esprit Se former autrement	# EXAMEN Semestre : 1 ☐ 2 ■ Session : Principale ■ Rattrapage ☐

Module : **SGBD**
Enseignants : **Équipe SGBD**
Classes : **3A**
Documents autorisés : OUI ☐ NON ■ Nombre de pages : **02**
Date : **17-05-2018** Heure : **10h30** Durée : **01h30**

Nous souhaitons créer une application qui permet la gestion d'audience de diverses chaines télévisées. Une partie du schéma relationnel de l'application est donnée ci-dessous :

CHAINETV (**idCH**, nomCH, typeCH)
ANIMATEUR (**idA**, nomA, nationalité, gender)
PROGRAMME **(idP, #idCH,** #idA, titre, typeP, périodicité, jour, heure_debut, durée)
SONDAGE (**heureDeb, heureFin, dateS, #(idP, idCH)**, audimat)

Remarques :

- La **périodicité** peut être (*quotidienne, hebdomadaire*).
- Le type de chaine, **typeCh,** peut être (*ipTV, TNT, satellite*).
- Le type de programme, **typeP**, peut être (*politique, sport, news, documentaire, divertissement*).
- L'**Audimat** correspond au *nombre* de téléspectateurs qui regardent le programme à un moment donné.
- La **durée** d'un programme est exprimée en *minutes*.
- On suppose qu'un programme n'est diffusé qu'une seule fois par jour.

Travail demandé :

1. Créer une vue **V_PROG_MENS** qui permet d'afficher le titre du programme, le nom de l'animateur ainsi que le nom de la chaine pour les programmes dont la **périodicité** est *hebdomadaire*. **(2 pts)**

```
CREATE OR REPLACE VIEW V_PROG_MENS AS SELECT titre, nomA, nomCH FROM CHAINETV ch JOIN PROGRAMME p ON ch.idCH=ch.idCH JOIN ANIMATEUR a ON a.idA=p.idA WHERE trim(lower(périodicité))=' hebdomadaire';
```

2. Écrire une procédure stockée **PS_AUD_PROG** qui permet d'afficher le total des audimats pour chaque type de programme. **(2 pts)**

```
CREATE OR REPLACE PROCEDURE PS_AUD_PROG IS
Cursor curprog is select sum(audimat) somme,typeP FROM PROGRAMME p JOIN SONDAGE s ON p.idP=s.idP and p.idch=s.idch GROUP BY typeP;
BEGIN
FOR prog IN curprog
LOOP
Dbms_output.put_line('Type programme '|| prog.typeP||' - '||prog.typeP);
END LOOP;
END;
/
```

3. Écrire une fonction stockée **FN_NB_PROG(p_idA number)** qui permet de retourner le nombre de programmes présentés par l'animateur dont l'identifiant est donné en paramètre. Si l'animateur n'existe pas la fonction doit retourner -1. **(2 pts)**

```
CREATE OR REPLACE FUNCTION FN_NB_PROG (p_idA number)
RETURN INT
IS
Nb int;
BEGIN
SELECT COUNT(*) into nb from ANIMATEUR WHERE ida=p_ida;
IF nb=0 then
Return -1;
else
SELECT COUNT(*) into nb from PROGRAMME WHERE ida=p_ida;
Return nb;
End if;
END;
/
```

4. Écrire une procédure stockée **PS_NOM_PROG** qui permet d'afficher pour chaque animateur, ayant plus de deux programmes, son nom et sa nationalité ainsi que les titres, types et périodicité des programmes qu'il présente. Utiliser la fonction **FN_NB_PROG**. **(3 pts)**

```
CREATE OR REPLACE PROCEDURE PS_NOM_PRG IS
CURSOR curanim IS SELECT nomA, nationalité FROM ANIMATEUR WHERE FN_NB_PROG(idA)>2;
CURSOR curprog(p_idA ANIMATEUR.idA%TYPE) IS SELECT titre, typeP, périodicité FROM PROGRAMME WHERE idA=p_idA;
BEGIN
FOR anim IN curanim
LOOP
Dbms_output.put_line('Animateur '||anim.nomA||' '||anim.nationalité);
        FOR prog IN curprog(anim.idA)
        LOOP
```

```
        Dbms_output.put_line('Programme '|| prog.titre||' '||prog.typeP||' '||prog. périodicité);
        END LOOP;
END LOOP;
END;
/
```

