Bases De Données Relationnelles

Promo 2014

Durée : 1h30

Remarques :

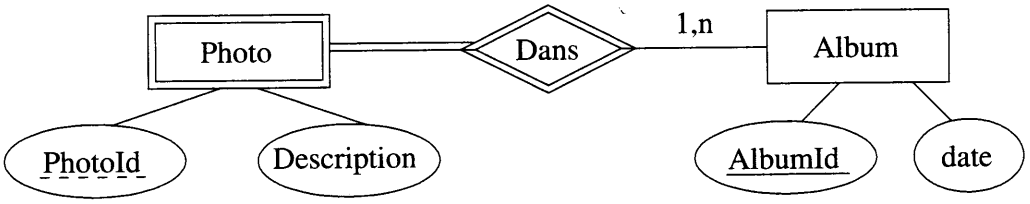
- Documents non autorisés.
- Les questions peuvent avoir plusieurs réponses possibles : il y a toujours au moins une réponse correcte et au moins une réponse fausse.
- Une bonne réponse est notée 1pt, une mauvaise réponse est notée -0.5 et une réponse incomplète 0.5.
- Donner vos réponses sur les feuilles prévues pour les QCMs.

Modélisation

1. Dans le diagramme E/A, une entité est représentée par :
   a) un rectangle
   b) un losange
   c) une ellipse
   d) deux rectangles emboités

2. Une entité faible est :
   a) Une entité qui ne possède pas d'attributs.
   b) Une entité qui ne participe pas aux autres associations du schéma.
   c) Une entité qui est identifiée par le rôle qu'elle joue dans une association.
   d) Une entité dont tous ses attributs sont facultatifs.

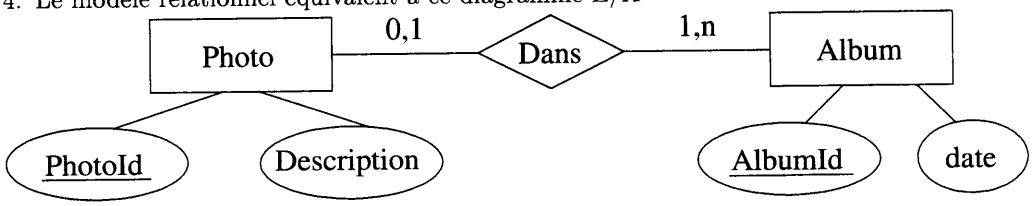
3. Le modèle relationnel équivalent à ce diagramme E/A



est :

a) Photo(PhotoId, description) ; Dans(PhotoId, AlbumId) ; Album(AlbumId, date)
b) Photo(AlbumId, Photoid, description) ; Album(AlbumId, date)
c) Photo(PhotoId, AlbumId, date, description)
d) Photo(PhotoId, AlbumId, description, date)
e) Photo(PhotoId, AlbumId, description) ; Album(AlbumId, date)

4. Le modèle relationnel équivalent à ce diagramme E/A



est :

a) Photo(PhotoId, description) ; Dans(PhotoId, AlbumId) ; Album(AlbumId, date)

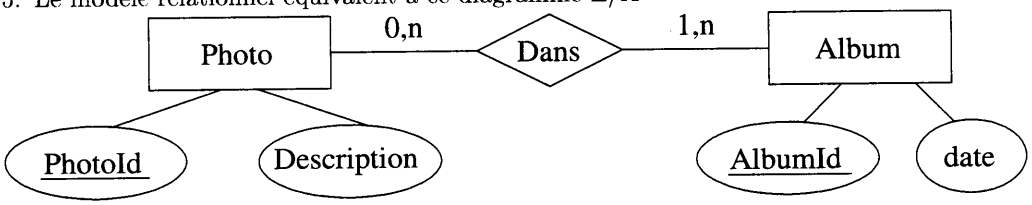
b) Photo(AlbumId, Photoid, description) ; Album(AlbumId, date)

c) Photo(PhotoId, AlbumId, date, description)

d) Photo(PhotoId, AlbumId, description, date)

e) Photo(PhotoId, AlbumId, description) ; Album(AlbumId, date)

