

Table des matières

[Table des matières 2](#_Toc199192295)

[Introduction : 2](#_Toc199192296)

[Privilèges des Tables : 2](#_Toc199192297)

[Création des Vues : 3](#_Toc199192298)

[FICHE JEU : 3](#_Toc199192299)

[SORTIES RECENTES : 4](#_Toc199192300)

[Création des Fonctions : 4](#_Toc199192301)

[FICHE DETAILLEE : 4](#_Toc199192302)

[MEILLEURS JEUX : 6](#_Toc199192303)

[Création des Procédures : 7](#_Toc199192304)

[AJOUTER DATE SORTIE : 7](#_Toc199192305)

[AJOUTER MODE MULTIJOUEUR : 8](#_Toc199192306)

[Création des Déclencheurs : 9](#_Toc199192307)

[Tables LOG : 9](#_Toc199192308)

[Code Triggers : 9](#_Toc199192309)

[Amélioration du code : 33](#_Toc199192310)

[SR : 33](#_Toc199192311)

[Script création de Table et insertion de donnée : 33](#_Toc199192312)

[DETAIL SORTIES : 35](#_Toc199192313)

# Introduction :

Les triggers ont été inspiré de ce d’Arthur PRUVOST-RIVIÈRE car je n’avais pas réussi à comprendre la manière d’ont-ils faillait les créer dans ce projet.

J’ai également utilisé ChatGPT afin de corriger une erreur de TimeOut sur la vue FICHE\_JEU et une erreur de nombre de caractère maximale pour la fonction FICHE\_DETAILLE. L’erreur de la fonction était sur la construction des JSON avec les JSON\_ARRAYAGG.

# Gestion des droits :

Le code PL/SQL a fonctionné au début mais ne l’est actuellement plus avec une erreur sur les tables BDCAMPING.

BEGIN

FOR obj IN (

SELECT object\_name, object\_type

FROM user\_objects

WHERE object\_type IN ('TABLE', 'VIEW')

) LOOP

-- Droits pour AnalyseJV (lecture sauf LOG)

IF obj.object\_name <> 'LOG' THEN

IF obj.object\_type IN ('TABLE', 'VIEW') THEN

EXECUTE IMMEDIATE 'GRANT SELECT ON ' || obj.object\_name || ' TO AnalyseJV';

END IF;

END IF;

-- Droits pour GestionJV

IF obj.object\_type IN ('TABLE', 'VIEW') THEN

IF obj.object\_name = 'LOG' THEN

EXECUTE IMMEDIATE 'GRANT SELECT ON ' || obj.object\_name || ' TO GestionJV';

ELSE

EXECUTE IMMEDIATE 'GRANT SELECT, INSERT, UPDATE, DELETE ON ' || obj.object\_name || ' TO GestionJV';

END IF;

END IF;

END LOOP;

-- Fonctions/proc�dures pour AnalyseJV (lecture uniquement)

FOR obj IN (

SELECT object\_name

FROM user\_procedures

WHERE object\_type IN ('FUNCTION', 'PROCEDURE')

AND object\_name NOT IN ('PROC1\_MODIF', 'PROC2\_MODIF') -- � adapter

) LOOP

EXECUTE IMMEDIATE 'GRANT EXECUTE ON ' || obj.object\_name || ' TO AnalyseJV';

END LOOP;

-- Fonctions/proc�dures pour GestionJV (acc�s total)

FOR obj IN (

SELECT object\_name

FROM user\_procedures

WHERE object\_type IN ('FUNCTION', 'PROCEDURE')

) LOOP

EXECUTE IMMEDIATE 'GRANT ALL ON ' || obj.object\_name || ' TO GestionJV';

END LOOP;

END;

/

# Vues imposées :

FICHE JEU :

CREATE OR REPLACE VIEW FICHE\_JEU AS

SELECT J.IDJEU AS identifiant, J.TITREJEU AS titre, MIN(DS.DATESORTIE) AS premiere\_date\_sortie, COALESCE(J.STATUTJEU, 'Publié') AS statut,

LISTAGG(DISTINCT CASE WHEN CJ.estDeveloppeur = 1 THEN C.nomcompagnie END, ', ') WITHIN GROUP (ORDER BY C.nomcompagnie) AS compagnies,

LISTAGG(DISTINCT G.nomgenre, ', ') WITHIN GROUP (ORDER BY G.nomgenre) AS genres,

LISTAGG(DISTINCT P.nomplateforme, ', ') WITHIN GROUP (ORDER BY P.nomplateforme) AS plateformes,

ROUND(CAST(J.SCOREIGDB AS NUMBER) / 10, 2) AS score\_utilisateur,

ROUND(CAST(J.SCOREAGREGEJEU AS NUMBER) / 10, 2) AS score\_critique

FROM

jeu J

LEFT JOIN datesortie DS ON J.idjeu = DS.idjeu

LEFT JOIN compagniejeu CJ ON J.idjeu = CJ.idjeu

LEFT JOIN compagnie C ON CJ.idcompagnie = C.idcompagnie

LEFT JOIN genrejeu GJ ON J.idjeu = GJ.idjeu

LEFT JOIN genre G ON GJ.idgenre = G.idgenre

LEFT JOIN plateforme P ON DS.idplateforme = P.idplateforme

GROUP BY

J.idjeu, J.titrejeu, J.statutjeu, J.SCOREIGDB, J.scoreagregeJeu

ORDER BY

J.idJeu ASC;

## SORTIES RECENTES :

CREATE OR REPLACE VIEW SORTIES\_RECENTES AS

SELECT J.idJeu, J.titreJeu, DS.datesortie, LISTAGG(DISTINCT P.nomplateforme, ', ') WITHIN GROUP (ORDER BY P.nomplateforme) AS plateforme

FROM jeu J

JOIN datesortie DS ON ds.idJeu = j.idJeu

JOIN plateforme P ON P.idplateforme = DS.idplateforme

WHERE DS.datesortie <= SYSDATE

GROUP BY J.idJeu, J.titreJeu, DS.datesortie

ORDER BY DS.datesortie DESC, J.titrejeu ASC;

# Fonctions imposées :

## FICHE DETAILLEE :

CREATE OR REPLACE FUNCTION FICHE\_DETAILLEE(p\_id\_jeu IN JEU.IdJeu%TYPE) RETURN CLOB

IS

v\_json CLOB;

v\_count NUMBER;

BEGIN

-- Vérifie si le jeu existe

SELECT COUNT(\*) INTO v\_count FROM JEU WHERE idJeu = p\_id\_jeu;

IF v\_count = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Jeu inexistant');

END IF;

-- Génère le JSON

SELECT JSON\_OBJECT(

'titre' VALUE J.TitreJeu,

'résumé' VALUE J.ResumeJeu,

'mode(s) de jeu' VALUE (

SELECT JSON\_ARRAYAGG(m.NomModalite RETURNING CLOB)

FROM MODALITE M

JOIN MODALITEJEU MJ ON MJ.idModalite = M.idModalite

WHERE MJ.idJeu = J.IdJeu

),

'développeur(s)' VALUE (

SELECT JSON\_ARRAYAGG(

JSON\_OBJECT(

'id' VALUE C.IdCompagnie,

'nom' VALUE C.NomCompagnie

RETURNING CLOB)

ORDER BY C.NomCompagnie

RETURNING CLOB)

