VERSION FINALE :

PROJET de Base de Données

UE LSIN513 – Année 2024-2025

Groupe : TD1

KHAZEM Lynda

MACHTER Massinissa

HATEM Nadir

Partie 1 : Conception et Modélisation

1. Description des besoins de notre application :

1. Contexte du projet

L'objectif de ce projet est de développer une base de données pour une chaîne de télévision afin de gérer efficacement les informations liées aux concepts : émissions, programmes, salariés, invités, publicités, audiences, et leurs relations spécifiques. Cette base de données permettra de centraliser et d'organiser toutes les informations liées aux activités de la chaîne.

2. Description des concepts et fonctionnalités :

Une **émission** peut être diffusée une ou plusieurs fois, et chaque diffusion est appelée un **programme**. Si une émission est récurrente, comme un talk-show ou une série, elle aura plusieurs programmes à des dates différentes. À chaque diffusion d'un programme, **l'audience** est enregistrée pour savoir combien de personnes ont regardé l'émission. Une émission peut également avoir un type (spéciale, hebdomadaire…), une catégorie (film, documentaire,…) et un genre spécifique, comme divertissement, scientifique, action, horreur...

Les **salariés** de la chaîne jouent des rôles clés dans la production des émissions. Un salarié peut être soit un **animateur** ou **technicien** avec une spécialité précise.

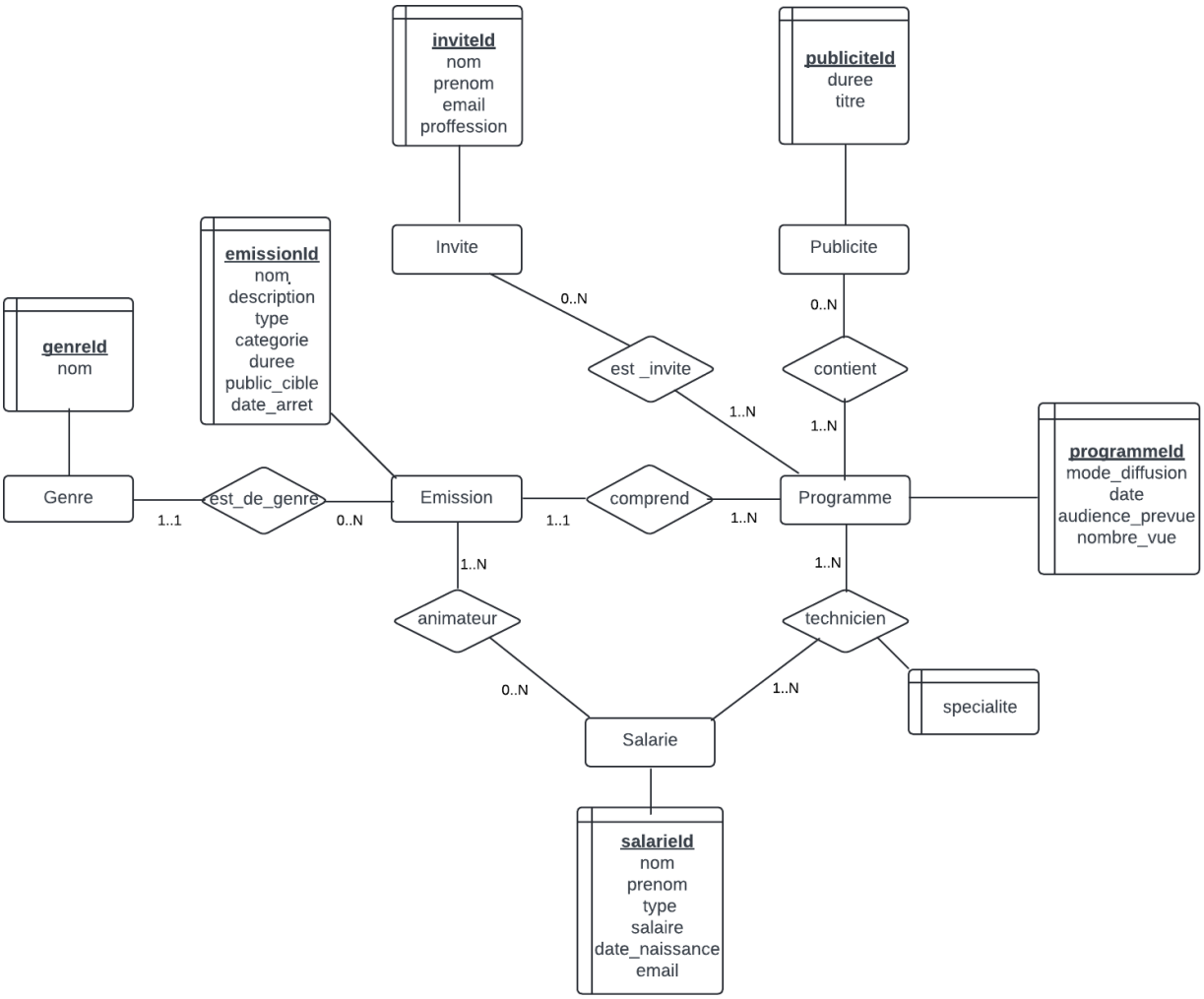
Dans les émissions où il y a des animateurs, ceux-ci jouent un rôle central, notamment dans les émissions de divertissement, de débats ou d'interviews, où ils présentent et animent le contenu. En revanche, dans des émissions comme les films ou les documentaires, il n'y a pas d'animateur. En plus de cela, une émission peut aussi recevoir des **invités**, qui viennent participer à certaines émissions pour discuter ou performer (exemple : Talk-show).

