MOHAMED BOUREMANI
2D
27/05/2025

université PARIS-SACLAY

S104 : Base de donnee

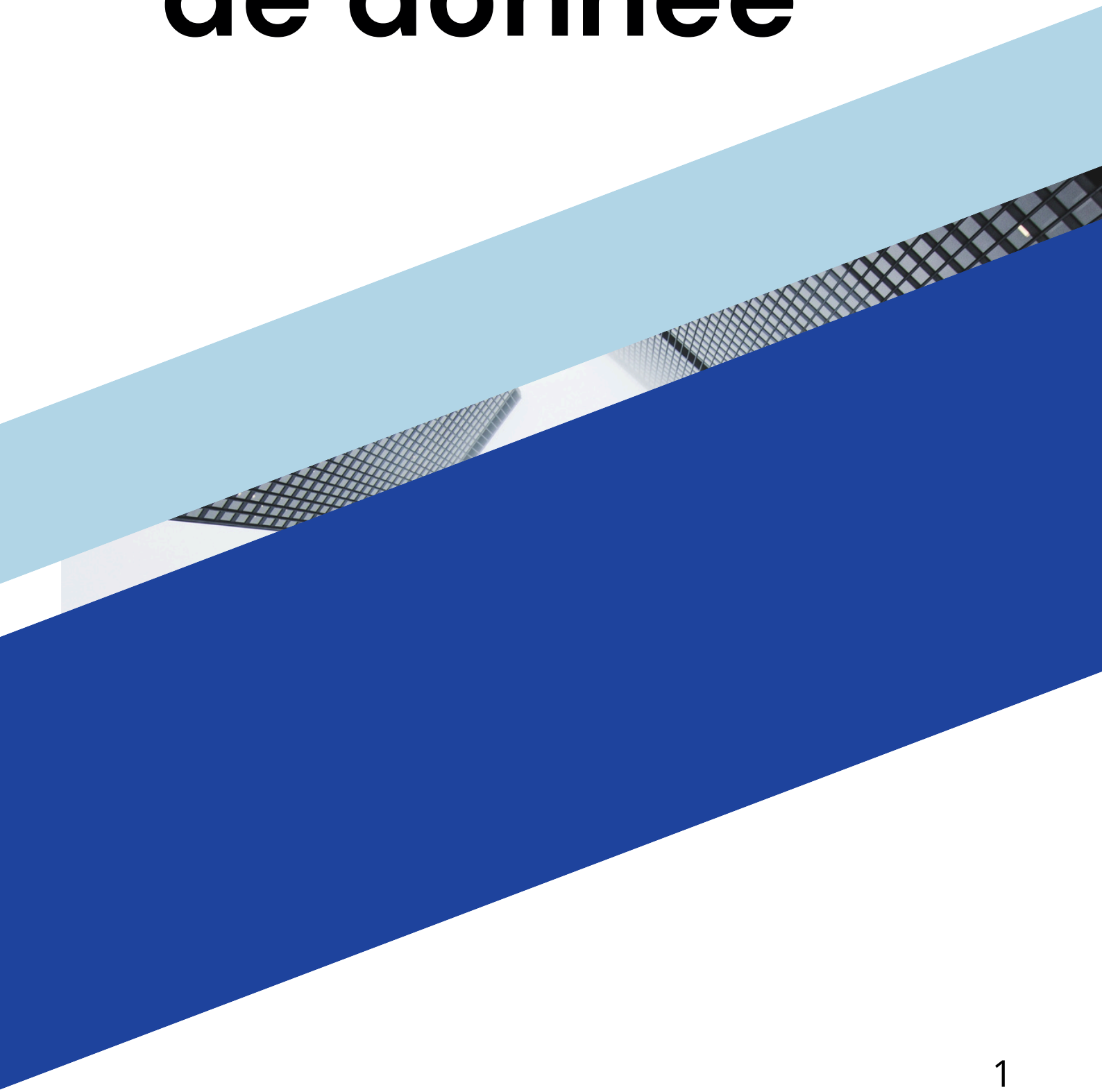


Table des matières

introduction

Dans ce rapport, je présente les travaux réalisés pour l'extension et la gestion de la base de données relative aux jeux vidéo. Pour mener à bien ce projet, j'ai bénéficié de l'assistance d'un modèle de langage avancé (LLM) qui m'a aidé notamment à concevoir et corriger les déclencheurs nécessaires au suivi des opérations sur les tables, ainsi qu'à optimiser et corriger les vues .