FROM COMPAGNIE C

JOIN COMPAGNIEJeu CJ ON C.IdCompagnie = CJ.idCompagnie

WHERE CJ.IdJeu = J.IdJeu AND CJ.estDeveloppeur = 1

),

'publieur(s)' VALUE (

SELECT JSON\_ARRAYAGG(

JSON\_OBJECT(

'id' VALUE C.IdCompagnie,

'nom' VALUE C.NomCompagnie

RETURNING CLOB)

ORDER BY C.NomCompagnie

RETURNING CLOB)

FROM COMPAGNIE C

JOIN COMPAGNIEJeu CJ ON C.IdCompagnie = CJ.idCompagnie

WHERE CJ.IdJeu = J.IdJeu AND CJ.estPublieur = 1

),

'plateforme(s)' VALUE (

SELECT JSON\_ARRAYAGG(

JSON\_OBJECT(

'nom' VALUE P.NomPlateforme,

'date sortie' VALUE DS.DateSortie,

'statut' VALUE NVL(DS.StatutSortie, 'Full release')

RETURNING CLOB)

ORDER BY P.NomPlateforme, DS.DateSortie

RETURNING CLOB)

FROM PLATEFORME P

JOIN DATESORTIE DS ON DS.idPlateforme = P.idPlateforme

WHERE DS.IdJeu = J.IdJeu

),

'score' VALUE J.ScoreJeu,

'nb votes' VALUE J.NombreNotesJeu,

'score critiques' VALUE J.ScoreAgregeJeu,

'nb votes critiques' VALUE J.NombreNotesAgregeesJeu

RETURNING CLOB

) INTO v\_json

FROM JEU J

WHERE J.IdJeu = p\_id\_jeu;

RETURN v\_json;

END;

/

## MEILLEURS JEUX :

CREATE OR REPLACE FUNCTION MEILLEURS\_JEUX(id\_plateforme IN NUMBER)

RETURN CLOB

IS

v\_json CLOB;

v\_count NUMBER;

BEGIN

-- Vérifier que la plateforme existe

SELECT COUNT(\*) INTO v\_count

FROM PLATEFORME

WHERE idPlateforme = id\_plateforme;

IF v\_count = 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Plateforme inexistante');

END IF;

-- Construire le JSON des meilleurs jeux

SELECT JSON\_OBJECT(

'jeux' VALUE JSON\_ARRAYAGG(

JSON\_OBJECT(

'id' VALUE t.idJeu,

'titre' VALUE t.titreJeu,

'popscore' VALUE t.popscore,

'rang' VALUE t.rang

)RETURNING CLOB

) RETURNING CLOB

)

INTO v\_json

FROM (

SELECT

J.idJeu,

J.titreJeu,

(0.5 \* SUM(CASE WHEN P1.mesurePopularite = 'visits' THEN P1.valeurpopularite ELSE 0 END) +

0.25 \* SUM(CASE WHEN P1.mesurePopularite = 'played' THEN P1.valeurpopularite ELSE 0 END) +

0.15 \* SUM(CASE WHEN P1.mesurePopularite = 'playing' THEN P1.valeurpopularite ELSE 0 END) +

0.10 \* SUM(CASE WHEN P1.mesurePopularite = 'Want to Play' THEN P1.valeurpopularite ELSE 0 END)

) AS popscore,

RANK() OVER (

ORDER BY (

0.5 \* SUM(CASE WHEN P1.mesurePopularite = 'visits' THEN P1.valeurpopularite ELSE 0 END) +

0.25 \* SUM(CASE WHEN P1.mesurePopularite = 'played' THEN P1.valeurpopularite ELSE 0 END) +

0.15 \* SUM(CASE WHEN P1.mesurePopularite = 'playing' THEN P1.valeurpopularite ELSE 0 END) +

0.10 \* SUM(CASE WHEN P1.mesurePopularite = 'Want to Play' THEN P1.valeurpopularite ELSE 0 END)

) DESC

) AS rang

FROM JEU J

JOIN DATESORTIE DS ON DS.idJeu = J.idJeu

JOIN PLATEFORME P ON P.idPlateforme = DS.idPlateforme

JOIN POPULARITE P1 ON P1.idJeu = J.idJeu

WHERE P.idPlateforme = id\_plateforme

GROUP BY J.idJeu, J.titreJeu

) t

WHERE t.rang <= 100;

RETURN v\_json;

END;

/

# Création des Procédures :

## AJOUTER DATE SORTIE :

CREATE OR REPLACE PROCEDURE AJOUTER\_DATE\_SORTIE(

p\_idJeu IN NUMBER,

p\_idPlateforme IN NUMBER,

p\_dateSortie IN DATE,

p\_regionSortie IN VARCHAR2,

p\_statut IN VARCHAR2

) IS

v\_count NUMBER;

BEGIN

-- Vérification 1 : Jeu inexistant

SELECT COUNT(\*) INTO v\_count FROM JEU WHERE idJeu = p\_idJeu;

IF v\_count = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Jeu inexistant');

END IF;

-- Vérification 2 : Plateforme inexistante

SELECT COUNT(\*) INTO v\_count FROM PLATEFORME WHERE idPlateforme = p\_idPlateforme;

IF v\_count = 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Plateforme inexistante');

END IF;

-- Vérification 3 : Région inconnue

SELECT COUNT(\*) INTO v\_count FROM REGION WHERE nomRegion = p\_regionSortie;

IF v\_count = 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Region inconnue');

END IF;

-- Vérification 4 : Doublon de sortie (même jeu + plateforme + région)

SELECT COUNT(\*) INTO v\_count FROM DATESORTIE WHERE idJeu = p\_idJeu AND idPlateforme = p\_idPlateforme AND regionSortie = p\_regionSortie;

IF v\_count > 0 THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Sortie deja enregistree');

END IF;

-- Insertion de la nouvelle sortie

INSERT INTO DATESORTIE (idJeu, idPlateforme, dateSortie, regionSortie, statutSortie)

VALUES (p\_idJeu, p\_idPlateforme, p\_dateSortie, p\_regionSortie, p\_statut);

END;

/

## AJOUTER MODE MULTIJOUEUR :

CREATE OR REPLACE PROCEDURE AJOUTER\_MODE\_MULTIJOUEUR (

p\_id\_jeu IN NUMBER,

p\_id\_plateforme IN NUMBER,

p\_drop\_in IN NUMBER,

p\_mode\_coop\_campagne IN NUMBER,

p\_mode\_coop\_lan IN NUMBER,

p\_mode\_coop\_offline IN NUMBER,

p\_mode\_coop\_online IN NUMBER,

p\_mode\_split\_screen IN NUMBER,

p\_nb\_joueurs\_max\_coop\_offline IN NUMBER,

p\_nb\_joueurs\_max\_offline IN NUMBER,

p\_nb\_joueurs\_max\_coop\_online IN NUMBER,

p\_nb\_joueurs\_max\_online IN NUMBER

) IS

p\_count\_jeu NUMBER;

p\_count\_plateforme NUMBER;

p\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO p\_count\_jeu FROM JEU Where idJeu = p\_id\_jeu;

IF p\_count\_jeu = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Jeu inexistant');

END IF;

SELECT COUNT(\*) INTO p\_count\_plateforme FROM PLATEFORME Where idPlateforme = p\_id\_plateforme;

IF p\_count\_plateforme = 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Plateforme inexistante');

END IF;

SELECT 1 INTO p\_count FROM MODEMULTIJOUEUR WHERE idJeu = p\_id\_jeu AND idPlateforme = p\_id\_plateforme;