5. Le modèle relationnel équivalent à ce diagramme E/A



est :

a) Photo(PhotoId, description) ; Dans(PhotoId, AlbumId) ; Album(AlbumId, date)

b) Photo(AlbumId, Photoid, description) ; Album(AlbumId, date)

c) Photo(PhotoId, AlbumId, date, description)

d) Photo(PhotoId, AlbumId, description, date)

e) Photo(PhotoId, AlbumId, description) ; Album(AlbumId, date)

6. Dans le cas de l'héritage pour le diagramme E/A :

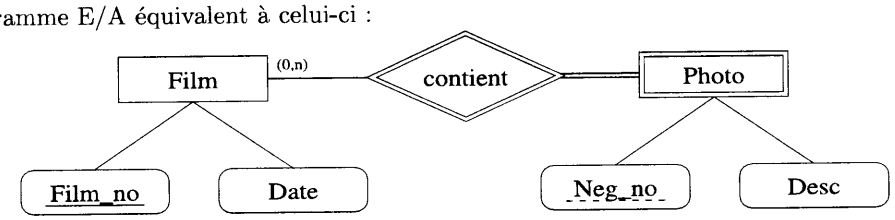
a) Les sous entités ont leur propre identifiant.

b) Les sous entités héritent, en plus des attributs normaux, de l'identifiant de l'entité de base.

c) Les sous entités n'héritent pas de tous les attributs mais uniquement de la clé.

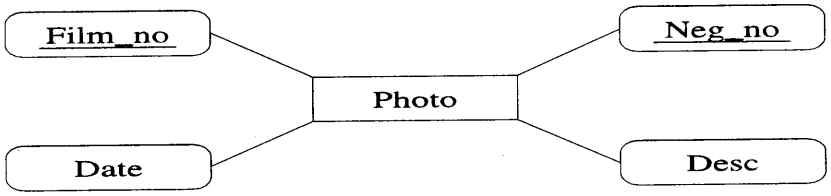
d) Les sous entités héritent de tous les attributs sauf la clé.

7. Le diagramme E/A équivalent à celui-ci :

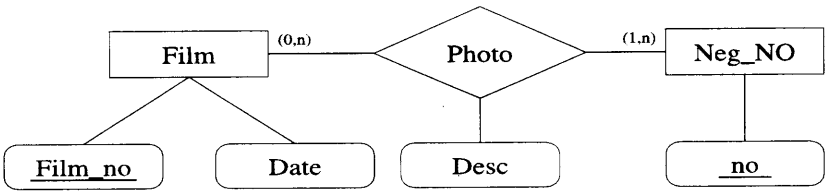


est :

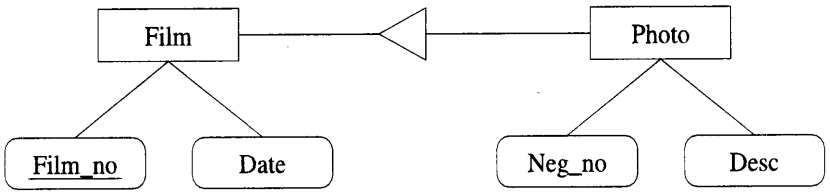
a)



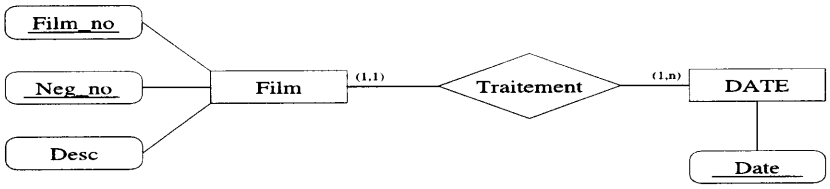
b)



c)



d)



8. Dans le modèle relationnel, un attribut clé étrangère est :
   a) Un attribut qui figure dans plusieurs tables.
   b) Un attribut qui est la clé primaire d'une autre table.
   c) Une partie de la clé primaire de la table.
   d) Toutes les réponses sont fausses.

9. Une clé étrangère est définie en SQL en utilisant le mot clé
   a) PRIMARY KEY
   b) FOREIGN KEY
   c) UNIQUE KEY
   d) OUTER KEY

10. Dans le modèle relationnel :
    a) Une table a une clé primaire unique.
    b) La clé primaire doit être composée d'un seul attribut.
    c) Une table peut avoir plusieurs clés primaires.
    d) Une table doit avoir au moins une clé étrangère.

