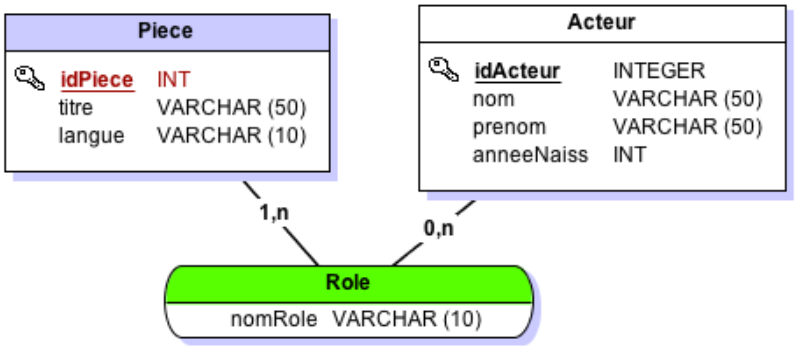
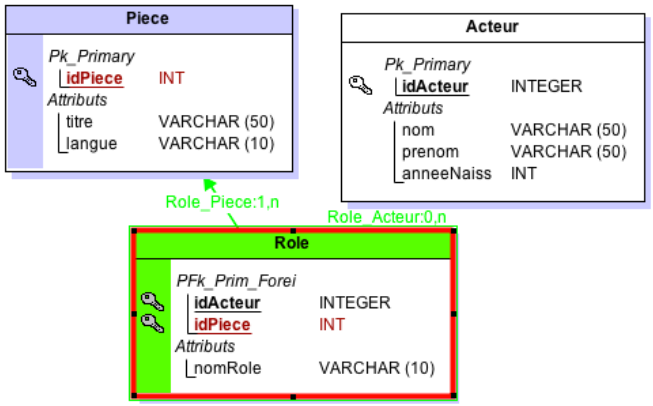
SUJET METROPOLE 1

On peut reproduire le MCD dans JMerise:



On obtient les tables:



1. Si ***Piece*** et ***Acteur*** sont vides, il n'est pas possible d'insérer une entrée dans la relation ***Role*** car les valeurs des clés étrangères ***idPiece*** et ***idRole*** *ne sont pas définies. Le SGBD retournera donc une erreur.*

2. Pour ajouter ce rôle, on doit écrire la requête suivante :

```
INSERT INTO Role VALUES(46721, 389761, 'Tartuffe')
```

3. La requête SQL ci-dessous a pour but d'actualiser la colonne ***Langue*** en ajoutant la valeur '***Anglais***' pour tous les enregistrements dont l'attribut ***Langue*** a pour valeur '***Britannique***' ou '***Américain***'

Requête SQL étudiée :

```
UPDATE Piece
SET langue = "Anglais"
WHERE langue = "Américain" OR langue = "Britannique";
```

4.

a.
```
SELECT nom, prenom FROM Acteur WHERE anneeNaiss > 1990
```

b.
```
SELECT anneeNaiss FROM Acteur ORDER BY anneeNaiss DESC LIMIT 1
```

ou mieux:

```
SELECT MAX(anneeNaiss) FROM Acteur
```

c.
```
SELECT Role.nomRole FROM Role
INNER JOIN Acteur
ON Role.idActeur = Acteur.idActeur
WHERE Acteur.prenom = 'Vincent'
AND Acteur.nom = 'Macaigne'
```

ou avec seulement la clause WHERE:

```
SELECT Role.nomRole FROM Role, Acteur
WHERE Acteur.prenom = 'Vincent' AND Acteur.nom =
'Macaigne' AND Role.idActeur = Acteur.idActeur
```

d.
```
SELECT Piece.titre FROM Piece
INNER JOIN Role
ON Role.idPiece = Piece.idPiece
INNER JOIN Acteur
ON Role.idActeur = Acteur.idActeur
WHERE Acteur.prenom = 'Jeanne'
AND Acteur.nom = 'Balibar'
```

ou avec seulement la clause WHERE:

```
SELECT Piece.titre FROM Piece, Acteur, Role
WHERE Piece.langue = 'Russe' AND Acteur.prenom =
'Jeanne' AND Acteur.nom = 'Balibar' AND
Role.idActeur = Acteur.idActeur AND Role.idPiece =
Piece.idPiece
```

SUJET METROPOLE 2

1. Le code SQL ci-dessous provoque une erreur car le premier et le dernier enregistrement ont une **même clé primaire**. Cela viole la contrainte d´intégrité.