RAISE\_APPLICATION\_ERROR(-20005, 'Mode Multijoueur deja enregistre');

IF (p\_mode\_coop\_online = 1 AND p\_nb\_joueurs\_max\_coop\_online = 0) OR (p\_mode\_coop\_online = 0 AND p\_nb\_joueurs\_max\_coop\_online > 0) THEN

RAISE\_APPLICATION\_ERROR(-20006, 'Données incohérentes : ModeCoopOnline');

END IF;

IF (p\_mode\_coop\_offline = 1 AND p\_nb\_joueurs\_max\_coop\_offline = 0) OR (p\_mode\_coop\_offline = 0 AND p\_nb\_joueurs\_max\_coop\_offline > 0) THEN

RAISE\_APPLICATION\_ERROR(-20006, 'Données incohérentes : ModeCoopOffline');

END IF;

INSERT INTO modemultijoueur (

idjeu, idplateforme, DropIn, ModeCoopCampagne, ModeCoopLAN,

ModeCoopOffline, ModeCoopOnline, ModeSplitScreen,

NbJoueursMaxCoopOffline, NbJoueursMaxOffline,

NbJoueursMaxCoopOnline, NbJoueursMaxOnline

) VALUES (

p\_id\_jeu, p\_id\_plateforme, p\_drop\_in, p\_mode\_coop\_campagne, p\_mode\_coop\_lan,

p\_mode\_coop\_offline, p\_mode\_coop\_online, p\_mode\_split\_screen,

p\_nb\_joueurs\_max\_coop\_offline, p\_nb\_joueurs\_max\_offline,

p\_nb\_joueurs\_max\_coop\_online, p\_nb\_joueurs\_max\_online

);

END;

/

# Déclencheurs imposés:

## Tables LOG :

DROP TABLE LOG;

CREATE TABLE LOG (

idLog NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

idAuteur VARCHAR2(50) NOT NULL,

action VARCHAR2(20) NOT NULL,

dateHeureAction TIMESTAMP DEFAULT CURRENT\_TIMESTAMP NOT NULL,

idEnregistrement VARCHAR2(100) NOT NULL,

colonneMaj VARCHAR2(100),

valeurAvant VARCHAR2(4000),

valeurApres VARCHAR2(4000),

nomTable VARCHAR2(50) NOT NULL

);

## Code Triggers :

BEGIN

FOR t IN (

SELECT trigger\_name

FROM user\_triggers

) LOOP

EXECUTE IMMEDIATE 'DROP TRIGGER "' || t.trigger\_name || '"';

END LOOP;

END;

/

CREATE OR REPLACE TRIGGER declencheur\_CATEGORIEJEU\_INSERT

AFTER INSERT ON CATEGORIEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdCategorieJeu, NULL, NULL, ':NEW.IdCategorieJeu || ' || ' || :NEW.NomCategoriejeu |', 'CATEGORIEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_CATEGORIEJEU\_UPDATE

AFTER UPDATE ON CATEGORIEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdCategorieJeu, NULL, ':OLD.IdCategorieJeu || ' || ' || :OLD.NomCategoriejeu |', ':NEW.IdCategorieJeu || ' || ' || :NEW.NomCategoriejeu |', 'CATEGORIEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_CATEGORIEJEU\_DELETE

AFTER DELETE ON CATEGORIEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdCategorieJeu, NULL, ':OLD.IdCategorieJeu || ' || ' || :OLD.NomCategoriejeu |', NULL, 'CATEGORIEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_CATEGORIEPLATEFORME\_INSERT

AFTER INSERT ON CATEGORIEPLATEFORME

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdCategoriePlateforme, NULL, NULL, ':NEW.IdCategoriePlateforme || ' || ' || :NEW.NomCategoriePlateforme |', 'CATEGORIEPLATEFORME');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_CATEGORIEPLATEFORME\_UPDATE

AFTER UPDATE ON CATEGORIEPLATEFORME

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdCategoriePlateforme, NULL, ':OLD.IdCategoriePlateforme || ' || ' || :OLD.NomCategoriePlateforme |', ':NEW.IdCategoriePlateforme || ' || ' || :NEW.NomCategoriePlateforme |', 'CATEGORIEPLATEFORME');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_CATEGORIEPLATEFORME\_DELETE

AFTER DELETE ON CATEGORIEPLATEFORME

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdCategoriePlateforme, NULL, ':OLD.IdCategoriePlateforme || ' || ' || :OLD.NomCategoriePlateforme |', NULL, 'CATEGORIEPLATEFORME');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_CLASSIFICATIONAGE\_INSERT

AFTER INSERT ON CLASSIFICATIONAGE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdClassification, NULL, NULL, ':NEW.IdClassification || ' || ' || :NEW.OrganismeClassification || ' || ' || :NEW.Classification || ' || ' || :NEW.SynopsisClassification |', 'CLASSIFICATIONAGE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_CLASSIFICATIONAGE\_UPDATE

AFTER UPDATE ON CLASSIFICATIONAGE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdClassification, NULL, ':OLD.IdClassification || ' || ' || :OLD.OrganismeClassification || ' || ' || :OLD.Classification || ' || ' || :OLD.SynopsisClassification |', ':NEW.IdClassification || ' || ' || :NEW.OrganismeClassification || ' || ' || :NEW.Classification || ' || ' || :NEW.SynopsisClassification |', 'CLASSIFICATIONAGE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_CLASSIFICATIONAGE\_DELETE

AFTER DELETE ON CLASSIFICATIONAGE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdClassification, NULL, ':OLD.IdClassification || ' || ' || :OLD.OrganismeClassification || ' || ' || :OLD.Classification || ' || ' || :OLD.SynopsisClassification |', NULL, 'CLASSIFICATIONAGE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_COMPAGNIE\_INSERT

AFTER INSERT ON COMPAGNIE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdCompagnie, NULL, NULL, ':NEW.IdCompagnie || ' || ' || :NEW.NomCompagnie || ' || ' || :NEW.DescrCompagnie || ' || ' || :NEW.PaysCompagnie || ' || ' || :NEW.DateFondationCompagnie || ' || ' || :NEW.DateMAJCompagnie || ' || ' || :NEW.CompagnieParent |', 'COMPAGNIE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_COMPAGNIE\_UPDATE

AFTER UPDATE ON COMPAGNIE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdCompagnie, NULL, ':OLD.IdCompagnie || ' || ' || :OLD.NomCompagnie || ' || ' || :OLD.DescrCompagnie || ' || ' || :OLD.PaysCompagnie || ' || ' || :OLD.DateFondationCompagnie || ' || ' || :OLD.DateMAJCompagnie || ' || ' || :OLD.CompagnieParent |', ':NEW.IdCompagnie || ' || ' || :NEW.NomCompagnie || ' || ' || :NEW.DescrCompagnie || ' || ' || :NEW.PaysCompagnie || ' || ' || :NEW.DateFondationCompagnie || ' || ' || :NEW.DateMAJCompagnie || ' || ' || :NEW.CompagnieParent |', 'COMPAGNIE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_COMPAGNIE\_DELETE

AFTER DELETE ON COMPAGNIE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdCompagnie, NULL, ':OLD.IdCompagnie || ' || ' || :OLD.NomCompagnie || ' || ' || :OLD.DescrCompagnie || ' || ' || :OLD.PaysCompagnie || ' || ' || :OLD.DateFondationCompagnie || ' || ' || :OLD.DateMAJCompagnie || ' || ' || :OLD.CompagnieParent |', NULL, 'COMPAGNIE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_DATESORTIE\_INSERT