Chaque programme est également soutenu par une équipe de techniciens, qui s'occupent de la gestion technique. Ils assurent le bon déroulement du programme en prenant en charge le son, l'image, la lumière, et les caméras. Que le programme soit en direct ou préenregistré ou en replay, leur travail est essentiel pour garantir une qualité de diffusion optimale.

De plus, un programme a une durée connue et, pendant cette durée, il peut inclure des **publicités**. Chaque publicité a une durée précise.

2. Modélisation :

Modèle E/A détaillé :



3. Contraintes d'intégrité :

1. Simples :

Concepts	Identifiant unique et obligatoire	Valeurs obligatoires	Validation de valeurs
Emission	emissionId	nom,type,public_cible,duree	Public_cible doit être un entier qui représente l'âge minimum. duree est positive (0<duree<3h)
Programme	programmeId	Date,nombre_vue,audience_prevue	nombre_vue et audience_prevue doivent être des entiers positifs (>=0)
Salarie	salarieId	nom,prenom,date_nissance,salaire	Salaire >smic actuel
Invite	InviteId	nom,prenom	Rien
Genre	genreId	nom	Rien
Publicite	publiciteId	duree	duree ne doit pas dépasser 5 min

Plus de cela, s'ajoute les liens entre les différents éléments permettent de connecter des informations entre concepts. Cela garantit que chaque donnée est bien reliée à une autre et reste cohérente, par exemple un programme ne peut exister sans l'émission à laquelle il est lié.

2. Complexes :

- On ne peut pas avoir deux programmes diffusés à la même date et heure
- Dans un programme la durée totale des publicités ne peut pas dépasser 15% de la durée totale du programme
- Les émissions destinées à un public majeur doivent être diffusées uniquement entre 23h et 6h.
- Une émission en état d'arrêt ne peut plus être programmée.
- Un invité peut être invite au plus une seule et une seule fois par jour.
- Un animateur ne peut animer au plus 3 émissions différentes.
- Un technicien ne peut travailler à plus de 3 programmes par jour.
- Un salarie peut être soit un animateur ou soit un technicien et pas les deux en même temps.

4. Droits et Vues – confidentialité des données:

1. Vues :

- Vue des émissions par type.

- Vue des animateurs avec leurs émissions et leur audience moyenne.

- Durée de total de séquences publicitaires par émissions.

- Durée moyenne de travail journalière des techniciens selon la spécialité.

- Liste des salaries ayant travaillé la journée du 'dd/mm/aa'.

- Vue sur toutes les émissions du '01-11-2024' et leur public cible.

2. Droits :

- Administrateur BD : accès complet à toute la BD et ses fonctionnalités.

- Programmateur: accès total aux émissions et programmes.

- RH : gestions des salaries.

- Animateur: accès en lecture à ses informations personnelles et accès totale aux émissions sur lesquelles il travaille.

- audimat : peut consulter les informations sur les audiences des émissions (faire de l'analyse) mais sans modifier les données.

- User Guest (Tout public) : accès en lecture seule aux informations des émissions et leurs programmes sans pouvoir accéder aux informations sur l'audience.

5. Requêtes : Qu'est-ce qui intéresse les utilisateurs dans notre BD?

1- Quelles sont les émissions disponibles sur la chaine avec leur genre ?

2- Liste de toutes les émissions de type spéciale.

3- Quelles émissions ont une durée supérieure à une heure ?

4- Quels sont les techniciens qui ont travaillé sur les programmes de l'émission 'x' ?

5- Quels sont les publicités passées lors de la diffusion en direct des programmes du d/mm/aa?

6- Quelles émissions sont enregistrées dans la chaine mais n'ont pas encore eu de programme diffusé ?

7- Durée totale des programmes diffusés par mode de diffusion.

8- Quels sont les programmes contenants les publicités les plus longues ?

9- Quels animateurs ont animé plus de deux émissions différentes ?

10- Quels sont les animateurs ainsi que les techniciens qui ont travaillés sur l'émission 'x' ?

11- liste des invites ayant participé à des programmes qui ne contiennent pas de publicités ?

12- Quelle est l'émission qui a le plus d'invités enregistrés ?

13- Salaire moyen des animateurs pour chaque émission, uniquement si leur salaire moyen dépasse la moyenne de tout les salariés ?

14- Noms d'émissions qui ont enregistré l'audience la plus faible ?

15- Quels programmes de chaque émission ont enregistré une audience plus que prévue ?

16- Quelles sont les émissions dont les invités ne sont présents qu'aux programmes de ces émissions ? En citant les noms des invites.

17- Trouver les personnes qui ont été invité dans toutes les programme en direct de l'émission 'y' dans la semaine 'x'.

18- Nombre total d'émissions par genre avec nombre total de vues supérieure à l'audience totale prévue.

19- Quelles sont les émissions qui ont été programmées plus de 10 fois ?

20- quelle est la durée moyenne des publicités selon les différentes audiences enregistrées ? Triée par ordre croissant.

21- Pour chaque technicien, fournir la liste des émissions sur lesquelles il a travaillé, ainsi que le nombre de programmes où il a participé pour chacune de ces émissions ?

