1 présentation du modèle relationnel

Le modèle est basé sur une application que vous connaissez bien, à savoir la base ski.

- CLASSEMENT (Refcomp# , Nomski #, Rang)
- SKIEUR (Nomski , Specialite, Nomstat #)
- STATION (nomstat, altstat, paystat, capstat)
- COMPETITION (Refcomp , Datcomp, Nomstat #)

Les clés primaires sont soulignées, les clés étrangéres sont suivies d'un #

2 Création des tables

- Les dates sont de type date
- les attributs rang, capstat, sont de type entier
- les autres attributs sont de type chaîne de caractère, longueur max : 30

2.1 Expliquez l'ordre de création des tables

Il faut respecter les contraintes d'intégrité référentielle (relation clé étrangère - clé primaire)

- STATION (pas de clé étrangère)
- SKIEUR (clé étrangère sur STATION)
- COMPETITION (clé étrangère sur STATION)
- CLASSEMENT (clé étrangère sur COMPETITION et SKIEUR)

2.2 rédigez la requête permettant de créer en 1 instruction la table SKIEUR (en nommant les contraintes).

```
CREATE TABLE SKIEUR
    (
        nomski varchar(30) NOT NULL,
        specialite varchar(30),
        nomstat varchar(30),
        CONSTRAINT SKIEUR_pkey PRIMARY KEY (nomski),
        CONSTRAINT SKIEUR_nomstat_fkey FOREIGN KEY (nomstat)
          REFERENCES STATION (nomstat)
    );
```

La table CLASSEMENT a été créée à l'aide de cette requête :

```
CREATE TABLE CLASSEMENT
    (
        refcomp varchar (30),
        nomski varchar (30)
        rang varchar(30)
    );
```

2.3 sans détruire la table, rédigez les requêtes permettant de rendre la table conforme au modèle relationnel.

```
ALTER TABLE CLASSEMENT ADD CONSTRAINT pk_CLASSEMENT PRIMARY KEY (refcomp,nomski);
ALTER TABLE CLASSEMENT ADD CONSTRAINT FK_CLASSEMENT_ski FOREIGN KEY (nomski) REFERENCES SKIEUR (nomski) ;
ALTER TABLE CLASSEMENT ADD CONSTRAINT FK_CLASSEMENT_comp FOREIGN KEY (refcomp) REFERENCES COMPETITION (refcomp) ;
```

On peut également ajouter les contraintes en une seule opération :

```
ALTER TABLE CLASSEMENT ADD
        (
        CONSTRAINT pk_CLASSEMENT PRIMARY KEY (refcomp,nomski) ,
        CONSTRAINT FK_CLASSEMENT_ski  FOREIGN KEY (nomski) REFERENCES
        SKIEUR (nomski) ,
        CONSTRAINT    FK_CLASSEMENT_comp        FOREIGN    KEY    (refcomp)
        REFERENCES COMPETITION (refcomp)
        );
```

Il ne faut pas non plus modifier le type de rang :

ALTER TABLE CLASSEMENT ALTER column rang TYPE integer ;

(on peut utiliser MODIFY au lieu de ALTER)

Par contre cette syntaxe ne fonctionne pas puisque les 2 types sont différents, et on ne peut passer de l'un à l'autre. Il faut donc supprimer la colonne et la recréer :

```
ALTER TABLE CLASSEMENT DROP  COLUMN rang
ALTER TABLE CLASSEMENT ADD COLUMN rang TYPE INTEGER
```

3 requêtes de sélection

Rédigez les requêtes SQL suivantes :

3.1 Donnez la liste des SKIEURs classés dans les trois premiers d'au moins une compétition

```
SELECT  distinct nomski
FROM CLASSEMENT
WHERE rang in (1,2,3));
```

3.2 Donnez la liste des STATIONs françaises qui n'ont pas organisé de compétition

```
SELECT nomstat
FROM STATION
WHERE paystat = 'France' AND nomstat NOT IN (SELECT nomstat
FROM COMPETITION);
```

3.3 Donnez l'altitude et la capacité des STATIONs où se sont déroulées comp3 et comp4

```
SELECT nomstat,capstat,altstat
FROM STATION
WHERE nomstat in (SELECT nomstat
FROM COMPETITION
WHERE refcomp in ('comp3','comp4'));
```

Autre formulation :

```
SELECT DISTINCT nomstat, capstat, altstat
FROM STATION S,  COMPETITION C
WHERE S.nomstat = C.nomstat
AND refcomp in ('comp3','comp4'));
```

3.4 Donnez le CLASSEMENT des compétitions comp1 et comp2

```
(SELECT refcomp,rang,nomski
FROM CLASSEMENT
WHERE refcomp = 'comp1'
order by rang)
union
(SELECT refcomp,rang,nomski
FROM CLASSEMENT
WHERE refcomp = 'comp2'
order by rang);
```

Autre solution :

```
SELECT refcomp,rang,nomski
FROM CLASSEMENT
WHERE refcomp IN ('comp1','comp2')
order by refcomp, rang)
```