AFTER INSERT ON DATESORTIE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdDateSortie, NULL, NULL, ':NEW.IdDateSortie || ' || ' || :NEW.DateSortie || ' || ' || :NEW.RegionSortie || ' || ' || :NEW.StatutSortie || ' || ' || :NEW.DateMAJDateSortie |', 'DATESORTIE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_DATESORTIE\_UPDATE

AFTER UPDATE ON DATESORTIE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdDateSortie, NULL, ':OLD.IdDateSortie || ' || ' || :OLD.DateSortie || ' || ' || :OLD.RegionSortie || ' || ' || :OLD.StatutSortie || ' || ' || :OLD.DateMAJDateSortie |', ':NEW.IdDateSortie || ' || ' || :NEW.DateSortie || ' || ' || :NEW.RegionSortie || ' || ' || :NEW.StatutSortie || ' || ' || :NEW.DateMAJDateSortie |', 'DATESORTIE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_DATESORTIE\_DELETE

AFTER DELETE ON DATESORTIE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdDateSortie, NULL, ':OLD.IdDateSortie || ' || ' || :OLD.DateSortie || ' || ' || :OLD.RegionSortie || ' || ' || :OLD.StatutSortie || ' || ' || :OLD.DateMAJDateSortie |', NULL, 'DATESORTIE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_FRANCHISE\_INSERT

AFTER INSERT ON FRANCHISE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdFranchise, NULL, NULL, ':NEW.IdFranchise || ' || ' || :NEW.NomFranchise |', 'FRANCHISE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_FRANCHISE\_UPDATE

AFTER UPDATE ON FRANCHISE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdFranchise, NULL, ':OLD.IdFranchise || ' || ' || :OLD.NomFranchise |', ':NEW.IdFranchise || ' || ' || :NEW.NomFranchise |', 'FRANCHISE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_FRANCHISE\_DELETE

AFTER DELETE ON FRANCHISE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdFranchise, NULL, ':OLD.IdFranchise || ' || ' || :OLD.NomFranchise |', NULL, 'FRANCHISE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_JEU\_INSERT

AFTER INSERT ON JEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdJeu, NULL, NULL, ':NEW.IdJeu || ' || ' || :NEW.TitreJeu || ' || ' || :NEW.TitreVersionJeu || ' || ' || :NEW.HistoireJeu || ' || ' || :NEW.ResumeJeu || ' || ' || :NEW.ScoreAgregeJeu || ' || ' || :NEW.NombreNotesAgregeesJeu || ' || ' || :NEW.ScoreIGDB || ' || ' || :NEW.NombreNotesIGDBJeu || ' || ' || :NEW.ScoreJeu || ' || ' || :NEW.NombreNotesJeu || ' || ' || :NEW.TempsJeu\_Normal || ' || ' || :NEW.TempsJeu\_Rapide || ' || ' || :NEW.TempsJeu\_Complet || ' || ' || :NEW.NombreTempsJeu || ' || ' || :NEW.StatutJeu || ' || ' || :NEW.DateMAJJeu || ' || ' || :NEW.VersionParent || ' || ' || :NEW.IdJeuParent || ' || ' || :NEW.FranchisePrincipaleJeu || ' || ' || :NEW.CategorieJeu |', 'JEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_JEU\_UPDATE

AFTER UPDATE ON JEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.TitreJeu || ' || ' || :OLD.TitreVersionJeu || ' || ' || :OLD.HistoireJeu || ' || ' || :OLD.ResumeJeu || ' || ' || :OLD.ScoreAgregeJeu || ' || ' || :OLD.NombreNotesAgregeesJeu || ' || ' || :OLD.ScoreIGDB || ' || ' || :OLD.NombreNotesIGDBJeu || ' || ' || :OLD.ScoreJeu || ' || ' || :OLD.NombreNotesJeu || ' || ' || :OLD.TempsJeu\_Normal || ' || ' || :OLD.TempsJeu\_Rapide || ' || ' || :OLD.TempsJeu\_Complet || ' || ' || :OLD.NombreTempsJeu || ' || ' || :OLD.StatutJeu || ' || ' || :OLD.DateMAJJeu || ' || ' || :OLD.VersionParent || ' || ' || :OLD.IdJeuParent || ' || ' || :OLD.FranchisePrincipaleJeu || ' || ' || :OLD.CategorieJeu |', ':NEW.IdJeu || ' || ' || :NEW.TitreJeu || ' || ' || :NEW.TitreVersionJeu || ' || ' || :NEW.HistoireJeu || ' || ' || :NEW.ResumeJeu || ' || ' || :NEW.ScoreAgregeJeu || ' || ' || :NEW.NombreNotesAgregeesJeu || ' || ' || :NEW.ScoreIGDB || ' || ' || :NEW.NombreNotesIGDBJeu || ' || ' || :NEW.ScoreJeu || ' || ' || :NEW.NombreNotesJeu || ' || ' || :NEW.TempsJeu\_Normal || ' || ' || :NEW.TempsJeu\_Rapide || ' || ' || :NEW.TempsJeu\_Complet || ' || ' || :NEW.NombreTempsJeu || ' || ' || :NEW.StatutJeu || ' || ' || :NEW.DateMAJJeu || ' || ' || :NEW.VersionParent || ' || ' || :NEW.IdJeuParent || ' || ' || :NEW.FranchisePrincipaleJeu || ' || ' || :NEW.CategorieJeu |', 'JEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_JEU\_DELETE

AFTER DELETE ON JEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.TitreJeu || ' || ' || :OLD.TitreVersionJeu || ' || ' || :OLD.HistoireJeu || ' || ' || :OLD.ResumeJeu || ' || ' || :OLD.ScoreAgregeJeu || ' || ' || :OLD.NombreNotesAgregeesJeu || ' || ' || :OLD.ScoreIGDB || ' || ' || :OLD.NombreNotesIGDBJeu || ' || ' || :OLD.ScoreJeu || ' || ' || :OLD.NombreNotesJeu || ' || ' || :OLD.TempsJeu\_Normal || ' || ' || :OLD.TempsJeu\_Rapide || ' || ' || :OLD.TempsJeu\_Complet || ' || ' || :OLD.NombreTempsJeu || ' || ' || :OLD.StatutJeu || ' || ' || :OLD.DateMAJJeu || ' || ' || :OLD.VersionParent || ' || ' || :OLD.IdJeuParent || ' || ' || :OLD.FranchisePrincipaleJeu || ' || ' || :OLD.CategorieJeu |', NULL, 'JEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_GENRE\_INSERT

AFTER INSERT ON GENRE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdGenre, NULL, NULL, ':NEW.IdGenre || ' || ' || :NEW.NomGenre |', 'GENRE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_GENRE\_UPDATE

AFTER UPDATE ON GENRE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdGenre, NULL, ':OLD.IdGenre || ' || ' || :OLD.NomGenre |', ':NEW.IdGenre || ' || ' || :NEW.NomGenre |', 'GENRE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_GENRE\_DELETE

AFTER DELETE ON GENRE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdGenre, NULL, ':OLD.IdGenre || ' || ' || :OLD.NomGenre |', NULL, 'GENRE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MODALITE\_INSERT

