USTL

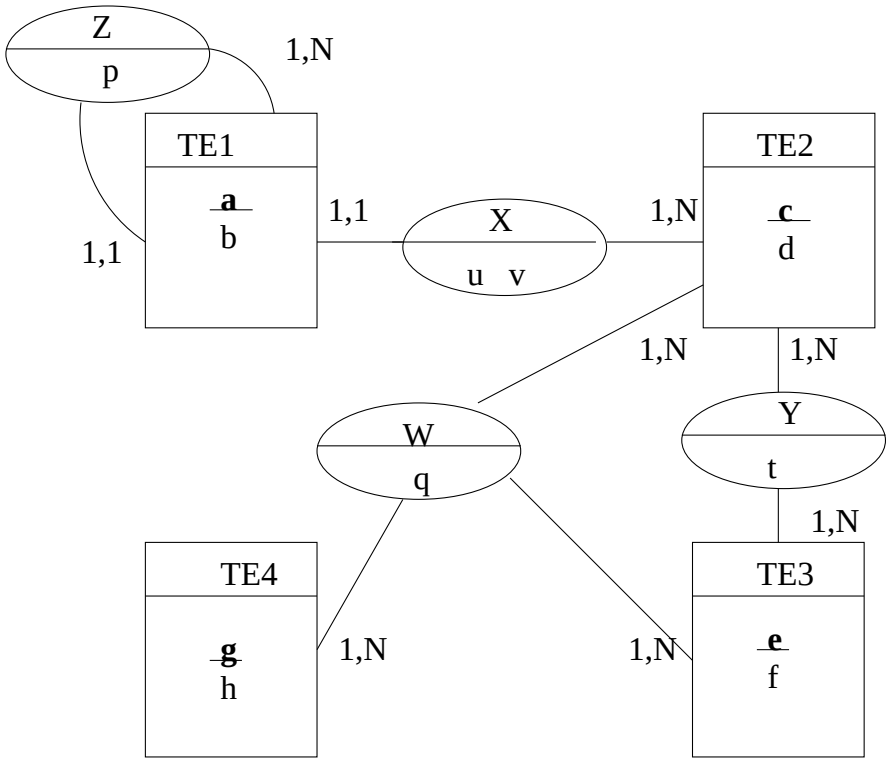
UFR IEEA-INFO

Examen de Bases de Données - janvier 2007

Durée 2h, aucun document autorisé

Question 1 :

En appliquant les règles classiques de conception, donnez le **schéma relationnel** (liste des tables comportant le nom des attributs) correspondant au **modèle conceptuel des données** (MCD) de la figure. Pour chaque table, vous soulignerez d'un trait continu les clé primaires et vous distinguerez les attributs jouant le rôle de clé de référence en les soulignant d'un trait en pointillé :



Question 2 :

On donne le **schéma relationnel** suivant :

`Stations(numsta, nomsta, altitude, numreg)`
`Regions(numreg, nomreg)`
`Activites(numact, libelle)`
`propose(numsta, numact)`

Celui-ci permet de construire la base destinée à gérer les activités des offices de tourismes de différentes stations de sport d'hiver en France. Ces stations, décrites par la table de même nom, proposent différentes activités : ski, luge, mais aussi tennis, piscine, ... La table `propose` est une table d'association traduisant le lien maillé entre une station et les activités qu'elle propose aux touristes. Les attributs de ces différentes tables sont soit des chaines dynamiques (type VARCHAR), soit des entiers (type INTEGER). Toutes leurs clés sont numériques.

Ecrivez, en PL/SQL, la procédure `recherche_activites(reg varchar, alt integer)` qui affiche les activités proposées par les stations de la région dont le nom est donné par le paramètre `reg` et dont l'altitude est supérieure à celle du paramètre `alt`.

Vous utiliserez pour l'affichage les procédures du paquetage DBMS_OUTPUT qui sont : `DBMS_OUTPUT.put`, `DBMS_OUTPUT.put_line` et `DBMS_OUTPUT.new_line`.

La présentation des résultats devra prendre la forme suivante :

```
...
Les Menuires
   ski de piste
   ski de fond
   promenades en raquettes
   promenades en traineau
   luge
   tennis

Meribel
   ski de piste
   ski de fond
   excursion avec attelage de chiens
   tennis
   piscine
...
```

Question 3 :

Une société fabrique et commercialise un certain nombre d'articles. Elle dispose de deux sites séparés constitués d'une usine et d'un magasin. Ces deux sites ont leur propre système informatique et sont munis chacun d'un SGBD.

Chaque site gère une base de données dont le même schéma relationnel est le suivant :

```
articles(numart, designation, prix)
clients(numcli, nomcli, adresse)
stocks(numart, quantite)
```

Chaque SGBD peut accéder à l'autre et en manipuler les tables et leurs données. Pour la base `usine`, les tables gérées par le magasin sont définies par la syntaxe : `nomTable@magasin`, tandis que la base `magasin` voit les tables correspondantes sous le nom `nomTable@usine`.

Chaque base constitue la copie de l'autre, et les mises à jours sont synchrones, au temps de réaction du réseau près. Cela signifie que toute action sur une table de l'usine, par exemple, sera aussitôt répercutée sur la même table du magasin. En particulier, les données de la table `stocks` concernent les stocks de l'entreprise dans sa globalité, et pas seulement ceux d'une des deux entités.

Le moyen utilisé pour obtenir que les tables soient en permanence des miroirs les unes des autres est le *trigger*. Lorsqu'un client vient acheter un article donné au magasin, la gestion concerne en parallèle la base `usine`. La quantité en stock doit y être réduite de la même quantité, ou bien l'article doit disparaître du stock des deux côtés s'il n'y en avait pas à l'usine et que tout a été acheté au magasin. Lorsque l'usine produit un article, une insertion doit être faite des 2 côtés si les stocks étaient vides, ou bien la quantité doit être augmentée de la même valeur du côté magasin si l'article est présent en stock quelque part.

Ecrivez le trigger `gestion_article`, actif au niveau de la base `usine`, qui répercute toutes les opérations effectuées sur sa table `stocks` dans la base `magasin`