5. Écrire une fonction stockée **FN_POUR_AUDIMAT (p_idCH number, p_date date)** qui retourne, pour une date donnée, le pourcentage de l'audimat (voir encadré ci-dessous) d'une chaine TV dont l'identifiant est donné en paramètre. Si la chaine TV n'existe pas la fonction doit retourner -1. **(3 pts)**

> Le pourcentage de l'audimat d'une chaine = audimat total chaine / audimat total

```
CREATE OR REPLACE FUNCTION FN_POUR_AUDIMAT (p_idCH number, p_date date)
RETURN number
IS
Nb number;
Tot number;
BEGIN
SELECT COUNT(*) into nb from CHAINETV WHERE idch=p_idch;
IF nb=0 then
Return -1;
Else
SELECT SUM(audimat) into tot from SONDAGE;
If tot!=0 then
SELECT SUM(audimat)/tot into nb from SONDAGE WHERE idch=p_idch and dates=p_date;
Return nb;
Else
Return 0;
End if;
End if;
END;
/
```

6. Écrire une procédure stockée **PS_CLASS_CHAINE (p_date date)** qui affiche, pour une date donnée, une liste numérotée des noms des chaines selon un ordre décroissant du pourcentage de leur audimat. Utiliser la fonction **FN_POUR_AUDIMAT**. **(2 pts)**

```
CREATE OR REPLACE PROCEDURE PS_CLASS_CHAINE (p_date date) IS
CURSOR curchaine IS SELECT nomch, row_number() over(order by FN_POUR_AUDIMAT(idch) DESC) nb FROM CHAINETV WHERE p_date=dateS ORDER BY  FN_POUR_AUDIMAT(idch) DESC;
BEGIN
FOR chaine IN curchaine
LOOP
Dbms_output.put_line(chaine.nomch||' - '||chaine.nb);
END LOOP;
END;
/
```

7. Écrire une procédure stockée **PS_TOP_PROG (p_heureDeb number, p_heureFin number, p_date date)** qui permet d'afficher le nom du(des) programme(s) qui a(ont) réalisé l'audimat le plus élevé pour la plage horaire et la date indiquées en paramètre.

   Traiter les exceptions suivantes :

   - L'heure de fin doit être supérieure à l'heure de début
   - La date ne doit pas être supérieure à la date d'hier. **(3 pts)**

```
CREATE OR REPLACE PROCEDURE PS_TOP_PROG (p_heureDeb number, p_heureFin number, p_date date) IS
CURSOR curprog IS SELECT nomp FROM PROGRAMME p JOIN SONDAGE s ON s.idch=p.idch and s.idp=p.idp where p_heureDeb= heureDeb and  p_heureFin= heureFin and p_date=dates group by p.idp having sum(audimat) = (select max(sum(audimat)) from sondage where p_heureDeb= heureDeb and  p_heureFin= heureFin and p_date=dates group by p.idp);
BEGIN
IF p_heureDeb > p_heureFin THEN
RAISE_APPLICATION_ERROR(-20001,'L''heure de début doit être < à l''heure de fin');
Elsif p_date>=sysdate THEN RAISE_APPLICATION_ERROR(-20002,'Le jour de diffusion doit être inférieur à '||sysdate-1);
END IF;
FOR prog IN curprog
LOOP
Dbms_output.put_line(prog.nomp);
END LOOP;
END;
/
```

8. Écrire un trigger **TRIG_PERIODICITE** qui permet de vérifier la valeur de la colonne ***jour*** lors de l'insertion au niveau de la table **PROGRAMME** :

   a. si la périodicité est **'quotidienne'** la colonne ***jour*** sera nulle.

   b. si la périodicité est **'hebdomadaire'**, la colonne ***jour*** doit avoir une valeur, sinon, l'insertion doit être bloquée. **(3 pts)**

```
CREATE OR REPLACE TRIGGER TRIG_PERIODICITE
BEFORE
INSERT
ON PROGRAMME
FOR EACH ROW
BEGIN
IF trim(lower(:new.periodicité )) = 'quotidienne' then
:new.jour:=NULL;
ELSE
IF :new.jour IS NULL then
RAISE_APPLICATION_ERROR(-20001,'Le jour de diffusion doit être précisé');
End if;
End if;
END;
/
```

Bon courage