AFTER INSERT ON MODALITE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdModalite, NULL, NULL, ':NEW.IdModalite || ' || ' || :NEW.NomModalite |', 'MODALITE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MODALITE\_UPDATE

AFTER UPDATE ON MODALITE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdModalite, NULL, ':OLD.IdModalite || ' || ' || :OLD.NomModalite |', ':NEW.IdModalite || ' || ' || :NEW.NomModalite |', 'MODALITE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MODALITE\_DELETE

AFTER DELETE ON MODALITE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdModalite, NULL, ':OLD.IdModalite || ' || ' || :OLD.NomModalite |', NULL, 'MODALITE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MODEMULTIJOUEUR\_INSERT

AFTER INSERT ON MODEMULTIJOUEUR

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdModeMultiJoueur, NULL, NULL, ':NEW.IdModeMultiJoueur || ' || ' || :NEW.DropIn || ' || ' || :NEW.ModeCoopCampagne || ' || ' || :NEW.ModeCoopLAN || ' || ' || :NEW.ModeCoopOffline || ' || ' || :NEW.ModeCoopOnline || ' || ' || :NEW.ModeSplitScreen || ' || ' || :NEW.NbJoueursMaxCoopOffline || ' || ' || :NEW.NbJoueursMaxOffline || ' || ' || :NEW.NbJoueursMaxCoopOnline || ' || ' || :NEW.NbJoueursMaxOnline || ' || ' || :NEW.IdJeu || ' || ' || :NEW.IdPlateforme |', 'MODEMULTIJOUEUR');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MODEMULTIJOUEUR\_UPDATE

AFTER UPDATE ON MODEMULTIJOUEUR

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdModeMultiJoueur, NULL, ':OLD.IdModeMultiJoueur || ' || ' || :OLD.DropIn || ' || ' || :OLD.ModeCoopCampagne || ' || ' || :OLD.ModeCoopLAN || ' || ' || :OLD.ModeCoopOffline || ' || ' || :OLD.ModeCoopOnline || ' || ' || :OLD.ModeSplitScreen || ' || ' || :OLD.NbJoueursMaxCoopOffline || ' || ' || :OLD.NbJoueursMaxOffline || ' || ' || :OLD.NbJoueursMaxCoopOnline || ' || ' || :OLD.NbJoueursMaxOnline || ' || ' || :OLD.IdJeu || ' || ' || :OLD.IdPlateforme |', ':NEW.IdModeMultiJoueur || ' || ' || :NEW.DropIn || ' || ' || :NEW.ModeCoopCampagne || ' || ' || :NEW.ModeCoopLAN || ' || ' || :NEW.ModeCoopOffline || ' || ' || :NEW.ModeCoopOnline || ' || ' || :NEW.ModeSplitScreen || ' || ' || :NEW.NbJoueursMaxCoopOffline || ' || ' || :NEW.NbJoueursMaxOffline || ' || ' || :NEW.NbJoueursMaxCoopOnline || ' || ' || :NEW.NbJoueursMaxOnline || ' || ' || :NEW.IdJeu || ' || ' || :NEW.IdPlateforme |', 'MODEMULTIJOUEUR');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MODEMULTIJOUEUR\_DELETE

AFTER DELETE ON MODEMULTIJOUEUR

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdModeMultiJoueur, NULL, ':OLD.IdModeMultiJoueur || ' || ' || :OLD.DropIn || ' || ' || :OLD.ModeCoopCampagne || ' || ' || :OLD.ModeCoopLAN || ' || ' || :OLD.ModeCoopOffline || ' || ' || :OLD.ModeCoopOnline || ' || ' || :OLD.ModeSplitScreen || ' || ' || :OLD.NbJoueursMaxCoopOffline || ' || ' || :OLD.NbJoueursMaxOffline || ' || ' || :OLD.NbJoueursMaxCoopOnline || ' || ' || :OLD.NbJoueursMaxOnline || ' || ' || :OLD.IdJeu || ' || ' || :OLD.IdPlateforme |', NULL, 'MODEMULTIJOUEUR');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MOTCLE\_INSERT

AFTER INSERT ON MOTCLE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdMotCle, NULL, NULL, ':NEW.IdMotCle || ' || ' || :NEW.NomMotCle |', 'MOTCLE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MOTCLE\_UPDATE

AFTER UPDATE ON MOTCLE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdMotCle, NULL, ':OLD.IdMotCle || ' || ' || :OLD.NomMotCle |', ':NEW.IdMotCle || ' || ' || :NEW.NomMotCle |', 'MOTCLE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MOTCLE\_DELETE

AFTER DELETE ON MOTCLE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdMotCle, NULL, ':OLD.IdMotCle || ' || ' || :OLD.NomMotCle |', NULL, 'MOTCLE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MOTEUR\_INSERT

AFTER INSERT ON MOTEUR

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdMoteur, NULL, NULL, ':NEW.IdMoteur || ' || ' || :NEW.NomMoteur || ' || ' || :NEW.DescrMoteur || ' || ' || :NEW.DateMAJMoteur |', 'MOTEUR');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MOTEUR\_UPDATE

AFTER UPDATE ON MOTEUR

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdMoteur, NULL, ':OLD.IdMoteur || ' || ' || :OLD.NomMoteur || ' || ' || :OLD.DescrMoteur || ' || ' || :OLD.DateMAJMoteur |', ':NEW.IdMoteur || ' || ' || :NEW.NomMoteur || ' || ' || :NEW.DescrMoteur || ' || ' || :NEW.DateMAJMoteur |', 'MOTEUR');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MOTEUR\_DELETE

AFTER DELETE ON MOTEUR

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdMoteur, NULL, ':OLD.IdMoteur || ' || ' || :OLD.NomMoteur || ' || ' || :OLD.DescrMoteur || ' || ' || :OLD.DateMAJMoteur |', NULL, 'MOTEUR');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_PLATEFORME\_INSERT

AFTER INSERT ON PLATEFORME

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdPlateforme, NULL, NULL, ':NEW.IdPlateforme || ' || ' || :NEW.NomPlateforme || ' || ' || :NEW.AbbreviationPlateforme || ' || ' || :NEW.NomAlternatifPlateforme || ' || ' || :NEW.DescriptifPlateforme || ' || ' || :NEW.GenerationPlateforme || ' || ' || :NEW.IdCategoriePlateforme |', 'PLATEFORME');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_PLATEFORME\_UPDATE

AFTER UPDATE ON PLATEFORME

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdPlateforme, NULL, ':OLD.IdPlateforme || ' || ' || :OLD.NomPlateforme || ' || ' || :OLD.AbbreviationPlateforme || ' || ' || :OLD.NomAlternatifPlateforme || ' || ' || :OLD.DescriptifPlateforme || ' || ' || :OLD.GenerationPlateforme || ' || ' || :OLD.IdCategoriePlateforme |', ':NEW.IdPlateforme || ' || ' || :NEW.NomPlateforme || ' || ' || :NEW.AbbreviationPlateforme || ' || ' || :NEW.NomAlternatifPlateforme || ' || ' || :NEW.DescriptifPlateforme || ' || ' || :NEW.GenerationPlateforme || ' || ' || :NEW.IdCategoriePlateforme |', 'PLATEFORME');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_PLATEFORME\_DELETE