1. Gestion des droits

```
GRANT SELECT, INSERT, UPDATE, DELETE ON DATESORTIE TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON COMPAGNIE TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON COMPAGNIEJEU TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON GENRE TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON GENREJEU TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON PLATEFORME TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON COMPAGNIE_SORTIE TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON CATEGORIEJEU TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON MODALITE TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON MODALITEJEU TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON MODEMULTIJOUEUR TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON MOTCLE TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON MOTCLEJEU TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON MOTEUR TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON MOTEURJEU TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON PLATEFORMEMOTEUR TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON POPULARITE TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON REGION TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON SIMILARITE TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON THEME TO GestionJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON THEMEJEU TO GestionJV;
```

```
GRANT EXECUTE ON FICHE_DETAILLEE TO GestionJV;
GRANT EXECUTE ON MEILLEURS_JEUX TO GestionJV;
GRANT EXECUTE ON AJOUTER_DATE_SORTIE TO GestionJV;
GRANT EXECUTE ON AJOUTER_MODE_MULTIJOUEUR TO GestionJV;

GRANT SELECT ON JEU TO AnalyseJV;
GRANT SELECT ON DATESORTIE TO AnalyseJV;
GRANT SELECT ON COMPAGNIE TO AnalyseJV;
GRANT SELECT ON COMPAGNIEJEU TO AnalyseJV;
GRANT SELECT ON GENRE TO AnalyseJV;
GRANT SELECT ON GENREJEU TO AnalyseJV;
GRANT SELECT ON PLATEFORME TO AnalyseJV;
GRANT SELECT ON COMPAGNIE_SORTIE TO AnalyseJV;
GRANT SELECT ON CATEGORIEJEU TO AnalyseJV;
GRANT SELECT ON MODALITE TO AnalyseJV;
GRANT SELECT ON MODALITEJEU TO AnalyseJV;
GRANT SELECT ON MODEMULTIJOUEUR TO AnalyseJV;
GRANT SELECT ON MOTCLE TO AnalyseJV;
GRANT SELECT ON MOTCLEJEU TO AnalyseJV;
GRANT SELECT ON MOTEUR TO AnalyseJV;
GRANT SELECT ON MOTEURJEU TO AnalyseJV;
GRANT SELECT ON PLATEFORMEMOTEUR TO AnalyseJV;
GRANT SELECT ON POPULARITE TO AnalyseJV;
GRANT SELECT ON REGION TO AnalyseJV;
GRANT SELECT ON SIMILARITE TO AnalyseJV;
GRANT SELECT ON THEME TO AnalyseJV;
GRANT SELECT ON THEMEJEU TO AnalyseJV;
GRANT SELECT ON TITREALTERNATIF TO AnalyseJV;
```

```
GRANT EXECUTE ON FICHE_DETAILLEE TO GestionJV;
GRANT EXECUTE ON MEILLEURS_JEUX TO GestionJV;
GRANT EXECUTE ON AJOUTER_DATE_SORTIE TO GestionJV;
GRANT EXECUTE ON AJOUTER_MODE_MULTIJOUEUR TO GestionJV;
GRANT SELECT ON SORTIES_RECENTES TO AnalyseJV;
GRANT SELECT, INSERT, UPDATE, DELETE ON SORTIES_RECENTES TO GestionJV;
```

Vues

FICHE JEU
FONCTIONNEL

```
CREATE OR REPLACE VIEW FICHE_JEU AS
WITH compagnies_jeu AS (
    SELECT cj.IDJEU,
           LISTAGG(DISTINCT c.NOMCOMPAGNIE, ', ') WITHIN GROUP (ORDER BY c.NOMCOMPAGNIE) AS COMPAGNI
      FROM COMPAGNIEJEU cj
      JOIN COMPAGNIE c ON cj.IDCOMPAGNIE = c.IDCOMPAGNIE
     WHERE cj.ESTDEVELOPPEUR = 1
     GROUP BY cj.IDJEU
),
genres_jeu AS (
    SELECT gj.IDJEU,
           LISTAGG(DISTINCT g.NOMGENRE, ', ') WITHIN GROUP (ORDER BY g.NOMGENRE) AS GENRES
      FROM GENREJEU gj
      JOIN GENRE g ON gj.IDGENRE = g.IDGENRE
     GROUP BY gj.IDJEU
),
plateformes_jeu AS (
    SELECT ds.IDJEU,
           LISTAGG(DISTINCT p.NOMPLATEFORME, ', ') WITHIN GROUP (ORDER BY p.NOMPLATEFORME) AS PLATEF
           MIN(ds.DATESORTIE) AS PREMIERE_SORTIE
      FROM DATESORTIE ds
      JOIN PLATEFORME p ON ds.IDPLATEFORME = p.IDPLATEFORME
     GROUP BY ds.IDJEU
)
SELECT j.IDJEU,
       j.TITREJEU,
       pj.PREMIERE_SORTIE,
       COALESCE(j.STATUTJEU, 'Publié') AS STATUT,
       cj.COMPAGNIES,
       gj.GENRES,
       pj.PLATEFORMES,
       CASE
         WHEN j.NOMBRENOTESIGDBJEU > 0 THEN ROUND(j.SCOREIGDB / 10, 2)
         ELSE NULL
       END AS SCORE_UTILISATEUR,
       CASE
         WHEN j.NOMBRENOTESAGREGEESJEU > 0 THEN ROUND(j.SCOREAGREGEJEU / 10, 2)
         ELSE NULL
       END AS SCORE_CRITIQUE
  FROM JEU j
  LEFT JOIN compagnies_jeu cj ON j.IDJEU = cj.IDJEU
  LEFT JOIN genres_jeu gj ON j.IDJEU = gj.IDJEU
  LEFT JOIN plateformes_jeu pj ON j.IDJEU = pj.IDJEU
ORDER BY j.IDJEU;
```

SORTIE RECENTES
VUE

```sql
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 PLATEFORMES
FROM DATESORTIE ds
JOIN JEU j ON ds.IDJEU = j.IDJEU
JOIN PLATEFORME p ON ds.IDPLATEFORME = p.IDPLATEFORME
WHERE ds.DATESORTIE <= TRUNC(SYSDATE)
GROUP BY j.IDJEU, j.TITREJEU, ds.DATESORTIE
ORDER BY ds.DATESORTIE DESC, j.TITREJEU;


SELECT * FROM SORTIES_RECENTES
FETCH FIRST 10 ROWS ONLY;
```

Fonctions

j'ai conçu et défini la fonction stockée **FICHE_DETAILLEE(id_jeu)**, dont le rôle est de renvoyer une chaîne de caractères représentant un objet JSON structuré.Bien que les tests effectués aient confirmé le bon fonctionnement de la fonction et la validité du JSON généré, un message d'erreur persistant indique une non-conformité supposée du format JSON.