Formes Normales

11. Soit le schéma relationnel $S = \{R(A, B, C, D, E)\}$ sur lequel on a défini une famille F de dépendances fonctionnelles :

$$F = \{DE \rightarrow C, C \rightarrow B, B \rightarrow E\}$$

En quelle forme normale est S ?

a) 1FN
b) 2FN
c) 3FN
d) BCNF

12. Soit la décomposition de S en

$$S_1 = \{R_1(A, D, E), R_2(D, C, E), R_3(B, C), R_4(B, E)\}$$

Cette décomposition est :
a) Avec perte d'informations.
b) Sans perte d'informations.
c) Avec perte de dépendances
d) Sans perte de dépendances.

13. En quelle forme normale est S_1 ?
a) 1FN
b) 2FN
c) 3FN
d) BCNF

14. Soit la décomposition de S en

$$S_2 = \{R_1(A, D, C), R_2(C, E), R_3(B, C), R_4(B, E)\}$$

Cette décomposition est :
a) Avec perte d'informations.
b) Sans perte d'informations.
c) Avec perte de dépendances
d) Sans perte de dépendances.

15. En quelle forme normale est S_2 ?
a) 1FN
b) 2FN
c) 3FN
d) BCNF

SQL

16. Soit le modèle relationnel suivant :

Cmd(<u>nclt, nprod, datecmd</u>, qte)

Le script correct pour la création de cette table en langage SQL est :

a)
```
CREATE TABLE Cmd (
    nclt INT NOT NULL PRIMARY KEY,
    nprod INT NOT NULL PRIMARY KEY,
    datecmd DATE NOT NULL PRIMARY KEY,
    qte INT DEFAULT 1 CHECK(Qte > 0)
    )
```

b)
```
CREATE TABLE Cmd (
    nclt INT CONSTRAINT PK_Cmd PRIMARY KEY,
    nprod INT CONSTRAINT PK_Cmd PRIMARY KEY,
    datecmd DATE CONSTRAINT PK_Cmd PRIMARY KEY,
    qte INT DEFAULT 1 CHECK(Qte > 0)
    )
```

c)
```
CREATE TABLE Cmd (
    nclt INT NOT NULL CONSTRAINT PK_Cmd UNIQUE,
    nprod INT NOT NULL CONSTRAINT PK_Cmd UNIQUE,
    datecmd DATE NOT NULL CONSTRAINT PK_Cmd UNIQUE,
    qte INT DEFAULT 1 CHECK(Qte > 0)
    )
```

d)
```
CREATE TABLE Cmd (
    nclt INT,
    nprod INT,
    datecmd DATE,
    qte INT DEFAULT 1 CHECK(qte > 0),
    CONSTRAINT PK_Cmd PRIMARY KEY(nclt, nprod, jour)
    )
```

17. Soit la script SQL suivant :

```
CREATE TABLE Etudiant
(
uid INT NOT NULL PRIMARY KEY,
login CHAR(8) UNIQUE,
nom VARCHAR(20),
prenom VARCHAR(50),
idpromo INT NOT NULL CONSTRAINT FK_Etudiant_Promo REFERENCES Promo(idpromo)
)
```

Quelles sont les définitions correctes de la table `Promo` qui ne vont pas générer des erreurs au moment de la création de la table `Etudiant` ?

a)
```
CREATE TABLE Promo
    (
    num INT PRIMARY KEY,
    idpromo INT UNIQUE,
    parrain VARCHAR(20) NOT NULL
    )
```

b)
```
CREATE TABLE Promo
    (
    num INT PRIMARY KEY,
    idpromo INT,
    parrain VARCHAR(20) NOT NULL
    )
```

c)
```
CREATE TABLE Promo
    (
    num INT UNIQUE,
    idpromo INT PRIMARY KEY,
    parrain VARCHAR(20) NOT NULL
    )
```

d) Toutes les réponses sont fausses.

18. L'insertion d'un nouveau tuple dans la table `Etudiant`

    a) sera validé pour toute les valeurs possibles du tuple.

    b) sera validé dans le cas où la valeur de l'attribut `idpromo` est `NULL`

    c) sera invalidé dans le cas où aucun tuple de la table `Promo` ne correspond à la valeur de l'attribut `idpromo`.

    d) peut provoquer l'insertion automatique d'un autre tuple dans la table `Promo` si aucun tuple de cette table ne correspond à la valeur de l'attribut `idpromo`.