22- Quelles sont les plages horaires ou l'audience est la plus élevée?

6. Traduction du schéma E/A en relationnel :

Notre base de données a la structure suivante :

Genre (**genreId**,nom)
Emission (**emissionId**, nom, description,type,categorie,duree,public_cible,date_arret, genreId)
Programme (**programmeId**, mode_diffusion, date, audience_prevue, nombre_vue , emissionId)
Invite (**inviteId**, nom, prenom, email, profession)
Est_invite (programmeId, inviteId**)**
Salarie (**salarieId**, nom, prenom,type, salaire, date_naissance, email)
Animateur (salarieId, emissionId)
Technicien (salarieId ,programmeId,specialite)
Publicite (**publiciteid**,titre,duree)
Contient (programmeId, publiciteId)

Rq : le champ **date** dans **programme** comprend à la fois la date et l'heure de début de programme.

Partie 2 : Implémentation de notre BD sous Oracle :

1. Création du schéma de la base de données:

Création des tables avec les différents contraints d'intégrités simples :

```
create table GENRE (
        genreId number(2) PRIMARY KEY,
        nom varchar(60) NOT NULL
);
```

Rq : durée représente la durée en minutes.

```
create table EMISSION (
        emissionId number(4) PRIMARY KEY,
        nom varchar(60) NOT NULL ,
        description varchar(120) ,
        type varchar(60),
        categorie varchar(60) NOT NULL,
        duree number NOT NULL,
        public_cible NUMBER(3) NOT NULL,
        date_arret date ,
        genreId number(2),
        CONSTRAINT chk_public_cible CHECK (public_cible BETWEEN 0 AND 18),
        CONSTRAINT chk_duree_emission CHECK (duree BETWEEN 0 AND 180),
        FOREIGN KEY (genreId) references GENRE(genreId) );
```

Rq : la colonne date contient la date et l'heure exacte

```
create table PROGRAMME (
        programmeId number(4) PRIMARY KEY,
        mode_diffusion varchar(20),
        "date" DATE NOT NULL unique,
        audience_prevue integer NOT NULL,
        nombre_vue integer NOT NULL ,
        emissionId number(4),
        CONSTRAINT chk_nombre_vue CHECK (nombre_vue>=0),
        CONSTRAINT chk_audience_prevue CHECK (audience_prevue>=0),
        FOREIGN KEY (emissionId) references EMISSION(emissionId)
);

create table INVITE (
        inviteId NUMBER(4) PRIMARY KEY,
        nom VARCHAR2(60) NOT NULL,
        prenom VARCHAR2(60) NOT NULL,
        email VARCHAR2(100) ,
        profession VARCHAR2(60)
);

create table EST_INVITE (
   programmeId NUMBER(4),
   inviteId NUMBER(4),
   PRIMARY KEY (programmeId, inviteId),
   FOREIGN KEY (programmeId) REFERENCES PROGRAMME(programmeId),
   FOREIGN KEY (inviteId) REFERENCES INVITE(inviteId)
);
```

```
create table SALARIE (
        salarieID number(4) PRIMARY KEY ,
        nom varchar(60) NOT NULL,
        prenom varchar(60)NOT NULL,
        "type" varchar(20) NOT NULL,
        salaire number NOT NULL,
        date_nissance date NOT NULL,
        email varchar(100),
        CONSTRAINT chk_type_salarie CHECK (type in ('Animateur','Technicien')),
        CONSTRAINT chk_salaire CHECK (salaire>=1766.92)-- >= smic actuel
);

create table ANIMATEUR (
        salarieId NUMBER(4),
        emissionId NUMBER(4),
        PRIMARY KEY (salarieId , emissionId),
        FOREIGN KEY (salarieId) REFERENCES SALARIE(salarieId) ,
        FOREIGN KEY (emissionId) REFERENCES EMISSION(emissionId)
);

create table TECHNICIEN (
        salarieId NUMBER(4),
        programmeId NUMBER(4),
        specialite varchar(80),
        PRIMARY KEY (salarieId , programmeId),
        FOREIGN KEY (salarieId) REFERENCES SALARIE(salarieId) ,
        FOREIGN KEY (programmeId) REFERENCES PROGRAMME(programmeId)  );

create table PUBLICITE(
        publiciteId number(4) PRIMARY KEY,
```

```
    titre varchar(120),
    duree number(1) NOT NULL,
    CONSTRAINT chk_duree_pub CHECK (duree>0 and duree<=5)
);

create table CONTIENT(
    programmeId NUMBER(4),
    publiciteId NUMBER(4),
    PRIMARY KEY (programmeId,publiciteId),
    FOREIGN KEY (publiciteId) REFERENCES PUBLICITE(publiciteId) ,
    FOREIGN KEY (programmeId) REFERENCES PROGRAMME(programmeId)
);
```

2. Intégrité des données : les triggers :

1) On ne peut pas avoir deux programmes diffusés à la même date et heure

```
Create or replace trigger trg_programmes_simultanes before insert on programme for each row
DECLARE
    prog_avant number :=0;
    prog_apres number :=0;
    d Emission.duree%TYPE;
BEGIN
    select duree into d from emission
    where emissionId = :new.emissionId;

    select count(*) into prog_avant
    from programme p join emission e on p.emissionId = e.emissionId
    where p."date" + (e.duree/1440) > :new."date"  and p."date" <= :new."date";
    select count(*) into prog_apres
    from programme p
     where :new."date" +(d/1440) > p."date"
     and p."date" >= :new."date";
```

```
    if prog_avant>0 or prog_apres>0 then
        raise_application_error(-20003, 'Insertion impossible : chevauchement de
        programmes détecté.');
    end if;

END;
/
```