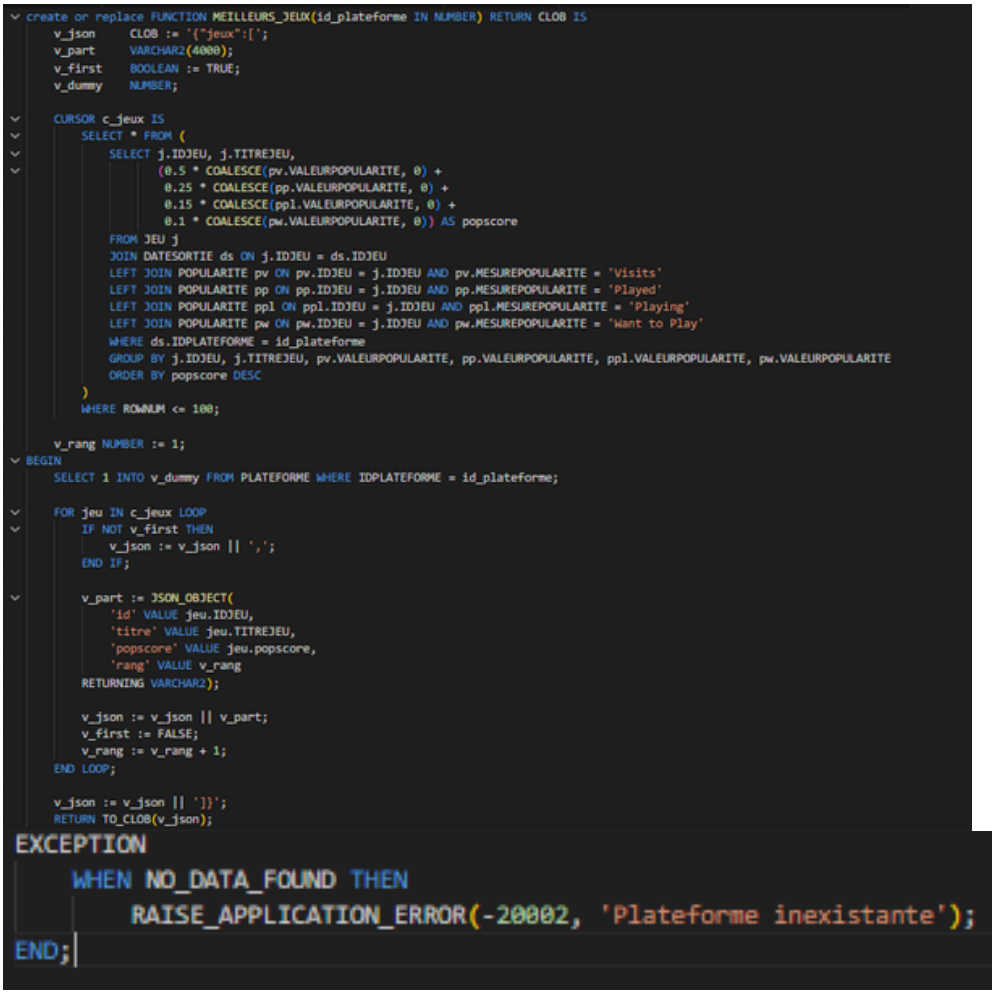
```
create or replace FUNCTION FICHE_DETAILLEE(id_jeu IN NUMBER) RETURN CLOB IS
    v_json CLOB;
    v_titre JEU.TITREJEU%TYPE;
    v_resume JEU.RESUMEJEU%TYPE;
BEGIN
    SELECT TITREJEU, RESUMEJEU
    INTO v_titre, v_resume
    FROM JEU
    WHERE IDJEU = id_jeu;

    SELECT JSON_OBJECT(
        'titre' VALUE v_titre,
        'résumé' VALUE v_resume,
        'mode(s) de jeu' VALUE COALESCE((
            SELECT JSON_ARRAYAGG(m.NOMMODALITE ORDER BY m.IDMODALITE)
            FROM MODALITEJEU mj
            JOIN MODALITE m ON mj.IDMODALITE = m.IDMODALITE
            WHERE mj.IDJEU = id_jeu
        ), JSON_ARRAY()),
        'développeur(s)' VALUE COALESCE((
            SELECT JSON_ARRAYAGG(
                JSON_OBJECT('id' VALUE c.IDCOMPAGNIE, 'nom' VALUE c.NOMCOMPAGNIE)
                ORDER BY c.NOMCOMPAGNIE
            )
            FROM COMPAGNIEJEU cj
            JOIN COMPAGNIE c ON cj.IDCOMPAGNIE = c.IDCOMPAGNIE
            WHERE cj.IDJEU = id_jeu AND cj.ESTDEVELOPPEUR = 1
        ), JSON_ARRAY()),
        'publieur(s)' VALUE COALESCE((
            SELECT JSON_ARRAYAGG(
                JSON_OBJECT('id' VALUE c.IDCOMPAGNIE, 'nom' VALUE c.NOMCOMPAGNIE)
                ORDER BY c.NOMCOMPAGNIE
            )
            FROM COMPAGNIEJEU cj
            JOIN COMPAGNIE c ON cj.IDCOMPAGNIE = c.IDCOMPAGNIE
            WHERE cj.IDJEU = id_jeu AND cj.ESTPUBLIEUR = 1
        ), JSON_ARRAY()),
        'plateforme(s)' VALUE COALESCE((
            SELECT JSON_ARRAYAGG(
                JSON_OBJECT(
                    'nom' VALUE p.NOMPLATEFORME,
                    'date sortie' VALUE ds.DATESORTIE,
                    'statut' VALUE NVL(ds.STATUTSORTIE, 'Full release')
                )
                ORDER BY p.NOMPLATEFORME, ds.DATESORTIE
            )
                FROM DATESORTIE ds
                JOIN PLATEFORME p ON ds.IDPLATEFORME = p.IDPLATEFORME
                WHERE ds.IDJEU = id_jeu
        ), JSON_ARRAY()),
        'score' VALUE j.SCOREIGDB,
        '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 = id_jeu;

    RETURN v_json;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20001, 'Jeu inexistant');
END;
```