AFTER DELETE ON PLATEFORME

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdPlateforme, NULL, ':OLD.IdPlateforme || ' || ' || :OLD.NomPlateforme || ' || ' || :OLD.AbbreviationPlateforme || ' || ' || :OLD.NomAlternatifPlateforme || ' || ' || :OLD.DescriptifPlateforme || ' || ' || :OLD.GenerationPlateforme || ' || ' || :OLD.IdCategoriePlateforme |', NULL, 'PLATEFORME');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_POPULARITE\_INSERT

AFTER INSERT ON POPULARITE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.MesurePopularite, NULL, NULL, ':NEW.MesurePopularite || ' || ' || :NEW.ValeurPopularite |', 'POPULARITE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_POPULARITE\_UPDATE

AFTER UPDATE ON POPULARITE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.MesurePopularite, NULL, ':OLD.MesurePopularite || ' || ' || :OLD.ValeurPopularite |', ':NEW.MesurePopularite || ' || ' || :NEW.ValeurPopularite |', 'POPULARITE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_POPULARITE\_DELETE

AFTER DELETE ON POPULARITE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.MesurePopularite, NULL, ':OLD.MesurePopularite || ' || ' || :OLD.ValeurPopularite |', NULL, 'POPULARITE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_REGION\_INSERT

AFTER INSERT ON REGION

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdRegion, NULL, NULL, ':NEW.IdRegion || ' || ' || :NEW.NomRegion |', 'REGION');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_REGION\_UPDATE

AFTER UPDATE ON REGION

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdRegion, NULL, ':OLD.IdRegion || ' || ' || :OLD.NomRegion |', ':NEW.IdRegion || ' || ' || :NEW.NomRegion |', 'REGION');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_REGION\_DELETE

AFTER DELETE ON REGION

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdRegion, NULL, ':OLD.IdRegion || ' || ' || :OLD.NomRegion |', NULL, 'REGION');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_THEME\_INSERT

AFTER INSERT ON THEME

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdTheme, NULL, NULL, ':NEW.IdTheme || ' || ' || :NEW.NomTheme |', 'THEME');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_THEME\_UPDATE

AFTER UPDATE ON THEME

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdTheme, NULL, ':OLD.IdTheme || ' || ' || :OLD.NomTheme |', ':NEW.IdTheme || ' || ' || :NEW.NomTheme |', 'THEME');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_THEME\_DELETE

AFTER DELETE ON THEME

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdTheme, NULL, ':OLD.IdTheme || ' || ' || :OLD.NomTheme |', NULL, 'THEME');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_TITREALTERNATIF\_INSERT

AFTER INSERT ON TITREALTERNATIF

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdTitreAlternatif, NULL, NULL, ':NEW.IdTitreAlternatif || ' || ' || :NEW.LibelleTitreAlternatif |', 'TITREALTERNATIF');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_TITREALTERNATIF\_UPDATE

AFTER UPDATE ON TITREALTERNATIF

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdTitreAlternatif, NULL, ':OLD.IdTitreAlternatif || ' || ' || :OLD.LibelleTitreAlternatif |', ':NEW.IdTitreAlternatif || ' || ' || :NEW.LibelleTitreAlternatif |', 'TITREALTERNATIF');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_TITREALTERNATIF\_DELETE

AFTER DELETE ON TITREALTERNATIF

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdTitreAlternatif, NULL, ':OLD.IdTitreAlternatif || ' || ' || :OLD.LibelleTitreAlternatif |', NULL, 'TITREALTERNATIF');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_COMPAGNIEJEU\_INSERT

AFTER INSERT ON COMPAGNIEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdJeu, NULL, NULL, ':NEW.IdJeu || ' || ' || :NEW.IdCompagnie || ' || ' || :NEW.EstDeveloppeur || ' || ' || :NEW.EstPorteur || ' || ' || :NEW.EstPublieur || ' || ' || :NEW.EstSoutien |', 'COMPAGNIEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_COMPAGNIEJEU\_UPDATE

AFTER UPDATE ON COMPAGNIEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdCompagnie || ' || ' || :OLD.EstDeveloppeur || ' || ' || :OLD.EstPorteur || ' || ' || :OLD.EstPublieur || ' || ' || :OLD.EstSoutien |', ':NEW.IdJeu || ' || ' || :NEW.IdCompagnie || ' || ' || :NEW.EstDeveloppeur || ' || ' || :NEW.EstPorteur || ' || ' || :NEW.EstPublieur || ' || ' || :NEW.EstSoutien |', 'COMPAGNIEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_COMPAGNIEJEU\_DELETE

AFTER DELETE ON COMPAGNIEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdCompagnie || ' || ' || :OLD.EstDeveloppeur || ' || ' || :OLD.EstPorteur || ' || ' || :OLD.EstPublieur || ' || ' || :OLD.EstSoutien |', NULL, 'COMPAGNIEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MODALITEJEU\_INSERT

AFTER INSERT ON MODALITEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdJeu, NULL, NULL, ':NEW.IdJeu || ' || ' || :NEW.IdModalite |', 'MODALITEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MODALITEJEU\_UPDATE

AFTER UPDATE ON MODALITEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdModalite |', ':NEW.IdJeu || ' || ' || :NEW.IdModalite |', 'MODALITEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MODALITEJEU\_DELETE

AFTER DELETE ON MODALITEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdModalite |', NULL, 'MODALITEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_GENREJEU\_INSERT

AFTER INSERT ON GENREJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdJeu, NULL, NULL, ':NEW.IdJeu || ' || ' || :NEW.IdGenre |', 'GENREJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_GENREJEU\_UPDATE

AFTER UPDATE ON GENREJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdGenre |', ':NEW.IdJeu || ' || ' || :NEW.IdGenre |', 'GENREJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_GENREJEU\_DELETE

AFTER DELETE ON GENREJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdGenre |', NULL, 'GENREJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_TITREALTERNATIFJEU\_INSERT

AFTER INSERT ON TITREALTERNATIFJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdJeu, NULL, NULL, ':NEW.IdJeu || ' || ' || :NEW.IdTitreAlternatif || ' || ' || :NEW.LibelleTitreAlternatif |', 'TITREALTERNATIFJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_TITREALTERNATIFJEU\_UPDATE

AFTER UPDATE ON TITREALTERNATIFJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdTitreAlternatif || ' || ' || :OLD.LibelleTitreAlternatif |', ':NEW.IdJeu || ' || ' || :NEW.IdTitreAlternatif || ' || ' || :NEW.LibelleTitreAlternatif |', 'TITREALTERNATIFJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_TITREALTERNATIFJEU\_DELETE

AFTER DELETE ON TITREALTERNATIFJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdTitreAlternatif || ' || ' || :OLD.LibelleTitreAlternatif |', NULL, 'TITREALTERNATIFJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_SIMILARITE\_INSERT

AFTER INSERT ON SIMILARITE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdJeu, NULL, NULL, ':NEW.IdJeu || ' || ' || :NEW.IdJeuSimilaire |', 'SIMILARITE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_SIMILARITE\_UPDATE

AFTER UPDATE ON SIMILARITE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdJeuSimilaire |', ':NEW.IdJeu || ' || ' || :NEW.IdJeuSimilaire |', 'SIMILARITE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_SIMILARITE\_DELETE

AFTER DELETE ON SIMILARITE

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdJeuSimilaire |', NULL, 'SIMILARITE');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MOTCLEJEU\_INSERT

AFTER INSERT ON MOTCLEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdJeu, NULL, NULL, ':NEW.IdJeu || ' || ' || :NEW.IdMotCle |', 'MOTCLEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MOTCLEJEU\_UPDATE