2) Dans un programme la durée totale des publicités ne peut pas dépasser 15% de la Durée totale du programme :

```
create or replace trigger TRG DUREE_PUB_CONTIENT_UN_PROG before insert on CONTIENT for
each row
    declare
        pub_total publicite.duree%TYPE :=0;
        duree_prog emission.duree%TYPE;
        duree_new_pub publicite.duree%TYPE;
    begin
        select sum(pub.duree) into pub_total
        from programme p,publicite pub,contient c
        where  p.programmeId=:new.programmeId and p.programmeId=c.programmeId
            and c.publiciteId=pub.publiciteId;

        select e.duree into duree_prog
        from emission e,programme p
        where e.emissionId=p.emissionId and p.programmeId=:new.programmeId;

        select pub.duree into duree_new_pub
        from publicite pub where pub.publiciteId=:new.publiciteId;
        if (pub_total + duree_new_pub) > (0.15 * duree_prog) then
        raise_application_error(-20009, 'Err:Durée des publicités dépasse 15% de la durée du
        programme.')
        end if;
    END;
    /
```

3) Les émissions destinées à un public majeur doivent être diffusées uniquement entre 23h et 6h :

```
create or replace trigger TRG_PUBLIC_MAJEUR_HEURE before insert on programme for each row
    declare
```

```
        age_min emission.public_cible%TYPE;
        heure_diff number;
    begin
        select e.public_cible into age_min
        from emission e
        where e.emissionId=:new.emissionId;
        if (age_min=18) then
            heure_diff:=TO_NUMBER(TO_CHAR(:new."date",'HH24'));
            if not (heure_diff>=23 or heure_diff<=6) then
                raise_application_error(-20010,'Err:Les émissions pour public majeur
                doivent être diffusées entre 23h et 6h.');
            end if;
        end if ;
    end;
    /
```

4) Une émission en état d'arrêt ne peut plus être programmée :

```
CREATE OR REPLACE TRIGGER TRG_EMISSION_ARRETEE BEFORE INSERT OR UPDATE ON
PROGRAMME FOR EACH ROW
DECLARE
     v_emission_arrete EMISSION.date_arret%TYPE;
BEGIN
    SELECT date_arret INTO v_emission_arrete
    FROM EMISSION
    WHERE emissionId = :NEW.emissionId;

    IF v_emission_arrete != NULL THEN
        RAISE_APPLICATION_ERROR(-20001, 'L''émission est arrêtée et ne peut plus être
        programmée.');
    END IF;
END;
/
```

5) Un invité peut être invite au plus une seule et une seule fois par jour :

```
create or replace trigger trg_invite_par_jour before insert on est_invite for each row
DECLARE
    nb_prog number(4);
BEGIN
    select count(*) into nb_prog
    from est_invite e join programme p on e.programmeId = p.programmeId
    join programme p_new on p_new.programmeId = :new.programmeId
    where :new.inviteId = e.inviteId
    and trunc(p_new."date") = trunc(p."date") ;
    if nb_prog >=1 then
        raise_application_error(-20004,'Insertion impossible :un invite est invité au plus 1 fois/j. ');
    end if;
END;
/
```

6) Un animateur ne peut animer au plus 3 émissions différentes :

```
CREATE OR REPLACE TRIGGER TRG_NOMBRE_EMISSIONS_ANIMATEUR BEFORE INSERT OR UPDATE
ON ANIMATEUR FOR EACH ROW
    DECLARE
        v_nb_emissions INTEGER;
    BEGIN
        SELECT COUNT(*) INTO v_nb_emissions
        FROM ANIMATEUR
        WHERE salarieId = :NEW.salarieId;

        IF v_nb_emissions >= 3 THEN
         RAISE_APPLICATION_ERROR(-20002, 'Un animateur ne peut animer plus de 3
        émissions.');
        END IF;
    END;
    /
```

7) Un technicien ne peut travailler à plus de 3 programmes par jour :

```
Create or replace trigger trg_technicien_prog  before insert on technicien for each row
    DECLARE
        nb_prog number(4);
    BEGIN
        select count(p.programmeId) into nb_prog
        from programme p join technicien t on p.programmeId = t.programmeId
        where t.salarieId = :new.salarieId
        and trunc(p."date") = (select trunc(p2."date")
                               from programme p2
                               where p2.programmeId = :new.programmeId);

        if nb_prog>=3 then
            raise_application_error(-20005,'Insertion impossible : Un technicien ne peut
            travailler à plus de 3 programmes/j.');
        end if;
    END
    /
```

8) Un salarie peut être soit un animateur ou soit un technicien et pas les deux en même temps.