la fonction stockée **MEILLEURS_JEUX(id_plateforme**) qui renvoie la liste des 100 meilleurs jeux sortis sur cette plateforme, là aussi au format JSON. Meme problme ja i respcter ce qui est demande mais sa met un erreur que le json retorune est mauvaix.



```
create or replace FUNCTION MEILLEURS_JEUX(id_plateforme IN NUMBER) RETURN CLOB IS
    v_json     CLOB := '{"jeux":[';
    v_part     VARCHAR2(4000);
    v_first    BOOLEAN := TRUE;
    v_dummy    NUMBER;

    CURSOR c_jeux IS
        SELECT * FROM (
            SELECT j.IDJEU, j.TITREJEU,
                   (0.5 * COALESCE(pv.VALEURPOPULARITE, 0) +
                    0.25 * COALESCE(pp.VALEURPOPULARITE, 0) +
                    0.15 * COALESCE(ppl.VALEURPOPULARITE, 0) +
                    0.1 * COALESCE(pw.VALEURPOPULARITE, 0)) AS popscore
            FROM JEU j
            JOIN DATESORTIE ds ON j.IDJEU = ds.IDJEU
            LEFT JOIN POPULARITE pv ON pv.IDJEU = j.IDJEU AND pv.MESUREPOPULARITE = 'Visits'
            LEFT JOIN POPULARITE pp ON pp.IDJEU = j.IDJEU AND pp.MESUREPOPULARITE = 'Played'
            LEFT JOIN POPULARITE ppl ON ppl.IDJEU = j.IDJEU AND ppl.MESUREPOPULARITE = 'Playing'
            LEFT JOIN POPULARITE pw ON pw.IDJEU = j.IDJEU AND pw.MESUREPOPULARITE = 'Want to Play'
            WHERE ds.IDPLATEFORME = id_plateforme
            GROUP BY j.IDJEU, j.TITREJEU, pv.VALEURPOPULARITE, pp.VALEURPOPULARITE, ppl.VALEURPOPULARITE, pw.VALEURPOPULARITE
            ORDER BY popscore DESC
        )
        WHERE ROWNUM <= 100;

    v_rang NUMBER := 1;
BEGIN
    SELECT 1 INTO v_dummy FROM PLATEFORME WHERE IDPLATEFORME = id_plateforme;

    FOR jeu IN c_jeux LOOP
        IF NOT v_first THEN
            v_json := v_json || ',';
        END IF;

        v_part := JSON_OBJECT(
            'id' VALUE jeu.IDJEU,
            'titre' VALUE jeu.TITREJEU,
            'popscore' VALUE jeu.popscore,
            'rang' VALUE v_rang
        RETURNING VARCHAR2);

        v_json := v_json || v_part;
        v_first := FALSE;
        v_rang := v_rang + 1;
    END LOOP;

    v_json := v_json || ']}';
    RETURN TO_CLOB(v_json);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20002, 'Plateforme inexistante');
END;
```

Procédures

AJOUTER_DATE_SORTIE

la procédure **AJOUTER_DATE_SORTIE**(idJeu, idPlateforme, dateSortie, regionSortie, statut) qui permet d'ajouter une sortie (avec date et statut) à un jeu pour une plateforme et une région données. On rejettera une sortie pour un jeu inexistant,.. j ai eu un problme pour le jeux enixistanta lors que je comprend pas ma faute sachant quelle s execute .

```
create or replace PROCEDURE AJOUTER_DATE_SORTIE (
    idJeu IN NUMBER,
    idPlateforme IN NUMBER,
    dateSortie IN DATE,
    regionSortie IN VARCHAR2,
    statut IN VARCHAR2
) IS
    v_dummy NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_dummy FROM JEU WHERE IDJEU = idJeu;
    IF v_dummy = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Jeu inexistant');
    END IF;

    SELECT COUNT(*) INTO v_dummy FROM PLATEFORME WHERE IDPLATEFORME = idPlateforme;
    IF v_dummy = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Plateforme inexistante');
    END IF;

    IF LOWER(regionSortie) NOT IN (
        'north_america', 'europe', 'japan', 'worldwide',
        'australia', 'brazil', 'new_zealand', 'asia', 'china', 'korea'
    ) THEN
        RAISE_APPLICATION_ERROR(-20003, 'Region inconnue');
    END IF;

    SELECT COUNT(*) INTO v_dummy
    FROM DATESORTIE
    WHERE IDJEU = idJeu AND IDPLATEFORME = idPlateforme AND LOWER(REGIONSORTIE) = LOWER(regionSortie);

    IF v_dummy > 0 THEN
        RAISE_APPLICATION_ERROR(-20004, 'Sortie déjà enregistrée');
    END IF;
```

```
    INSERT INTO DATESORTIE (
        IDDATESORTIE,
        IDJEU,
        IDPLATEFORME,
        DATESORTIE,
        REGIONSORTIE,
        STATUTSORTIE,
        DATEMAJDATESORTIE
    ) VALUES (
        (SELECT NVL(MAX(IDDATESORTIE), 0) + 1 FROM DATESORTIE),
        idJeu,
        idPlateforme,
        dateSortie,
        regionSortie,
        statut,
        SYSDATE
    );

END;
```