Requête SQL étudiée :

```
INSERT INTO Eleves VALUES (128, 'Dupont', 'Jean', 'T1') ;
INSERT INTO Eleves VALUES (200, 'Dupont', 'Jean', 'T1') ;
INSERT INTO Eleves VALUES (128, 'Dubois', 'Jean', 'T2')
;
```

2. Dans la définition de la relation ***Emprunts***, ce qui assure qu'on ne peut pas enregistrer un emprunt pour un élève qui n'a pas encore été inscrit dans la relation ***Eleves*** est la clé étrangère ***idEleve*** qui doit d'abord être définie par la clé primaire ***idEleve*** de la relation ***Eleves***.

3. **SELECT titre FROM Livre WHERE auteur = 'Molière'**

4. La requête SQL ci-dessous compte le nombre d'élèves enregistrés dans la classe T2.

Requête SQL étudiée :

```
SELECT COUNT(*)
FROM Eleves
WHERE classe = 'T2' ;
```

5.

```
UPDATE Emprunts SET dateRetour = '2020-09-30'
WHERE idEmprunt = 640 ;
```

6. La requête ci-dessous donne les noms et prénoms des élèves enregistrés dans la classe de T2 ayant emprunté un livre au CDI.

Requête SQL étudiée :

```
SELECT DISTINCT nom, prenom
FROM Élèves, Emprunts
WHERE Eleves.idEleve = Emprunts.idEleve
AND Eleves.classe = 'T2' ;
```

7. ***Il faut joindre les trois tables:***

```
SELECT Eleves.nom, Eleves.prenom FROM Eleves
INNER JOIN Emprunts
ON Eleves.idEleve = Emprunts.idEleve
INNER JOIN Livres
ON Livres.isbn = Emprunts.isbn
WHERE Livres.titre = 'Les misérables'
```

... ou avec seulement la clause WHERE:

```
SELECT Eleves.nom, Eleves.prenom
FROM Eleves, Emprunts, Livres
WHERE Livres.titre = 'Les misérables'
AND Livres.isbn = Emprunts.isbn
AND Emprunts.idEleve = Eleves.idEleve
```

POLYNESIE

PARTIE A

1. On donne les clés primaires des relations :

   Clients : ***idClient***
   Articles : idArticle

2. Le domaine des attributs :

   Email : **Varchar(50) --→ TEXTE de 50 caractères maximum**

   Quantite : **INT --→ ENTIER**

3.

```
CREATE TABLE Commandes (

IdCmd INT PRIMARY KEY,

IdClient INT,

Date DATE,

AdresseLivraison VARCHAR(90),

PaiementValide BOOLEAN,

LivraisonFaite BOOLEAN,

FOREIGN KEY(IdClient) REFERENCES Clients (IdClient)

);
```

PARTIE B

1. Lorsqu'un formulaire contient une quantité importante de données, il est préférable d'utiliser la méthode POST car les informations du formulaire à envoyer passent directement à l'intérieur la requête HTTP et **non à l'intérieur de l'URL**. Les navigateurs ont une taille limitée de caractères dans la barre d´adresse (8000 actuellement pour CHROME).

2. Pour la validation du paiement, il est préférable d'utiliser le protocole HTTPS car il est e**ntièrement chiffré** et donc bien plus sécurisé que le protocole HTTP.

3. Avant la validation du formulaire, l'intérêt de vérifier le format des informations saisies est de vérifier que les informations saisies sont cohérentes et logiques pour éviter de possibles bugs ou erreurs.

PARTIE C

1.

```
SELECT IdArticle, Libelle FROM Articles WHERE PrixEnCentimes < 1500
```

2. La requête ci-dessous sélectionne tous les identifiants des clients, leurs mails, les identifiants de leurs commandes et leurs adresses de livraison quand les paiements sont invalides.

Requête SQL étudiée :

```
SELECT u.IdClient, u.Email, v.IdCmd, v.AdresseLivraison
FROM Clients as u JOIN Commandes as v
ON u.IdClient = v.IdClient
WHERE v.PaiementValide = False;
```

3. Il faut à nouveau joindre trois tables:

```
SELECT Articles.Libelle FROM Articles
INNER JOIN ArticlesCommande
ON Articles.IdArticle = ArticlesCommande.IdArticle
INNER JOIN Commandes
ON ArticlesCommande.IdCmd = Commandes.IdCmd
WHERE Commandes.IdCmd = 1345
```

... ou avec seulement la clause WHERE:

```
SELECT Articles.Libelle
FROM Articles, ArticlesCommande, Commandes
WHERE Commandes.IdCmd = 1345
AND ArticlesCommande.IdCmd = Commandes.IdCmd
AND Articles.IdArticle = ArticlesCommande.IdArticle
```

4.

```
INSERT INTO Articles (Libelle, Description, PrixEnCentimes)
VALUES ('Imperméable', 'Cet imperméable se replie en forme de pochette.', 999)
```