AFTER UPDATE ON MOTCLEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdMotCle |', ':NEW.IdJeu || ' || ' || :NEW.IdMotCle |', 'MOTCLEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MOTCLEJEU\_DELETE

AFTER DELETE ON MOTCLEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdMotCle |', NULL, 'MOTCLEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_THEMEJEU\_INSERT

AFTER INSERT ON THEMEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdJeu, NULL, NULL, ':NEW.IdJeu || ' || ' || :NEW.IdTheme |', 'THEMEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_THEMEJEU\_UPDATE

AFTER UPDATE ON THEMEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdTheme |', ':NEW.IdJeu || ' || ' || :NEW.IdTheme |', 'THEMEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_THEMEJEU\_DELETE

AFTER DELETE ON THEMEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdTheme |', NULL, 'THEMEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_CLASSIFICATIONJEU\_INSERT

AFTER INSERT ON CLASSIFICATIONJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdJeu, NULL, NULL, ':NEW.IdJeu || ' || ' || :NEW.IdClassification |', 'CLASSIFICATIONJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_CLASSIFICATIONJEU\_UPDATE

AFTER UPDATE ON CLASSIFICATIONJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdClassification |', ':NEW.IdJeu || ' || ' || :NEW.IdClassification |', 'CLASSIFICATIONJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_CLASSIFICATIONJEU\_DELETE

AFTER DELETE ON CLASSIFICATIONJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdClassification |', NULL, 'CLASSIFICATIONJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MOTEURJEU\_INSERT

AFTER INSERT ON MOTEURJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdJeu, NULL, NULL, ':NEW.IdJeu || ' || ' || :NEW.IdMoteur |', 'MOTEURJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MOTEURJEU\_UPDATE

AFTER UPDATE ON MOTEURJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdMoteur |', ':NEW.IdJeu || ' || ' || :NEW.IdMoteur |', 'MOTEURJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_MOTEURJEU\_DELETE

AFTER DELETE ON MOTEURJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdMoteur |', NULL, 'MOTEURJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_COMPAGNIEMOTEUR\_INSERT

AFTER INSERT ON COMPAGNIEMOTEUR

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdCompagnie, NULL, NULL, ':NEW.IdCompagnie || ' || ' || :NEW.IdMoteur |', 'COMPAGNIEMOTEUR');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_COMPAGNIEMOTEUR\_UPDATE

AFTER UPDATE ON COMPAGNIEMOTEUR

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdCompagnie, NULL, ':OLD.IdCompagnie || ' || ' || :OLD.IdMoteur |', ':NEW.IdCompagnie || ' || ' || :NEW.IdMoteur |', 'COMPAGNIEMOTEUR');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_COMPAGNIEMOTEUR\_DELETE

AFTER DELETE ON COMPAGNIEMOTEUR

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdCompagnie, NULL, ':OLD.IdCompagnie || ' || ' || :OLD.IdMoteur |', NULL, 'COMPAGNIEMOTEUR');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_PLATEFORMEMOTEUR\_INSERT

AFTER INSERT ON PLATEFORMEMOTEUR

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdMoteur, NULL, NULL, ':NEW.IdMoteur || ' || ' || :NEW.IdPlateforme |', 'PLATEFORMEMOTEUR');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_PLATEFORMEMOTEUR\_UPDATE

AFTER UPDATE ON PLATEFORMEMOTEUR

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdMoteur, NULL, ':OLD.IdMoteur || ' || ' || :OLD.IdPlateforme |', ':NEW.IdMoteur || ' || ' || :NEW.IdPlateforme |', 'PLATEFORMEMOTEUR');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_PLATEFORMEMOTEUR\_DELETE

AFTER DELETE ON PLATEFORMEMOTEUR

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdMoteur, NULL, ':OLD.IdMoteur || ' || ' || :OLD.IdPlateforme |', NULL, 'PLATEFORMEMOTEUR');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_FRANCHISEJEU\_INSERT

AFTER INSERT ON FRANCHISEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdJeu, NULL, NULL, ':NEW.IdJeu || ' || ' || :NEW.IdFranchise |', 'FRANCHISEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_FRANCHISEJEU\_UPDATE

AFTER UPDATE ON FRANCHISEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdFranchise |', ':NEW.IdJeu || ' || ' || :NEW.IdFranchise |', 'FRANCHISEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_FRANCHISEJEU\_DELETE

AFTER DELETE ON FRANCHISEJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdFranchise |', NULL, 'FRANCHISEJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_LOCALISATIONJEU\_INSERT

AFTER INSERT ON LOCALISATIONJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'INSERT', SYSTIMESTAMP, :NEW.IdJeu, NULL, NULL, ':NEW.IdJeu || ' || ' || :NEW.IdRegion || ' || ' || :NEW.TitreLocalise |', 'LOCALISATIONJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_LOCALISATIONJEU\_UPDATE

AFTER UPDATE ON LOCALISATIONJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'UPDATE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdRegion || ' || ' || :OLD.TitreLocalise |', ':NEW.IdJeu || ' || ' || :NEW.IdRegion || ' || ' || :NEW.TitreLocalise |', 'LOCALISATIONJEU');

END;

/

CREATE OR REPLACE TRIGGER declencheur\_LOCALISATIONJEU\_DELETE

AFTER DELETE ON LOCALISATIONJEU

FOR EACH ROW

BEGIN

INSERT INTO LOG(idAuteur, action, dateHeureAction, idEnregistrement, colonneMaj, valeurAvant, valeurApres, nomTable)

VALUES(user, 'DELETE', SYSTIMESTAMP, :OLD.IdJeu, NULL, ':OLD.IdJeu || ' || ' || :OLD.IdRegion || ' || ' || :OLD.TitreLocalise |', NULL, 'LOCALISATIONJEU');

END;

/

**Seconde amélioration**:

## SR :

J’ai choisi d’ajouter les tables permettant de créer les 3 jeux qui m’ont été attribuer.

Une image contenant texte, Police, ligne, nombre

Le contenu généré par l’IA peut être incorrect.

## Script création de Table et insertion de donnée :

J’ai décidé de séparer les tables et insertions afin que ce soit plus lisible et plus simple à comprendre.

DROP TABLE DEVELOPPEURJEU;

DROP TABLE DEVELOPPEUR;

DROP TABLE SERIEJEU;

DROP TABLE SERIEE;

DROP TABLE PERSPECTIVEJOUEURJEU;

DROP TABLE PERSPECTIVEJOUEUR;

DROP TABLE EDITEURJEU;

DROP TABLE EDITEURE;

CREATE TABLE DEVELOPPEUR (

idDeveloppeur NUMBER PRIMARY KEY,

nomDeveloppeur VARCHAR2(255)

);

CREATE TABLE DEVELOPPEURJEU (

idDeveloppeur NUMBER,

idJeu NUMBER,

FOREIGN KEY (idDeveloppeur) REFERENCES Developpeur(idDeveloppeur),

FOREIGN KEY (idJeu) REFERENCES Jeu(idJeu)

);

CREATE TABLE SERIEE (

idSerie NUMBER PRIMARY KEY,

nomSerie VARCHAR2(255)

);

CREATE TABLE SERIEJEU (

idSerie NUMBER,

idJeu NUMBER,

FOREIGN KEY (idSerie) REFERENCES SERIEE(idSerie),

FOREIGN KEY (idJeu) REFERENCES Jeu(idJeu)