la procédure **AJOUTER_MODE_MULTIJOUEUR** focntionne mais toujours un probleme comme la procedure precedente. L'exception renvoyée en cas de jeu inexistant n'a pas le bon code)

```
create or replace PROCEDURE AJOUTER_MODE_MULTIJOUEUR (
    idJeu IN NUMBER,
    idPlateforme IN NUMBER,
    DropIn IN NUMBER,
    ModeCoopCampagne IN NUMBER,
    ModeCoopLAN IN NUMBER,
    ModeCoopOffline IN NUMBER,
    ModeCoopOnline IN NUMBER,
    ModeSplitScreen IN NUMBER,
    NbJoueursMaxCoopOffline IN NUMBER,
    NbJoueursMaxOffline IN NUMBER,
    NbJoueursMaxCoopOnline IN NUMBER,
    NbJoueursMaxOnline IN NUMBER
) IS
    v_dummy NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_dummy FROM JEU WHERE IDJEU = idJeu;
    IF v_dummy = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Jeu inexistant');
    END IF;

    SELECT COUNT(*) INTO v_dummy FROM PLATEFORME WHERE IDPLATEFORME = idPlateforme;
    IF v_dummy = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Plateforme inexistante');
    END IF;

    SELECT COUNT(*) INTO v_dummy FROM MODEMULTIJOUEUR
    WHERE IDJEU = idJeu AND IDPLATEFORME = idPlateforme;
    IF v_dummy > 0 THEN
        RAISE_APPLICATION_ERROR(-20005, 'Mode Multijoueur d□□ enregistr□');
    END IF;

    IF (ModeCoopOnline = 1 AND NbJoueursMaxCoopOnline = 0) OR
       (ModeCoopOffline = 1 AND NbJoueursMaxCoopOffline = 0) OR
       (NbJoueursMaxCoopOnline > 0 AND ModeCoopOnline = 0) OR
       (NbJoueursMaxCoopOffline > 0 AND ModeCoopOffline = 0) THEN
        RAISE_APPLICATION_ERROR(-20006, 'Donn□es incoh□rentes');
    END IF;
    INSERT INTO MODEMULTIJOUEUR (
        IDMODEMULTIJOUEUR,
        IDJEU,
        IDPLATEFORME,
        DROPIN,
        MODECOOPCAMPAGNE,
        MODECOOPLAN,
        MODECOOPOFFLINE,
        MODECOOPONLINE,
        MODESPLITSCREEN,
        NBJOUEURSMAXCOOPOFFLINE,
        NBJOUEURSMAXOFFLINE,
        NBJOUEURSMAXCOOPONLINE,
        NBJOUEURSMAXONLINE
    ) VALUES (
        (SELECT NVL(MAX(IDMODEMULTIJOUEUR), 0) + 1 FROM MODEMULTIJOUEUR),
        idJeu,
        idPlateforme,
        DropIn,
        ModeCoopCampagne,
        ModeCoopLAN,
        ModeCoopOffline,
        ModeCoopOnline,
        ModeSplitScreen,
        NbJoueursMaxCoopOffline,
        NbJoueursMaxOffline,
        NbJoueursMaxCoopOnline,
        NbJoueursMaxOnline
    );
END;
```

DECLENCHEUR JEU

J’ai utilisé des déclencheurs avec JSON pour tracer précisément les modifications, et j ai uttlise des llm pour avancer plus rapidement dans le projet car il y avait beaucoup de trier a faire

```
CREATE OR REPLACE TRIGGER triggerJeu
AFTER INSERT OR UPDATE OR DELETE ON JEU
FOR EACH ROW
DECLARE
    v_action VARCHAR2(20);
    v_ligneAvant CLOB;
    v_ligneApres CLOB;
BEGIN
    v_action := CASE
        WHEN DELETING THEN 'SUPPRESSION'
        WHEN UPDATING THEN 'MODIFICATION'
        WHEN INSERTING THEN 'AJOUT'
    END;

    IF v_action IN ('MODIFICATION','SUPPRESSION') THEN
        v_ligneAvant := JSON_OBJECT(
            'IDJEU' VALUE :OLD.IDJEU
        RETURNING CLOB);
    ELSE
        v_ligneAvant := NULL;
    END IF;

    IF v_action IN ('AJOUT','MODIFICATION') THEN
        v_ligneApres := JSON_OBJECT(
            'IDJEU' VALUE :NEW.IDJEU
        RETURNING CLOB);
    ELSE
        v_ligneApres := NULL;
    END IF;

    INSERT INTO LOG (
        idAuteur,
        action,
        dateHeureAction,
        tableConcernee,
        idLigneConcernee,
        ligneAvant,
        ligneApres
    )
    VALUES (
        USER,
        v_action,
        SYSTIMESTAMP,
        'JEU',
        COALESCE(TO_CHAR(:NEW.IDJEU), TO_CHAR(:OLD.IDJEU)),
    INSERT INTO LOG (
        idAuteur,
        action,
        dateHeureAction,
        tableConcernee,
        idLigneConcernee,
        ligneAvant,
        ligneApres
    )
    VALUES (
        USER,
        v_action,
        SYSTIMESTAMP,
        'JEU',
        COALESCE(TO_CHAR(:NEW.IDJEU), TO_CHAR(:OLD.IDJEU)),
        v_ligneAvant,
        v_ligneApres
    );
END;
```