Partager sur 2 triggers**:**

```
create or replace trigger TRG_SALARIE_UNIQUE_ROLE_ANIM before insert or update on animateur
for each row
    declare
        deja_technicien number ;
        type_s salarie.type%TYPE;
    begin
        select count(*) into deja_technicien
        from technicien t
        where t.salarieId =:new.salarieId;
```

```
        select s.type into type_s
        from salarie s where s.salarieId=:new.salarieId;

        if ((deja_technicien >0) or (type_s!='Animateur'))  then
                raise_application_error(-20011,'Err:Un salarié ne peut être à la fois
                animateur et technicien.');
        end if ;
end;
/
```

```
create or replace trigger TRG_SALARIE_UNIQUE_ROLE_TECHN before insert or update on technicien
for each row
        declare
                deja_animateur number ;
                type_s salarie.type%TYPE;
        begin
                select count(*) into deja_animateur
                from animateur a
                where a.salarieId =:new.salarieId;

                select s.type into type_s
                from salarie s where s.salarieId=:new.salarieId;

                if ((deja_animateur >0) or (type_s!='Technicien'))  then
                        raise_application_error(-20012,'Err:Un salarié ne peut être à la fois
                        animateur et technicien.');
                end if ;
        end;
        /
```

3. Jeu de données :

Le jeu de données a été soigneusement préparé pour répondre aux exigences du projet. Chaque table contient au moins 30 n-uplets, avec des valeurs cohérentes et conformes au schéma de la base de données. Une attention particulière a été portée à la pertinence et à la qualité des données afin de permettre la validation de requêtes complexes, notamment celles impliquant des jointures et des calculs avancés.

Les données ont été insérées en utilisant deux modes : un mode d'insertion simple via des commandes SQL classiques, et un mode de chargement massif avec Oracle Load pour optimiser le traitement de volumes importants de données. Le jeu de données a été récupéré à partir de sources web (aide : Chatgpt) et organisé pour faciliter son intégration et son exploitation dans la base de données.

Voici un aperçu :

- Par exemple sur la table Emission avec **SQL*Load : fichier '.ctl'**

```
LOAD DATA
INFILE *
APPEND
INTO TABLE EMISSION
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
TRAILING NULLCOLS
(emissionId,nom,description,type,categorie ,duree,public_cible,date_arret DATE 'DD-MM-YYYY'
,genreId)
BEGINDATA
1,"Le 20h","Journal télévisé de France ","Quotidienne","Emission infos",30,6,,6
2,"C'est dans l'air","Débat sur l'actualité politique et sociale","Hebdomadaire","Politique",60,18,,13
3,"Télématin","Programme d'information du matin","Quotidienne","Emission infos",120,6,,6
…
34,"Les Infiltrés","Un flic et un criminel infiltrent les deux camps opposés dans un milieu
criminel.",,"Film",151,18,,29
35,"Cœurs Égarés","Un couple est perturbé par la rupture de leur exclusivité émotionnelle et
sexuelle.",,"Film",92,18,,12
```

- Le Résultat de l'insertion dans **EMISSION** avec **SQL*LOAD** est décrit dans le fichier 'emission.log' :

```
Control File:   projet_oracle/data_emission.ctl
Data File:      projet_oracle/data_emission.ctl
…
Table EMISSION:
  34 Rows successfully loaded.
  0 Rows not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.

Space allocated for bind array:                148608 bytes(64 rows)
Read   buffer bytes: 1048576

Total logical records skipped:          0
Total logical records read:            34
Total logical records rejected:         0
Total logical records discarded:        0
```

- Par exemple sur la table Programme avec des **Insert** :

```
INSERT INTO PROGRAMME VALUES (1, 'premiere', TO_DATE('2024-10-28 07:20', 'YYYY-MM-DD
HH24:MI'), 250000, 262000, 18);
INSERT INTO PROGRAMME VALUES (2, 'direct', TO_DATE('2024-10-28 08:30', 'YYYY-MM-DD
HH24:MI'), 200000, 213000, 3);
INSERT INTO PROGRAMME VALUES (3, 'premiere', TO_DATE('2024-10-28 18:30', 'YYYY-MM-DD
HH24:MI'), 150000, 120000, 19);
….

INSERT INTO PROGRAMME VALUES (58, 'replay', TO_DATE('2024-11-03 2:30', 'YYYY-MM-DD
HH24:MI'), 10000, 10000, 3);
INSERT INTO PROGRAMME VALUES (59, 'replay', TO_DATE('2024-11-04 2:30', 'YYYY-MM-DD
HH24:MI'), 10000, 10000, 3);
```

- Nombre de tuples par table :

  GENRE : 30, EMISSION : 34, PROGRAMME : 59, INVITE : 30, EST_INVITE :38, SALARIE : 30, TECHNICIEN : 116, PUBLICITE : 30, CONTIENT : 41.

4. Manipulation des données :

• Requêtes :

1- Quelles sont les émissions disponibles sur la chaine avec leur genre ?

```
SELECT E.nom AS NomEmission, G.nom AS NomGenre
FROM EMISSION E JOIN GENRE G ON E.genreId = G.genreId;
```

2- Liste de toutes les émissions de type spéciale.

```
select *
from Emission
where type = 'Spéciale';
```

3- Quelles émissions ont une durée supérieure à une heure ?

```
select e.*
from emission e
where e.duree >60 ;
```

4- Quels sont les techniciens qui ont travaillé sur les programmes de l'émission 'x' ?

```
SELECT S.nom, S.prenom
FROM TECHNICIEN T JOIN PROGRAMME P ON T.programmeId = P.programmeId
JOIN EMISSION E ON P.emissionId = E.emissionId JOIN SALARIE S ON T.salarieId = S.salarieId
WHERE E.nom = 'x';
```

