TP 1-ENTREPOT DE DONNEES ET BIG DATA

- **Master Informatique**
- **Sujet :** Gestion des photos dans la plateforme flickr
- **Membres** :

  - 21908889: Adam DAIA **PARCOURS GL**
  - 22114603 : Laurencia DOVI LATE **PARCOURS IASD**

2023/2024

Table des matières

Introduction

Ce rapport décrit la modélisation de la base de données de Flickr, une plateforme de partage de photos. L'objectif était de comprendre la structure de la plateforme en se basant sur trois cahiers de charges distincts. Pour ce faire, nous avons créé des modèles UML pour chaque cahier de charge, en prenant en compte les entités, les associations, les cardinalités, les clés, et les contraintes requises.

Le projet a été divisé en trois phases principales : la première portant sur la modélisation des photos, la deuxième sur la gestion des publications, des albums, et des galeries, et la troisième sur les interactions entre les utilisateurs sur Flickr.

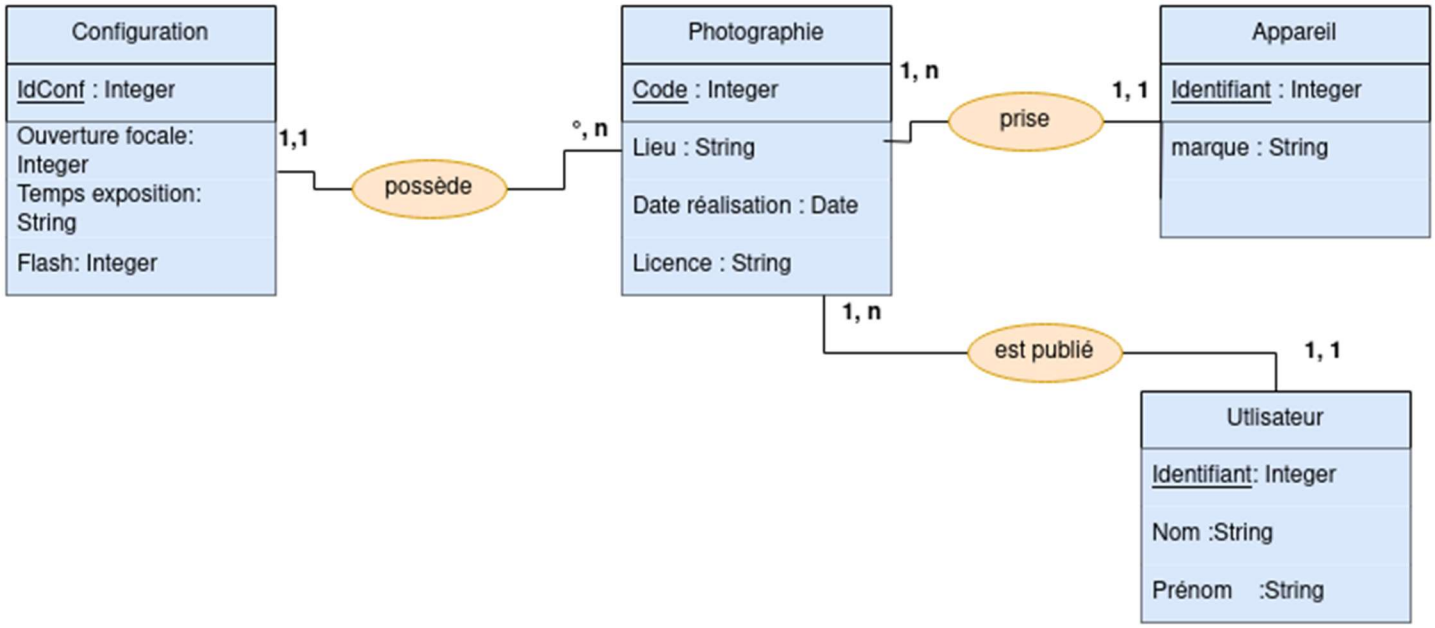
Chaque phase a été réalisée avec précision, en respectant les spécifications des cahiers des charges, et en détaillant les choix de modélisation pour représenter les aspects de la plateforme. Les défis rencontrés lors de la conception des modèles UML ont été relevés et expliqués.

Après la création et la validation des modèles UML, nous avons sélectionné le troisième modèle pour le traduire en un schéma relationnel, et l'implémenter dans le système de gestion de bases de données Oracle.

Ce rapport offre une vue complète du processus de modélisation pour comprendre la structure de la base de données de Flickr.

1 Modélisation conceptuel UML

1-1 Cahier des charges 1



Explication de la modélisation du cahier de charge 1

Le premier cahier des charges décrit la manière dont une photographie est modélisée au sein de la plateforme. Il définit une photographie comme un contenu numérique capturé à l'aide d'un appareil photo spécifique et d'une configuration d'appareil donnée à une date précise et associée à un lieu particulier. De plus, chaque photographie est publiée par un utilisateur de la plateforme à une date spécifique.