extension base de donne

creation de table

Pour enrichir notre modèle de données sans modifier les tables existantes, nous avons créé la table COMPAGNIE_SORTIE afin de lier chaque sortie de jeu à une ou plusieurs compagnies impliquées, en précisant leur rôle spécifique (développeur, éditeur, etc.). Cette extension permet de capturer précisément quelles entreprises ont participé à chaque version ou sortie d'un jeu

```
CREATE TABLE COMPAGNIE_SORTIE (
    IDDATESORTIE NUMBER NOT NULL,
    IDCOMPAGNIE NUMBER NOT NULL,
    ROLE VARCHAR2(20) NOT NULL,
    CONSTRAINT PK_COMPAGNIE_SORTIE PRIMARY KEY (IDDATESORTIE, IDCOMPAGNIE),
    CONSTRAINT FK_CS_DATESORTIE FOREIGN KEY (IDDATESORTIE) REFERENCES DATESORTIE(IDDATESORTIE),
    CONSTRAINT FK_CS_COMPAGNIE FOREIGN KEY (IDCOMPAGNIE) REFERENCES COMPAGNIE(IDCOMPAGNIE),
    CONSTRAINT CHK_ROLE CHECK (ROLE IN ('développeur', 'publieur', 'porteur', 'soutien'))
);
```