5- Quels sont les publicités passées lors de la diffusion en direct des programmes du 2024-11-01?

```
select p.titre
from publicite p,contient c,programme pr
where p.publiciteId = c.publiciteId and c.programmeId = pr.programmeId
and pr.mode_diffusion = 'direct'
and TRUNC(pr."date") = TO_DATE('2024-11-01', 'YYYY/MM/DD');
```

6- Quelles émissions sont enregistrées dans la chaine mais n'ont pas encore eu de programme diffusé ?

```
SELECT E.nom
FROM EMISSION E
WHERE E.emissionId not in (select emissionId from programme);
```

7- Durée totale des programmes diffusés par mode de diffusion.

```
select p.mode_diffusion,sum(e.duree) as duree_totale
```

```
from programme p join emission e
on p.emissionId = e.emissionId
group by p.mode_diffusion;
```

8- Quels sont les programmes contenants les publicités les plus longues ?

```
select p.programmeId,p."date"
from programme p , contient c , publicite pub
where p.programmeId=c.programmeId and c.publiciteId=pub.publiciteId
group by (p.programmeId,p."date")
having max(pub.duree)=(select max(duree) from publicite);
```

9- Quels animateurs ont animé plus de deux émissions différentes ?

```
SELECT S.salarieId, S.nom, S.prenom
FROM ANIMATEUR A
JOIN SALARIE S ON A.salarieId = S.salarieId
GROUP BY S.salarieId, S.nom, S.prenom
HAVING COUNT(DISTINCT A.emissionId) >= 2;
```

10- Quels sont les animateurs ainsi que les techniciens qui ont travaillés sur l'émission 'Le 20h' ?

```
select s.nom,s.prenom,s."type"
from salarie s, animateur a, emission e
where s.salarieId = a.salarieId  and a.emissionId = e.emissionId and e.nom = 'Le 20h'
union
select s.nom,s.prenom,s."type"
from salarie s, technicien t, programme p,emission e
where s.salarieId = t.salarieId  and t.programmeId = p.programmeId
and p.emissionId = e.emissionId and e.nom = 'Le 20h';
```

11- liste des invites ayant participé à des programmes qui ne contiennent pas de publicités ?

```
SELECT I.nom, I.prenom
FROM INVITE I JOIN EST_INVITE EI ON I.inviteId = EI.inviteId
JOIN PROGRAMME P ON EI.programmeId = P.programmeId
WHERE NOT EXISTS (
        SELECT 1
        FROM CONTIENT C
        WHERE C.programmeId = P.programmeId
);
```

12- Quelle est l'émission qui a le plus d'invités enregistrés ?

```
SELECT E.emissionId,E.nom
FROM EMISSION E JOIN PROGRAMME P ON E.emissionId = P.emissionId
JOIN EST_INVITE EI ON P.programmeId = EI.programmeId
GROUP BY E.emissionId,E.nom
HAVING COUNT(DISTINCT EI.inviteId) = (
SELECT MAX(nb_invites)
FROM (SELECT E.nom, COUNT(DISTINCT EI.inviteId) AS nb_invites
        FROM EMISSION E
        JOIN PROGRAMME P ON E.emissionId = P.emissionId
        JOIN EST_INVITE EI ON P.programmeId = EI.programmeId
        GROUP BY E.nom));
```

13- Salaire moyen des animateurs pour chaque émission, uniquement si leur salaire moyen dépasse la moyenne de tout les salariés ?

```
select e.emissionId,e.nom,avg(s.salaire) as Sal_moy_animateurs
from salarie s , animateur a , emission e
where s.salarieId=a.salarieId and a.emissionId= e.emissionId
group by (e.emissionId,e.nom)
having avg(s.salaire)>(select avg(salaire) from salarie);
```

14- Noms d'émissions qui ont enregistré l'audience la plus faible ?

```
select e.nom
from emission e where e.emissionId in
(select p.emissionId
from programme p
where p.nombre_vue=(select min(nombre_vue) from programme)
);
```

15- Quels programmes de chaque émission ont enregistré une audience plus que prévue ?

```
SELECT E.nom AS Emission, P.programmeId
FROM EMISSION E JOIN PROGRAMME P ON E.emissionId = P.emissionId
WHERE P.nombre_vue > P.audience_prevue;
```

16- Quelles sont les émissions dont les invités ne sont présents qu'aux programmes de ces émissions ? En citant les noms des invites.

```
select e.emissionId,e.nom as nom_emission ,i.nom as invite
from invite i, est_invite ei, programme p,emission e
where i.inviteId = ei.inviteId and ei.programmeId = p.programmeId
and p.emissionId = e.emissionId
and i.inviteId not in (select ei2.inviteId
                       from est_invite ei2, programme p2
                       where ei2.programmeId = p2.programmeId
                       and p2.emissionId != e.emissionId
                       )
group by e.emissionId,e.nom,i.nom;
```

17- Trouver les personnes qui ont été invité dans toutes les programme en direct de l'émission 'y' dans la semaine **'2024-11-01' - '2024-11-07'**.

```
select i.inviteId,i.nom,i.prenom
from invite i , est_invite inv , programme p , emission e
where i.inviteId=inv.inviteId and inv.programmeId=p.programmeId
and p.emissionId=e.emissionId and p.mode_diffusion='direct' and e.nom=y
and p."date" BETWEEN TO_DATE('2024-11-01', 'YYYY-MM-DD') AND TO_DATE('2024-
11-07', 'YYYY-MM-DD')
group by (i.inviteId,i.nom,i.prenom)
having count(distinct p.programmeId)=
(select count(p1.programmeId)
from programme p1 , emission e1
```

```
where p1.emissionId=e1.emissionId and p1.mode_diffusion='direct' and e1.nom=y  and
p1."date" BETWEEN TO_DATE('2024-11-01', 'YYYY-MM-DD') AND TO_DATE('2024-11-07',
'YYYY-MM-DD') );
```