);

CREATE TABLE PERSPECTIVEJOUEUR (

idPerspectiveJoueur NUMBER PRIMARY KEY,

nomPerspectiveJoueur VARCHAR2(255)

);

CREATE TABLE PERSPECTIVEJOUEURJEU (

idPerspectiveJoueur NUMBER,

idJeu NUMBER,

FOREIGN KEY (idPerspectiveJoueur) REFERENCES PERSPECTIVEJOUEUR(idPerspectiveJoueur),

FOREIGN KEY (idJeu) REFERENCES Jeu(idJeu)

);

CREATE TABLE EDITEURE (

idEditeur NUMBER PRIMARY KEY,

nomEditeur VARCHAR2(255)

);

CREATE TABLE EDITEURJEU (

idEditeur NUMBER,

idJeu NUMBER,

FOREIGN KEY (idEditeur) REFERENCES EDITEURE(idEditeur),

FOREIGN KEY (idJeu) REFERENCES Jeu(idJeu)

);

SELECT \* FROM GENREJEU GJ

JOIN GENRE G ON G.idGenre = GJ.idGenre

WHERE idJeu = 4035;

// Donne Street FIGHTER II

INSERT INTO EDITEURE (idEditeur, nomEditeur) VALUES (1, 'Capcom');

INSERT INTO EDITEURE (idEditeur, nomEditeur) VALUES (2, 'Playtronic');

INSERT INTO EDITEURJEU (idEditeur, idJeu) VALUES (1, 39306);

INSERT INTO EDITEURJEU (idEditeur, idJeu) VALUES (2, 39306);

INSERT INTO PERSPECTIVEJOUEUR (idPerspectiveJoueur, nomPerspectiveJoueur) VALUES (1, 'Side view');

INSERT INTO PERSPECTIVEJOUEURJEU (idPerspectiveJoueur, idJeu) VALUES (1, 39306);

INSERT INTO SERIEE (idSerie, nomSerie) VALUES (1, 'Street Fighter II');

INSERT INTO SERIEE (idSerie, nomSerie) VALUES (2, 'Street Fighter');

INSERT INTO SERIEJEU (idSerie, idJeu) VALUES (1, 39306);

INSERT INTO SERIEJEU (idSerie, idJeu) VALUES (2, 39306);

// Donnee Rapala Fishing Franzy 2009

INSERT INTO DEVELOPPEUR (idDeveloppeur, nomDeveloppeur) VALUES (1, 'FUN Labs');

INSERT INTO DEVELOPPEURJEU (idDeveloppeur, idJeu) VALUES (1, 7155);

INSERT INTO EDITEURE (idEditeur, nomEditeur) VALUES (3, 'Activision');

INSERT INTO EDITEURJEU (idEditeur, idJeu) VALUES (3, 7155);

// Donnee NBA Street

INSERT INTO DEVELOPPEUR (idDeveloppeur, nomDeveloppeur) VALUES (2, 'EA Canada');

INSERT INTO DEVELOPPEUR (idDeveloppeur, nomDeveloppeur) VALUES (3, 'NuFX');

INSERT INTO DEVELOPPEURJEU (idDeveloppeur, idJeu) VALUES (2, 4035);

INSERT INTO DEVELOPPEURJEU (idDeveloppeur, idJeu) VALUES (3, 4035);

INSERT INTO EDITEURE (idEditeur, nomEditeur) VALUES (4, 'EA Sports BIG');

INSERT INTO EDITEURE (idEditeur, nomEditeur) VALUES (5, 'Square Electronic Arts');

INSERT INTO EDITEURJEU (idEditeur, idJeu) VALUES (4, 4035);

INSERT INTO EDITEURJEU (idEditeur, idJeu) VALUES (5, 4035);

INSERT INTO PERSPECTIVEJOUEUR (idPerspectiveJoueur, nomPerspectiveJoueur) VALUES (2, 'Third person');

INSERT INTO PERSPECTIVEJOUEURJEU (idPerspectiveJoueur, idJeu) VALUES (2, 4035);

INSERT INTO SERIEE (idSerie, nomSerie) VALUES (3, 'NBA Street');

INSERT INTO SERIEJEU (idSerie, idJeu) VALUES (3, 4035);

## DETAIL SORTIES :

CREATE OR REPLACE FUNCTION DETAIL\_SORTIES(p\_idJeu IN NUMBER)

RETURN CLOB

IS

v\_json CLOB;

BEGIN

SELECT JSON\_OBJECT(

'idJeu' VALUE j.idJeu,

'titre' VALUE j.titreJeu,

'plateformes' VALUE (

SELECT JSON\_ARRAYAGG(

JSON\_OBJECT(

'nom' VALUE p.nomPlateforme,

'regions' VALUE (

SELECT JSON\_ARRAYAGG(

JSON\_OBJECT(

'nom' VALUE r.nomRegion,

'dates' VALUE (

SELECT JSON\_ARRAYAGG(

JSON\_OBJECT(

'date' VALUE ds.dateSortie,

'statut' VALUE ds.statutSortie

) ORDER BY ds.dateSortie

)

FROM DATESORTIE DS2

JOIN LOCALISATIONJEU LJ2 ON LJ2.idJeu = DS2.idJeu

WHERE DS2.idJeu = j.idJeu

AND DS2.idPlateforme = DS.idPlateforme

AND LJ2.idRegion = LJ.idRegion

)

) ORDER BY r.nomRegion

)

FROM DATESORTIE ds

JOIN LOCALISATIONJEU LJ ON LJ.idJeu = DS.idJeu

JOIN REGION R ON R.idRegion = LJ.idRegion

WHERE ds.idJeu = j.idJeu AND ds.idPlateforme = p.idPlateforme

),

'developpeurs' VALUE (

SELECT JSON\_ARRAYAGG(cj.idCompagnie ORDER BY cj.idCompagnie)

FROM COMPAGNIEJEU cj

WHERE cj.idJeu = j.idJeu AND cj.EstDeveloppeur = 1

),

'porteurs' VALUE (

SELECT JSON\_ARRAYAGG(cj.idCompagnie ORDER BY cj.idCompagnie)

FROM COMPAGNIEJEU cj

WHERE cj.idJeu = j.idJeu AND cj.EstPorteur = 1

),

'publieurs' VALUE (

SELECT JSON\_ARRAYAGG(cj.idCompagnie ORDER BY cj.idCompagnie)

FROM COMPAGNIEJEU cj

WHERE cj.idJeu = j.idJeu AND cj.EstPublieur = 1

),

'soutiens' VALUE (

SELECT JSON\_ARRAYAGG(cj.idCompagnie ORDER BY cj.idCompagnie)

FROM COMPAGNIEJEU cj

WHERE cj.idJeu = j.idJeu AND cj.EstSoutien = 1

)

)

ORDER BY (

SELECT MIN(ds.dateSortie)

FROM DATESORTIE ds

WHERE ds.idJeu = j.idJeu AND ds.idPlateforme = p.idPlateforme

)

)

FROM PLATEFORME p

WHERE EXISTS (

SELECT 1 FROM DATESORTIE ds

WHERE ds.idJeu = j.idJeu AND ds.idPlateforme = p.idPlateforme

)

)

)

INTO v\_json

FROM JEU j

WHERE j.idJeu = p\_idJeu;

RETURN v\_json;

END;

/