19. Soit le script de déclaration des deux tables `Etudiant` et `Groupe`

```
CREATE TABLE Etudiant
(
uid INT NOT NULL PRIMARY KEY,
login CHAR(8) UNIQUE,
nom VARCHAR(20),
prenom VARCHAR(50),
idgroupe INT NOT NULL;
);

CREATE TABLE Groupe
(
idgroupe INT PRIMARY KEY,
parrain VARCHAR(20) NOT NULL,
delegues INT NOT NULL CONSTRAINT FK_Groupe_Etudiant REFERENCES Etudiant(uid)
);

ALTER TABLE Etudiant
ADD CONSTRAINT FK_Etudiant_Groupe FOREIGN KEY (idgroupe) REFERENCES Groupe(idgroupe);
```

    a) On ne pourra jamais insérer un tuple dans ces deux tables.

    b) On doit commencer par insérer le groupe avant d'insérer le délégué correspondant.

    c) La contrainte `FK_Etudiant_Groupe` ne doit pas être rajoutée car elle rend la base inutilisable.

    d) Toutes les réponses sont fausses.

20. Soit le script SQL suivant :

```
CREATE TABLE Etudiant
(
uid INT NOT NULL PRIMARY KEY,
login CHAR(8) UNIQUE,
idpromo INT NOT NULL CONSTRAINT FK_Etd_Promo REFERENCES Promo(idpromo)
)
```

    L'attribut `login` doit vérifier la contrainte suivante :

    a) Plusieurs tuples de la table peuvent avoir la valeur `NULL` dans la colonne `login`

    b) Un seul tuple de la table peut avoir la valeur `NULL` dans la colonne `login`

    c) Deux tuples différents de la table `Etd` ne peuvent pas avoir la même valeur non `NULL` dans l'attribut `login`.

    d) Toutes les réponses sont fausses

21. Pour supprimer une table `A`, on utilise la requête :

a) `DELETE TABLE A`

b) `DROP TABLE A`

c) `ALTER TABLE A`

d) `REVOKE TABLE A`

22. La requête `DELETE FROM A`

a) supprime la table `A`

b) supprime tous les tuples de la table `A`

c) ne supprime aucun tuple de la table `A`

d) toutes les réponses sont fausses

SQL2

On utilisera dans cette partie la base de données suivante :

```
CREATE TABLE Client
(
nclt INT NOT NULL PRIMARY KEY,
nom VARCHAR(10) NOT NULL,
age INT,
codepostal INT
);

CREATE TABLE Produit
(
nprod INT NOT NULL PRIMARY KEY,
type VARCHAR(20) NOT NULL,
couleur VARCHAR(20)
);

CREATE TABLE Cmd
(
nclt INT REFERENCES Client(nclt),
nprod INT REFERENCES Produit(nprod),
datecmd DATE NOT NULL,
qte INT DEFAULT 1
CONSTRAINT PK_Cmd PRIMARY KEY (nclt, nprod, datecmd)
);
```

23. Pour afficher les Clients habitant à Paris, on utilisera :

a)
```
SELECT *
    FROM Client
    WHERE codepostal IN 'paris'
```

b)
```
SELECT *
    FROM Client
    WHERE codepostal BETWEEN 75000 AND 75999
```

c)
```
SELECT *
    FROM Client
    WHERE codepostal LIKE '75___'
```

d)
```
SELECT *
    FROM Client
    WHERE codepostal LIKE '75%'
```

24. Soit la requête SQL suivante :

```
SELECT *
FROM Client
WHERE AGE IN (13, 20, NULL)
```

La clause `WHERE` est :

a) Vraie si `age=13`

b) Fausse si `age=13`

c) Vraie si `age=NULL`

d) Fausse si `age=NULL`

25. la requête SQL suivante :

```
SELECT *
FROM Client
WHERE AGE = (SELECT MAX(age)
             FROM Client)
```

est équivalente à :

a)
```
SELECT *
FROM Client
WHERE AGE >= ALL(SELECT age
                 FROM Client)
```

b)
```
SELECT *
FROM Client
WHERE AGE >= ANY(SELECT age
                 FROM Client)
```

c)
```
SELECT *
FROM Client
WHERE AGE >= ALL(SELECT age
                 FROM Client
                 WHERE AGE IS NOT NULL)
```

d)
```
SELECT *
FROM Client
WHERE AGE >= ANY(SELECT age
                 FROM Client
                 WHERE AGE IS NOT NULL)
```

e) Toutes les réponses sont fausses.