18- Nombre total d'émissions par genre avec nombre total de vues supérieure à l'audience totale prévue.

```
select g.genreId,g.nom,count(tmp.emissionId) as nb_emiss_satisf
from genre g, (select distinct e.emissionId,e.genreId from emission e , programme p
        where e.emissionId=p.emissionId
        group by (e.emissionId,e.genreId)
        having sum(p.nombre_vue)>sum(p.audience_prevue)) tmp
where g.genreId=tmp.genreId(+)
group by (g.genreId,g.nom);
```

19- Quelles sont les émissions qui ont été programmées plus de 10 fois ?

```
SELECT E.nom
FROM EMISSION E
JOIN PROGRAMME P ON E.emissionId = P.emissionId
GROUP BY E.nom
HAVING COUNT(P.programmeId) > 10;
```

20- quelle est la durée moyenne des publicités selon les différentes audiences enregistrées ? Triée par ordre croissant.

```
select pr.nombre_vue,AVG(p.duree) as duree_moy_pub
from publicite p, contient c, programme pr
where p.publiciteId = c.publiciteId and c.programmeId = pr.programmeId
group by pr.nombre_vue
order by duree_moy_pub asc;
```

21- Pour chaque technicien, fournir la liste des émissions sur lesquelles il a travaillé, ainsi que le nombre de programmes où il a participé pour chacune de ces émissions ?

```
select s.salarieId,s.nom,s.prenom,e.nom as nom_emission,count(t.programmeId) as
nb_programmes
from salarie s,technicien t,programme p,emission e
where s.salarieId = t.salarieId and t.programmeId = p.programmeId
and p.emissionId = e.emissionId
group by s.salarieId,s.nom,s.prenom,e.nom;
```

22- Quelles sont les plages horaires ou l'audience est la plus élevée?

```
select DISTINCT
TO_CHAR(p."date", 'HH24:MI:SS') AS plage_debut ,
TO_CHAR(p."date"+ e.duree / 1440, 'HH24:MI:SS') AS plage_fin
from programme p , emission e
where p.emissionId=e.emissionId and
p.nombre_vue=(select max(nombre_vue) from programme);
```

- **Vues et Droit d'accès :**

1) Vue des émissions par type :

```
CREATE OR REPLACE VIEW V_EMISSIONS_PAR_TYPE AS
SELECT type, COUNT(*) AS nombre_emissions
FROM EMISSION
GROUP BY type;
```

2) Vue des animateurs avec leurs émissions et leur audience moyenne.

```
create or replace view V_ANIMATEURS_AUDIENCE as
(select s.salarieId, s.nom, s.prenom, COALESCE(e.emissionId, NULL) as emissionId,
COALESCE(e.nom, 'Aucune émission') as nom_emission,
COALESCE(AVG(p.nombre_vue), 0) as audience_moyenne
from salarie s left join animateur a on s.salarieId = a.salarieId
Left join emission e on a.emissionId = e.emissionId
Left join programme p on e.emissionId = p.emissionId
where s."type" = 'Animateur'
group by s.salarieId, s.nom, s.prenom, e.emissionId, e.nom);
```

3) Durée de total de séquences publicitaires par émissions.

```
create or replace view V_DUREE_PUB_PAR_EMISSION as
(select e.emissionId,e.nom,sum(p.duree) as duree_total_pub
From publicite p join contient c on p.publiciteId = c.publiciteId
Right join programme pr on c.programmeId = pr.programmeId
Right Join emission e on pr.emissionId = e.emissionId
group by e.emissionId,e.nom);
```

4) Durée moyenne de travail journalière des techniciens selon la spécialité.

```
CREATE OR REPLACE VIEW V_DUREE_MOYENNE_TECHNICIENS AS
SELECT t.specialite, p."date", Avg(e.duree) AS duree_moyenne_par_jour
FROM TECHNICIEN t JOIN PROGRAMME p ON t.programmeId = p.programmeId
JOIN EMISSION e ON p.emissionId = e.emissionId
GROUP BY t.specialite, p."date";
```

5)- Liste des salaries ayant travaillé la journée du 'dd/mm/aa'.

```
create view VUE_SALARIES_TRAVAIL_JOUR as (
select s.salarieId,s.nom,s.prenom,s.type
from salarie s
where s.salarieID in(
select a.salarieId
from animateur a , emission e , programme p
where a.emissionId = e.emissionId and e.emissionId=p.programmeId
and p.mode_diffusion='direct' and TO_CHAR(p."date", 'DD/MM/YY') = 'dd/mm/aa'
Union
select t.salarieId
from technicien t ,programme p
where t.programmeId = p.programmeId
and TO_CHAR(p."date", 'DD/MM/YY') = 'dd/mm/aa')
);
```

6) Vue sur toutes les émissions du '01-11-2024' et leur public cible.

```
create or replace view V_EMISSION_DU_JOUR as
(select e.emissionId,e.nom as nom_emission ,e.public_cible as age_min
from programme p, emission e
where p.emissionId = e.emissionId
and TRUNC(p."date") = TO_DATE('2024-11-01', 'YYYY-MM-DD'));
```

