type="xs:integer" />

 </xs:complexType>

 </xs:element>

 </xs:sequence>

 <xs:attribute name="idp" type="xs:integer" />

 </xs:complexType>

 </xs:element>

 </xs:sequence>

 </xs:complexType>

 </xs:element>

</xs:schema>

1. //participant[ idp = ‘2’]/blague
2. count( //participant/ville[ libellev = ‘Rabat’])
3. Formulaire HTML permettant à un internaute de s’inscrire afin de participer avec une blague :

<div >

<h3> Inscription: </h3>

<fieldset>

<form name="mon-formulaire1" action="" method="get">

<p>

 <input type="radio" name="civi" value="Mme" /> Madame

 <input type="radio" name="civi" value="Mlle" /> Mademoiselle

 <input type="radio" name="civi" value="Mr" /> Monsieur

</p>

<table border-width="0px">

<tr><td> Votre prenom :<span>\*</span></td><td>

 <input type="text" name="prenom" value="" required/>

<tr><td> Votre nom :<span>\*</span></td><td>

 <input type="text" id="nom" value="" required/>

</td></tr>

<tr><td>

 Telephone :<span>\*</span></td><td>

 <input type="tel" id="tel" value="" required/>

</td></tr>

<tr><td> Ville <span>\*</span></td><td>

 <select id="ville" required>

 <option value="parti" selected="selected">--- Choisissez une ville ---</option>

 <option value="1" >Casablanca</option>

 <option value="2">Rabat</option>

 <option value="3">Fes</option>

 <option value="4">Marrakech</option>

 </select>

</td></tr>

<tr><td> Intituler de votre blague : <span>\*</span></td><td>

 <input type="text" name="intitule" value="" required/>

</td></tr>

</table>

 Votre Blague <span>\*</span> :<br />

 <textarea name="le-message" rows="6" cols="40" required></textarea></br>

<p>Quel est le resultat de l'expression :

<span id="robot"></span><input type="text" id="reponse" size="5">

</p>

<p> <input type="button" value="Sauvegarder" onclick="sauvegarder();"/>

 <input type="reset" value="Annuler" />

</p>

</form>

</div>

</h4>

</div>

***Questions 5) et 6) :***

<script>

 var min=1;

 var max=10;

 var random1 = Math.floor(Math.random() \* (max - min)) + min;

 var random2 = Math.floor(Math.random() \* (max - min)) + min;

 document.getElementById("robot").innerHTML=random1+'+'+random2+'=';

 function sauvegarder(){

 var res=random1+random2;

 if(res!=document.getElementById("reponse").value){

 alert("reponse incorrecte");

 return false;

 }

 var tel=document.getElementById("tel").value;

 var nom=document.getElementById("nom").value;

 var prenom=document.getElementById("prenom").value;

 var ville=ville.options[ville.selectedIndex];

 var data = nom+" "+prenom+" "+ville+" "+civilite ;

 alert (data) ;

 }

</script>

**Dossier 4 : (30pts)**

SqlConnection cx;

 string sc = "data source=.; initial catalog=BD\_Festival; integrated security=yes";

 public Form1()

 {

 InitializeComponent();

 }

**Question 1):**

private void btAj\_Click(object sender, EventArgs e)

 {

 cx = new SqlConnection(sc);

 string req = "insert into Participant (idP,nom,prenom,Tel,idV) values (@idp,@nom,@prenom,@tel,@idv)";

 SqlCommand cmd = new SqlCommand(req, cx);

 cmd.Parameters.AddWithValue("@idp", t1.Text);

 cmd.Parameters.AddWithValue("@nom", t2.Text);

 cmd.Parameters.AddWithValue("@prenom", t3.Text);

 cmd.Parameters.AddWithValue("@tel", t4.Text);

 cmd.Parameters.AddWithValue("@idv", t5.Text);

 cx.Open();

 cmd.ExecuteNonQuery();

 cx.Close();

 }

 private void btSuppr\_Click(object sender, EventArgs e)

 {

 cx = new SqlConnection(sc);

 string req = "Delete Participant Where idP=@idp";

 SqlCommand cmd = new SqlCommand(req, cx);

 cmd.Parameters.AddWithValue("@idp", t1.Text);

 cx.Open();

 cmd.ExecuteNonQuery();

 cx.Close();

 }

 **Question 2):**

private void btXML\_Click(object sender, EventArgs e)

 {

 cx = new SqlConnection(sc);

 string req = "Select \* From Participant";

 SqlCommand cmd = new SqlCommand(req, cx);

 DataTable t = new DataTable();

 SqlDataReader r;

 cx.Open();

 r = cmd.ExecuteReader() ;

 t.Load(r);

 t.WriteXml("participants.xml");

 cx.Close();

 }

**Question 3):**

void AjouterBlague(int idBlague, int idP, string intitule, string contenu)

 {

 cx = new SqlConnection(sc);

 string req= "Select count(\*) From Participant Where idP=@idp";

 SqlCommand cmd = new SqlCommand(req, cx);

 cmd.Parameters.AddWithValue("@idp", idP);

 cx.Open();

 int n = (int)cmd.ExecuteScalar();

 cx.Close();

 if (n == 0)

 throw new Exception("Participant inexistant");

 req = "insert into Blague (idBlague,idP,intitule,contenu) values (@idblague,@idP,@intitule,@contenu)";

 cmd = new SqlCommand(req, cx);

 cmd.Parameters.AddWithValue("@idblague", idBlague);

 cmd.Parameters.AddWithValue("@idP", idP);

 cmd.Parameters.AddWithValue("@intitule", intitule);

 cmd.Parameters.AddWithValue("@contenu", contenu);

 cx.Open();

 cmd.ExecuteNonQuery();

 cx.Close();

 }

**Question 4):**

 public void SupprimerBlagues(int idP)

 {

 cx = new SqlConnection(sc);

 string req = "Delete Blague Where idP=@idp";

 SqlCommand cmd = new SqlCommand(req, cx);

 cmd.Parameters.AddWithValue("@idp", idP);

 cx.Open();

 cmd.ExecuteNonQuery();

 cx.Close();

 }

**Question 5):**

public int Votes(DateTime DateDebut, DateTime DateFin)

 {

 cx = new SqlConnection(sc);

 string req = "Select count(\*) From Votes Where date\_vote Between @dd And @df ";

 SqlCommand cmd = new SqlCommand(req, cx);

 cmd.Parameters.AddWithValue("@dd", DateDebut);

 cmd.Parameters.AddWithValue("@df", DateFin);

 cx.Open();

 int n = (int)cmd.ExecuteScalar();

 cx.Close();

 return n;

 }

**Question 6):**

 public List<int> lesBlagues()

 {

 List<int> li = new List<int>();

 cx = new SqlConnection(sc);

 string req = "Select top 3 idBlague From Votes V inner join Blague B on V.idBlague=B.idBlague Order By B.nbrVote Desc";

 SqlCommand cmd = new SqlCommand(req, cx);

 cx.Open();

 SqlDataReader r = cmd.ExecuteReader();

 while (r.Read())

 {

 li.Add((int)r["idBlague"]);

 }

 cx.Close();

 return li;

 }