26. Quel est le résultat de la requête suivante :

```
SELECT COUNT(*)
FROM Client
WHERE age = NULL
```

a) 0

b) 1

c) NULL

d) dépend du contenu de la colonne **age** de la table `client`

27. En langage SQL :

a) Dans la condition de jointure, on doit utiliser uniquement les attributs clés primaires ou les attributs clés étrangères des tables utilisées.

b) Dans la condition de jointure, on doit utiliser uniquement les attributs clés primaires des tables utilisées.

c) Dans la condition de jointure, on doit utiliser uniquement les attributs clés étrangères des tables utilisées

d) Toutes les réponses sont fausses

28. Quelles sont les requêtes équivalentes :

a)
```
SELECT *
FROM CLIENT
WHERE nclt IN (SELECT nclt
               FROM CMD
               WHERE nprod = 42)
```

b)
```
SELECT Client.nclt, nom, prenom, age, codepostal
FROM CLIENT JOIN Cmd ON Client.nclt = Cmd.nclt
WHERE nprod = 42
```

c)
```
SELECT Client.nclt, nom, prenom, age, codepostal
FROM CLIENT FULL OUTER JOIN Cmd ON Client.nclt = Cmd.nclt
WHERE nprod = 42
```

d)
```
SELECT DISTINCT nclt, nom, prenom, age, codepostal
FROM CLIENT NATURAL JOIN Cmd
WHERE nprod = 42
```

29. Quelles sont les requêtes équivalentes :

a)
```
SELECT nclt, nom
FROM Client
WHERE NOT EXISTS (SELECT *
                  FROM Prod
                  WHERE NOT EXISTS (SELECT DISTINCT(nprod)
                                    FROM Cmd
                                    WHERE nprod = Prod.nprod AND nclt = Client.nclt))
```

b)
```
SELECT nclt, nom
FROM Client
WHERE (SELECT COUNT(DISTINCT nprod)
       FROM Cmd
       WHERE nclt = Client.nclt) = (SELECT COUNT(*)
                                    FROM Produit)
```

c)
```
SELECT C.nclt, C.nom
FROM Client JOIN Cmd ON Client.nclt = Cmd.nclt
            CROSS JOIN Produit
GROUP BY C.nclt, C.nom
HAVING COUNT(DISTINCT Cmd.nprod) = COUNT(Produit.nprod)
```

d)
```
SELECT C.nclt, C.nom
FROM Client JOIN Cmd ON Client.nclt = Cmd.nclt
            CROSS JOIN Produit
GROUP BY C.nclt, C.nom
HAVING COUNT(Cmd.nprod) = COUNT(Produit.nprod)
```

30. Quelles sont les requêtes équivalentes :

a)
```
SELECT nclt, nom, (SELECT SUM(qte)
                   FROM Cmd
                   WHERE nclt = Client.nclt)
FROM Client
```

b)
```
SELECT C.nclt, nom, SUM(qte)
FROM Client C JOIN Cmd on C.nclt = Cmd.nclt
```

c)
```
SELECT C.nclt, nom, SUM(qte)
FROM Client C LEFT OUTER JOIN Cmd on C.nclt = Cmd.nclt
```

d)
```
SELECT C.nclt, nom, SUM(qte)
FROM Client C FULL OUTER JOIN Cmd on C.nclt = Cmd.nclt
```

31. Pour éliminer les doublons de la table `Client`, on utilisera

a)
```
DELETE FROM Client
WHERE EXISTS (SELECT *
              FROM Client c2
              WHERE c2.nom = c1.nom AND
                    c2.nclt <> c1.nclt)
```

b)
```
DELETE FROM Client
WHERE EXISTS (SELECT *
              FROM Client c2
              WHERE c2.nom = c1.nom AND
                    c2.nclt > c1.nclt)
```

c)
```
DELETE FROM Client
WHERE EXISTS (SELECT *
              FROM Client c2
              WHERE c2.nom = c1.nom AND
                    c2.nclt < c1.nclt)
```

d) Toutes les réponses sont fausses.