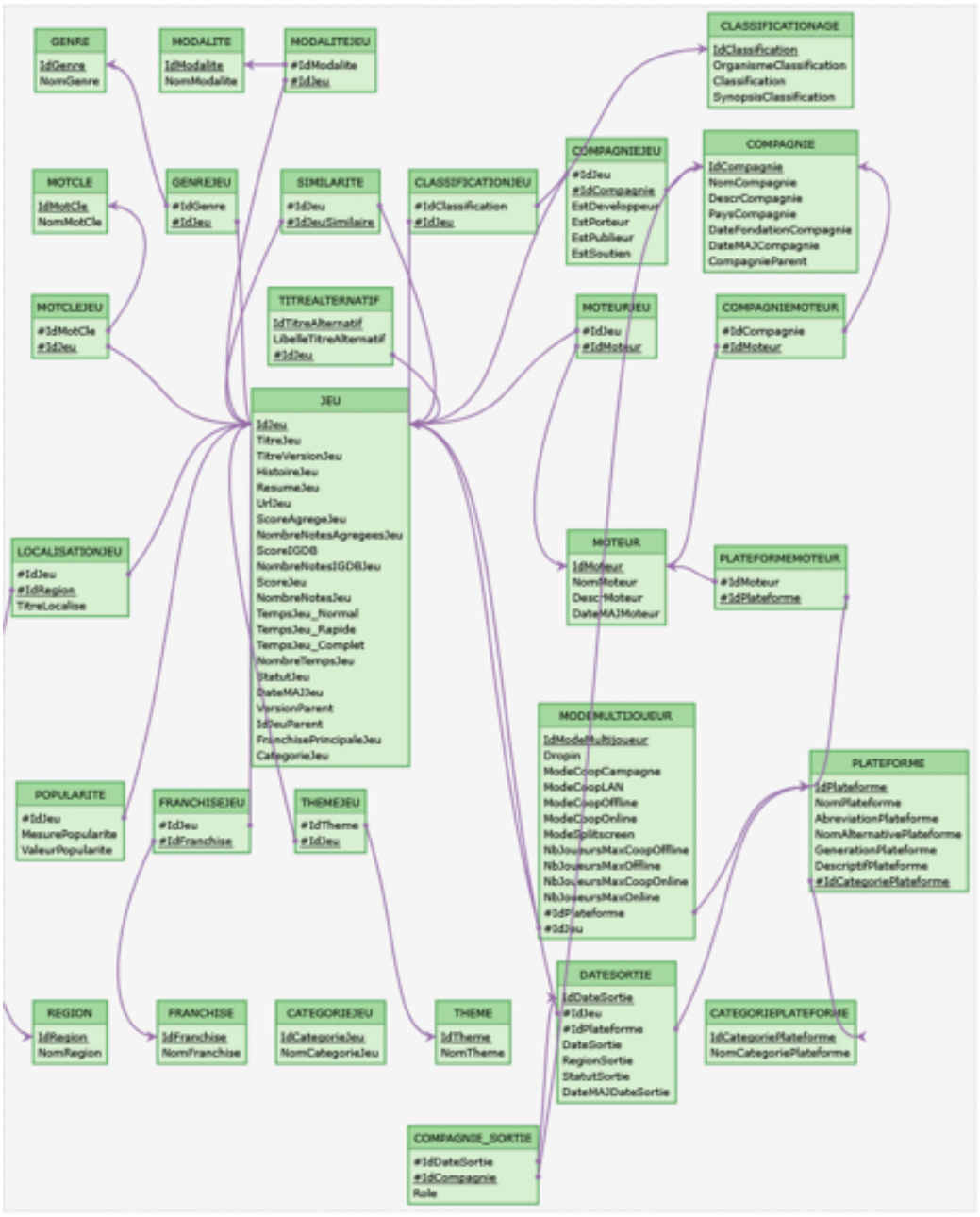
J'ai créé la fonction stockée **DETAIL_SORTIES(idJeu)** qui retourne un objet JSON complet et hiérarchisé listant, pour un jeu donné, les plateformes de sortie (par ordre chronologique), les régions, les dates et statuts de sortie, ainsi que les compagnies impliquées (développeurs, porteurs, publieurs, soutiens), triées comme spécifié.

```
create or replace FUNCTION DETAIL_SORTIES(idJeu IN NUMBER) RETURN CLOB IS
    v_json CLOB;
BEGIN
    SELECT JSON_OBJECT(
        'id' VALUE j.idjeu,
        'titre' VALUE j.titrejeu,
        'sorties' VALUE (
            SELECT JSON_ARRAYAGG(
                JSON_OBJECT(
                    'plateforme' VALUE p.nomplateforme,
                    'regions' VALUE (
                        SELECT JSON_ARRAYAGG(
                            JSON_OBJECT(
                                'region' VALUE ds.regionsortie,
                                'sorties' VALUE (
                                    SELECT JSON_ARRAYAGG(
                                        JSON_OBJECT(
                                            'date' VALUE TO_CHAR(ds2.datesortie, 'YYYY-MM-DD"T"HH24:MI:SS'),
                                            'statut' VALUE NVL(ds2.statutsortie, 'Full Release')
                                        ) ORDER BY ds2.datesortie
                                    )
                                    FROM DATESORTIE ds2
                                    WHERE ds2.idjeu = j.idjeu
                                      AND ds2.idplateforme = p.idplateforme
                                      AND ds2.regionsortie = ds.regionsortie
                                )
                            ) ORDER BY ds.regionsortie
                        )
                        FROM DATESORTIE ds
                        WHERE ds.idjeu = j.idjeu AND ds.idplateforme = p.idplateforme
                        GROUP BY ds.regionsortie
                    ),
                    'developpeurs' VALUE (
                        SELECT JSON_ARRAYAGG(
                            JSON_OBJECT('id' VALUE c.idcompagnie, 'nom' VALUE c.nomcompagnie) ORDER BY c.idcompagnie
                        )
                        FROM COMPAGNIE_SORTIE cs
                        JOIN COMPAGNIE c ON cs.idcompagnie = c.idcompagnie
                        JOIN DATESORTIE ds ON ds.iddatesortie = cs.iddatesortie
                        WHERE ds.idjeu = j.idjeu
                          AND ds.idplateforme = p.idplateforme
                          AND LOWER(cs.role) = 'developpeur'
                    ),
                    'porteurs' VALUE (
                        SELECT JSON_ARRAYAGG(
                            JSON_OBJECT('id' VALUE c.idcompagnie, 'nom' VALUE c.nomcompagnie) ORDER BY c.idcompagnie
                        )
                        FROM COMPAGNIE_SORTIE cs
                        JOIN COMPAGNIE c ON cs.idcompagnie = c.idcompagnie
                        JOIN DATESORTIE ds ON ds.iddatesortie = cs.iddatesortie
                        WHERE ds.idjeu = j.idjeu
                          AND ds.idplateforme = p.idplateforme
                          AND LOWER(cs.role) = 'porteur'
                    ),
                    'publieurs' VALUE (
                        SELECT JSON_ARRAYAGG(
                            JSON_OBJECT('id' VALUE c.idcompagnie, 'nom' VALUE c.nomcompagnie) ORDER BY c.idcompagnie
                        )
                        FROM COMPAGNIE_SORTIE cs
                        JOIN COMPAGNIE c ON cs.idcompagnie = c.idcompagnie
                        JOIN DATESORTIE ds ON ds.iddatesortie = cs.iddatesortie
                        WHERE ds.idjeu = j.idjeu
                          AND ds.idplateforme = p.idplateforme
                          AND LOWER(cs.role) = 'publieur'
                    ),
                    'soutiens' VALUE (
                        SELECT JSON_ARRAYAGG(
                            JSON_OBJECT('id' VALUE c.idcompagnie, 'nom' VALUE c.nomcompagnie) ORDER BY c.idcompagnie
                        )
                        FROM COMPAGNIE_SORTIE cs
                        JOIN COMPAGNIE c ON cs.idcompagnie = c.idcompagnie
                        JOIN DATESORTIE ds ON ds.iddatesortie = cs.iddatesortie
                        WHERE ds.idjeu = j.idjeu
                          AND ds.idplateforme = p.idplateforme
                          AND LOWER(cs.role) = 'soutien'
                    )
                ) ORDER BY MIN(ds.datesortie)
```

```
            FROM DATESORTIE ds
            JOIN PLATEFORME p ON ds.idplateforme = p.idplateforme
            WHERE ds.idjeu = j.idjeu
            GROUP BY p.idplateforme, p.nomplateforme
        )
    ) INTO v_json
    FROM JEU j
    WHERE j.idjeu = idJeu;

    RETURN v_json;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20001, 'Jeu inexistant');
```