3.5 Donnez la liste des SKIEURs ayant toujours été classés dans les 5 premiers (Nomski en majuscule).

```
SELECT Upper(nomski)
FROM SKIEUR
WHERE nomski NOT IN (SELECT nomski
FROM CLASSEMENT
WHERE rang > 5)
AND nomski IN (SELECT nommski FROM CLASSEMENT);
```

3.6 Donnez le CLASSEMENT moyen, mini et maxi du SKIEUR coco

```
SELECT nomski, round (avg(rang),2) AS moyen, Min(rang) as Mini, Max(rang) As Maxi
FROM CLASSEMENT
WHERE nomski='coco'
GROUP BY nomski;
```

Autre solution sans redonner le nom du skieur :

```
SELECT round (avg(rang),2) AS moyen, Min(rang) as Mini, Max(rang) As Maxi
FROM CLASSEMENT
WHERE nomski='coco'
```

3.7 Donnez le CLASSEMENT de toutes les compétitions se déroulant à Tignes

```
SELECT cl.refcomp,rang,nomski
FROM CLASSEMENT CL, COMPETITION CO
WHERE co.refcomp=cl.refcomp
AND nomstat='tignes'
ORDER BY refcomp,rang;
```

3.8 Donnez le nombre de victoires pour chaque SKIEUR

```
SELECT nomski,count(rang)
FROM CLASSEMENT
WHERE rang=1
GROUP BY nomski;
```

3.9 Donnez le meilleur et le plus mauvais CLASSEMENT pour chaque SKIEUR de Tignes

```
SELECT c.nomski,min(rang),max(rang)
FROM CLASSEMENT c,SKIEUR s
WHERE c.nomski=s.nomski
and nomstat = tignes'
GROUP BY c.nomski;
```

3.10 Donnez la liste des STATIONs d'où sont originaires au moins 5 SKIEURs

```
SELECT nomstat
FROM SKIEUR
GROUP BY nomstat
HAVING count(*) >= 5;
```

3.11 Donnez le nombre de STATION situées en France dont la capacité dépasse 15000 personnes

```
SELECT count(*) FROM STATION
WHERE  capstat > 15000
AND paystat = 'Frnace';
```

3.12 Donnez la liste des compétitions se déroulant en 2004

```
SELECT * FROM COMPETITION
WHERE YEAR (datcomp) = 2004;
```

Sous Postgresql :

```
SELECT * FROM COMPETITION
WHERE EXTRACT (year FROM datcomp) = 2004;
```

3.13 Donnez la liste des skieurs qui prétiquent la même spécialité que 'Tomba'

```
SELECT nomski FROM SKIEUR
WHERE specialite =
(SELECT specialite FROM SKIEUR WHERE nomsli = 'Tomba')
AND nomski <> 'Tomba';
```

Autre formulation :

```
SELECT R.nomski FROM SKIEUR R, skieur M
WHERE R.specialite = M.specialite
AND M.nomski =  'tomba'
AND R.nomski <> 'tomba';
```

4 requêtes de mise à jour

4.1 Insérez un jeu d'essai dans la base de donnée

```
INSERT INTO STATION values ('tignes',2000,'france',27000);
INSERT INTO SKIEUR values ('paulo','bosses','tignes');
INSERT INTO COMPETITION values ('comp1','28-12-2003','tignes');
INSERT INTO CLASSEMENT values ('comp1','paulo',1);
```

4.2 Insérez dans la table SKIEUR le SKIEUR Alphand qui a la même spécialité que le SKIEUR gropiron et qui est de la même STATION que le SKIEUR tomba.

```
INSERT INTO SKIEUR
SELECT 'Alphand', sk1.specialite, sk2.nomstat FROM SKIEUR sk1, SKIEUR sk2
WHERE sk1.nomski = 'Gropiron'
AND sk2.nomski = 'Tomba';
```

4.3 La capacité des grandes STATIONs (plus de 3000 personnes) est augmentées de 10%. Donnez la requête.

```
UPDATE STATION
SET capstat=capstat*1.1
WHERE CAPSTAT >3000;
```

4.4 Supprimer la STATION 'aussois, toutes les compétitions s'y déroulant étant transférées à serfs-chevaliers

```
UPDATE SKIEUR set nomstat='serfs-chevalier' WHERE nomstat='aussois';
UPDATE COMPETITION set nomstat='serfs-chevalier' WHERE nomstat='aussois';
DELETE FROM STATION WHERE nomstat='aussois';
```

4.5 Mettez à jour les données du SKIEUR Lafleche, sachant qu'il fait maintenant de la descente dans la STATION de Tignes

```
UPDATE SKIEUR
SET specialite = 'descente', nomstat = 'Tignes'
WHERE nomski = 'Lafleche';
```

5 travail sur les vues

5.1 rédigez la vue qui permette d'intervenir sur le CLASSEMENT et qui permette de visualiser en plus des STATIONs les dates ainsi que le lieu des compétitions

```
CREATE VIEW CLASSE AS
( SELECT CL.refcomp, nomski, rang, datcomp, nomstat
FROM CLASSEMENT CL, COMPETITION CO
WHERE cl.refcomp=co.refcomp) ;
```

5.2 rédigez la vue qui filtre les compétitions ayant lieu après le 1er mars

```
CREATE VIEW COMPET AS
(SELECT * FROM COMPETITION
WHERE datcomp > '03-01-2004');
```