- **Droit d'accès :**

1) Administrateur BD : accès complet à toute la base de données

```
CREATE ROLE AdministrateurBD;
GRANT ALL PRIVILEGES ON ALL TABLES TO AdministrateurBD;
GRANT ALL PRIVILEGES ON ALL VIEWS TO AdministrateurBD;
```

2) Programmateur: accès total aux émissions et programmes.

```
CREATE ROLE Programmateur;
GRANT SELECT, INSERT, UPDATE, DELETE ON EMISSION TO Programmateur;
GRANT SELECT, INSERT, UPDATE, DELETE ON PROGRAMME TO Programmateur;
```

3) Service RH : gestion des salaries

```
CREATE ROLE RH;
GRANT SELECT, INSERT, UPDATE, DELETE ON SALARIE TO RH;
GRANT SELECT ON VUE_SALARIES_TRAVAIL_JOUR TO RH;
GRANT SELECT ON V_DUREE_MOYENNE_TECHNICIENS TO RH
```

4) Animateur : accès uniquement aux Vues :

```
CREATE ROLE Animateur;
GRANT SELECT ON VUE_EMISSIONS_PAR_TYPE TO Animateur;
GRANT SELECT ON V_ANIMATEUR_AUDIENCE TO Animateur;
```

5) Audimat :

```
CREATE ROLE Audimat;
GRANT SELECT ON V_ANIMATEUR_AUDIENCE TO Audimat;
GRANT SELECT ON V_DUREE_PUB_PAR_EMISSION TO Audiamat;
```

6) Tout public (userguest) :

```
CREATE ROLE UserGuest;
```

- **Accès en lecture seule aux informations publiques**

```
GRANT SELECT ON emission (nom, type, duree) TO UserGuest;
GRANT SELECT ON programme (mode_diffusion, "date", emissionId) TO UserGuest;
```

- **Bloquer explicitement l'accès aux audiences**

```
REVOKE SELECT ON programme (audience_prevue, nombre_vue) FROM UserGuest;
```

5. Méta-données :

- **Script SQL nommé liste_ora_constraints qui donne une fois exécuté la liste Toutes les contraintes d'intégrité définies sur notre BD** :

liste_ora_constraints.sql

```
SELECT   uc.table_name AS Nom_table,uc.constraint_name AS
Nom_de_la_contrainte,uc.constraint_type AS Type_de_contrainte

CASE
WHEN uc.constraint_type = 'P' THEN 'Clé primaire'
WHEN uc.constraint_type = 'R' THEN 'Clé étrangère'
WHEN uc.constraint_type = 'C' THEN 'Contrainte de vérification'
 ELSE 'Autre'
  END AS "Description du type",
cc.search_condition AS "Corps de la contrainte"
FROM  user_constraints uc
LEFT JOIN user_cons_columns ucc ON uc.constraint_name = ucc.constraint_name
LEFT JOIN user_constraints cc ON ucc.constraint_name = cc.constraint_name
WHERE uc.constraint_type IN ('P', 'R', 'C')
ORDER BY uc.table_name, uc.constraint_type;
```

- **Script SQL nommé liste_ora_triggers qui donne une fois exécuté la liste de tous les triggers classes par le nom de la table :**

liste_ora_triggers.sql:

```
SELECT
  trigger_name AS Nom_du_trigger,
table_name AS Nom_table,
trigger_type AS Type_de_trigger
FROM user_triggers
ORDER BY table_name, trigger_name;
```

- **Scripts supplémentaire :**

- **Script qui donne une fois exécuté la liste de tous les triggers concernant la table "Programme" :**

```
SELECT
  trigger_name AS Nom_du_trigger,
  table_name AS Table_associee,
  trigger_type AS Type_de_trigger,
  trigger_body AS Corps_du_trigger
FROM user_triggers
WHERE UPPER(table_name) = 'PROGRAMME'
ORDER BY table_name, trigger_name;
```

- **Script permet de lister les utilisateurs et leurs privilèges de niveau système dans la base de données**

```
SELECT grantee AS Utilisateur,privilege AS Privilege,
   table_name AS Table_associee, grantor AS Accorde_par
FROM all_tab_privs
WHERE grantee NOT IN ('SYS', 'SYSTEM') and grantee IN (SELECT username FROM all_users)
ORDER BY grantee, privilege;
```

6. Répartition de travail :

Le travail a été effectué en groupe de manière collaborative et chacun des membres a contribué activement à la réalisation des différentes requêtes, vues et triggers, en suivant une répartition précise des tâches. Voici la répartition des tâches entre les membres de l'équipe :

Membre	Requêtes	Vues	Triggers
KHAZEM Lynda	2-5-7-10-16-20-21	2-3	1 -5-7
MACHTER Massinissa	3-8-13-14-17-18-22	4-6	2-3-8
HATEM Nadir	1-4-6-9-11-12-15-19	1-5	4-6

Le reste du travail, a également été réalisé de manière collaborative au sein de l'équipe. Nous avons partagé nos connaissances et nos idées lors de discussions de groupe pour la rédaction des documents ainsi que pour l'organisation générale du projet. Ces échanges ont eu lieu lors des séances de TD ainsi qu'en dehors, en réunion, où nous avons fait en sorte de nous aider mutuellement à avancer de manière cohérente sur toutes les parties du travail.