le SR MODIFIE AVEC AJOUT TABLE COMPAGNIE_SORTi



insertion de donnee pokemon

```sql
INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (576948, 1617, 'developpeur');
INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (576948, 70, 'publieur');

INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (510115, 1617, 'developpeur');
INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (510115, 70, 'publieur');

INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (576945, 1617, 'developpeur');
INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (576945, 70, 'publieur');

INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (510112, 1617, 'developpeur');
INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (510112, 70, 'publieur');

INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (510117, 1617, 'developpeur');
INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (510117, 70, 'publieur');

INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (514686, 1617, 'developpeur');
INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (514686, 70, 'publieur');

INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (538897, 1617, 'developpeur');
INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (538897, 70, 'publieur');

INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (510114, 1617, 'developpeur');
INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (510114, 70, 'publieur');

INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (510113, 1617, 'developpeur');
INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (510113, 70, 'publieur');

INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (514033, 1617, 'developpeur');
INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (514033, 70, 'publieur');

INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (576946, 1617, 'developpeur');
INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (576946, 70, 'publieur');

INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (576949, 1617, 'developpeur');
INSERT INTO COMPAGNIE_SORTIE (IDDATESORTIE, IDCOMPAGNIE, ROLE) VALUES (576949, 70, 'publieur');
```

insertion de donnee jeux 48455

```sql
INSERT INTO compagnie_sortie (iddatesortie, idcompagnie, role) VALUES (515131, 303, 'developpeur');
INSERT INTO compagnie_sortie (iddatesortie, idcompagnie, role) VALUES (515131, 303, 'publieur');
INSERT INTO compagnie_sortie (iddatesortie, idcompagnie, role) VALUES (515131, 17183, 'publieur');
```

Annexe sr

GENRE: IdGenre, NomGenre
MODALITE: IdModalite, NomModalite
MODALITEJEU: _#IdModalite>MODALITE>IdModalite, _#IdJeu>JEU>IdJeu
:
:
CLASSIFICATIONAGE: IdClassification, OrganismeClassification, Classification, SynopsisClassification

MOTCLE: IdMotCle, NomMotCle
GENREJEU: _#IdGenre>GENRE>IdGenre, _#IdJeu>JEU>IdJeu
SIMILARITE: _#IdJeu>JEU>IdJeu, _#IdJeuSimilaire>JEU>IdJeu
CLASSIFICATIONJEU: _#IdClassification>CLASSIFICATIONAGE>IdClassification, _#IdJeu>JEU>IdJeu
COMPAGNIEJEU: _#IdJeu>JEU>IdJeu, _#IdCompagnie>COMPAGNIE>IdCompagnie, EstDeveloppeur, EstPorteur, EstPublieur, EstSoutien
COMPAGNIE: IdCompagnie, NomCompagnie, DescrCompagnie, PaysCompagnie, DateFondationCompagnie, DateMAJCompagnie, CompagnieParent

MOTCLEJEU: _#IdMotCle>MOTCLE>IdMotCle, _#IdJeu>JEU>IdJeu
:
TITREALTERNATIF: IdTitreAlternatif, LibelleTitreAlternatif, _#IdJeu>JEU>IdJeu
:
MOTEURJEU: _#IdJeu>JEU>IdJeu, _#IdMoteur>MOTEUR>IdMoteur
COMPAGNIEMOTEUR: _#IdCompagnie>COMPAGNIE>IdCompagnie, _#IdMoteur>MOTEUR>IdMoteur

LOCALISATIONJEU: _#IdJeu>JEU>IdJeu, _#IdRegion>REGION>IdRegion, TitreLocalise
:
JEU: IdJeu, TitreJeu, TitreVersionJeu, HistoireJeu, ResumeJeu, UrlJeu, ScoreAgregeJeu, NombreNotesAgregeesJeu, ScoreIGDB, NombreNotesIGDBJeu, ScoreJeu, NombreNotesJeu, TempsJeu_Normal, TempsJeu_Rapide, TempsJeu_Complet, NombreTempsJeu, StatutJeu, DateMAJJeu, VersionParent, IdJeuParent, FranchisePrincipaleJeu, CategorieJeu
:
MOTEUR: IdMoteur, NomMoteur, DescrMoteur, DateMAJMoteur
PLATEFORMEMOTEUR: _#IdMoteur>MOTEUR>IdMoteur, _#IdPlateforme>PLATEFORME>IdPlateforme

POPULARITE: _#IdJeu>JEU>IdJeu, MesurePopularite, ValeurPopularite
FRANCHISEJEU: _#IdJeu>JEU>IdJeu, _#IdFranchise>FRANCHISE>IdFranchise
THEMEJEU: _#IdTheme>THEME>IdTheme, _#IdJeu>JEU>IdJeu
:
MODEMULTIJOUEUR: IdModeMultijoueur, Dropin, ModeCoopCampagne, ModeCoopLAN, ModeCoopOffline, ModeCoopOnline, ModeSplitscreen, NbJoueursMaxCoopOffline, NbJoueursMaxOffline, NbJoueursMaxCoopOnline,

Annexe log

```sql
CREATE TABLE LOG (
    idLog NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    idAuteur VARCHAR2(50) NOT NULL,
    action VARCHAR2(20) NOT NULL, -- 'AJOUT', 'MODIFICATION', 'SUPPRESSION'
    dateHeureAction TIMESTAMP DEFAULT SYSTIMESTAMP NOT NULL,
    tableConcernee VARCHAR2(50) NOT NULL,
    idLigneConcernee VARCHAR2(100),
    ligneAvant CLOB,
    ligneApres CLOB
);
```