32. Soit la vue suivante :

```
CREATE VIEW VClient
AS (SELECT *
    FROM Client
    WHERE age > 25)
WITH CHECK OPTION
```

a) La vue VClient est modifiable

b) La vue VClient n'est pas modifiable

c) L'insertion d'un tuple avec l'attribut age < 25 dans la vue `VClient` est possible.

d) L'insertion d'un tuple avec l'attribut age < 25 dans la vue `VClient` n'est pas possible.

33. Soit la création des deux triggers suivants :

```
CREATE or REPLACE TRIGGER TCLIENT1
[BEFORE | AFTER | INSTEAD OF] INSERT ON Client
FOR EACH ROW
BEGIN
INSERT INTO JOURNAL(nclt, operation)
VALUES (:new.nclt, 'INSERT');
END;
```

```
CREATE or REPLACE TRIGGER TCLIENT2
[BEFORE | AFTER | INSTEAD OF] INSERT ON Client
FOR EACH ROW
BEGIN
  :new.nom := UPPER(:new.nom);
END;
```

Le trigger TCLIENT1 doit être déclaré comme un trigger

a) BEFORE

b) AFTER

c) INSTEAD OF

34. Le trigger TCLIENT2 doit être déclaré comme un trigger

a) BEFORE

b) AFTER

c) INSTEAD OF

35. Supposant que l'opération suivante est valide :

```
INSERT INTO VClient
VALUES (42, 'toto', 25, 75013)
```

`VClient` est la vue créée dans les questions précédentes.

a) Le trigger TCLIENT1 sera exécuté

b) Le trigger TCLIENT1 ne sera pas exécuté

c) Le trigger TCLIENT2 sera exécuté

d) Le trigger TCLIENT2 ne sera pas exécuté

36. Supposant qu'on exécute l'opération suivante :

```
INSERT INTO Cmd
VALUES (42, 25, '25/01/2012', 3)
```

En supposant que le produit numéro 25 n'existe pas dans la table Produit :

a) L'opération d'insertion sera effectuée

b) L'opération d'insertion ne sera pas effectuée

c) S'il existe un trigger BEFORE INSERT sur la table `Cmd`, il sera executé

d) S'il existe un trigger AFTER INSERT sur la table `Cmd`, il sera executé

37. Soit la transaction suivante :

```
BEGIN TRANSACTION ISOLATION LEVEL READ COMMITED;
SELECT age
FROM Client
WHERE nclt = 1;

SELECT age
FROM Client
WHERE nclt = 1;

UPDATE Client
SET age = age + 1
WHERE nclt = 1;

COMMIT;
```

a) Les deux SELECT retourneront toujours la même valeur.

b) Les deux SELECT peuvent retourner deux valeurs différentes.

c) Cette transaction peut échouer

d) Cette transaction se terminera toujours normalement

38. Soit la transaction suivante :

```
BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT age
FROM Client
WHERE nclt = 1;

SELECT age
FROM Client
WHERE nclt = 1;

UPDATE Client
SET age = age + 1
WHERE nclt = 1;

COMMIT;
```

a) Les deux SELECT retourneront toujours la même valeur.

b) Les deux SELECT peuvent retourner deux valeurs différentes.

c) Cette transaction peut échouer

d) Cette transaction se terminera toujours normalement

39. Soit la requête suivante :

```
SELECT *
FROM Client
WHERE age = 20
```

Quel est le ou les meilleurs indexes pour améliorer le temps d'exécution de cette requête :

a) Un indexe plaçant sur l'attribut `nclt`

b) Un indexe non plaçant sur l'attribut `nclt`

c) Un indexe plaçant sur l'attribut `age`

d) Un indexe non plaçant sur l'attribut `age`

40. Soit la requête suivante :

```
SELECT *
FROM Client
WHERE age = 20
ORDER BY codepostal
```

Quel est le ou les meilleurs indexes pour améliorer le temps d'exécution de cette requête :

a) Un indexe plaçant sur l'attribut `age`

b) Un indexe non plaçant sur l'attribut `age`

c) Un indexe plaçant sur l'attribut `codepostal`

d) Un indexe non plaçant sur l'attribut `codepostal`