Cette modélisation est mise en œuvre comme suit :

- La classe **UTILISATEUR** enregistre les informations sur les utilisateurs de la plateforme, notamment leur identifiant, nom et prénom.

- La classe **APPAREIL** permet de stocker les détails sur les appareils photo utilisés, en incluant l'identifiant et la marque de l'appareil.

- La classe **CONFIGURATION** contient les paramètres de configuration spécifiques à chaque appareil photo, tels que l'ouverture focale, le temps d'exposition et l'état du flash.

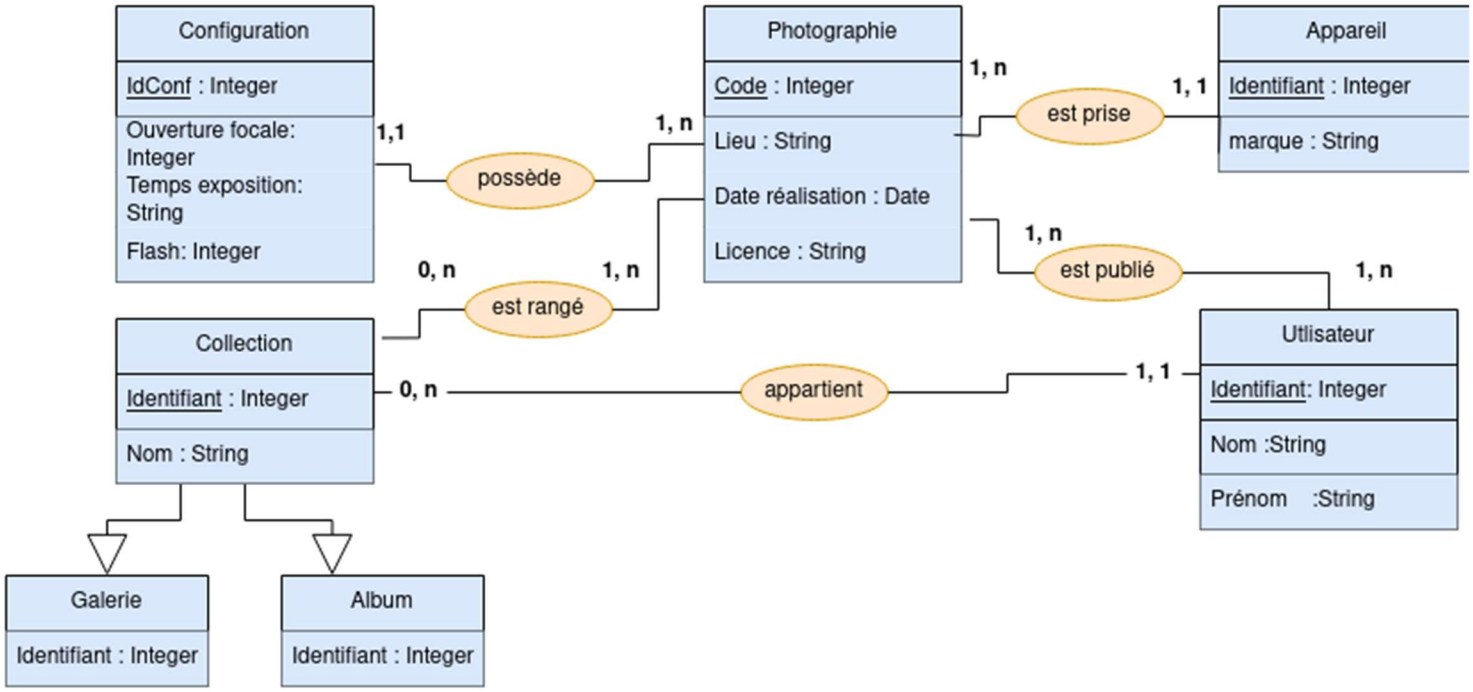
- La classe **CONTENU_NUMERIQUE** fournit une structure de base pour stocker divers contenus numériques, dont les photographies constituent une catégorie.

- La classe **PHOTOGRAPHIE** représente les photographies enregistrées dans le système. Elle enregistre des

informations cruciales telles que le lieu de réalisation, la date de réalisation et la licence associée à chaque photographie. Chaque photographie est liée à l'utilisateur qui l'a publiée à travers l'association **EST PUBLIE ** ,à l'appareil photo qui l'a capturée avec l'association **EST PRISE ** et à la configuration de cet appareil par l'association **POSSEDE**.

Ce premier cahier des charges pose les bases du système de gestion des photographies, définissant la structure et les relations entre les entités clés notamment l'entité central photographie.

1-2 Cahier des charges 2



Explication de la modélisation du cahier de charge 2

Ce second cahier de charges vise à définir la gestion des collections de photos, notamment les albums et les galeries. Ce dernier s'appuie sur le premier cahier de charge et élargit le champ d'action de la plateforme de gestion de photographies en introduisant la notion de collections de photos, notamment les albums et les galeries. Les principales fonctionnalités et spécifications liées à ce cahier de charge sont modélisés de façon suivante :

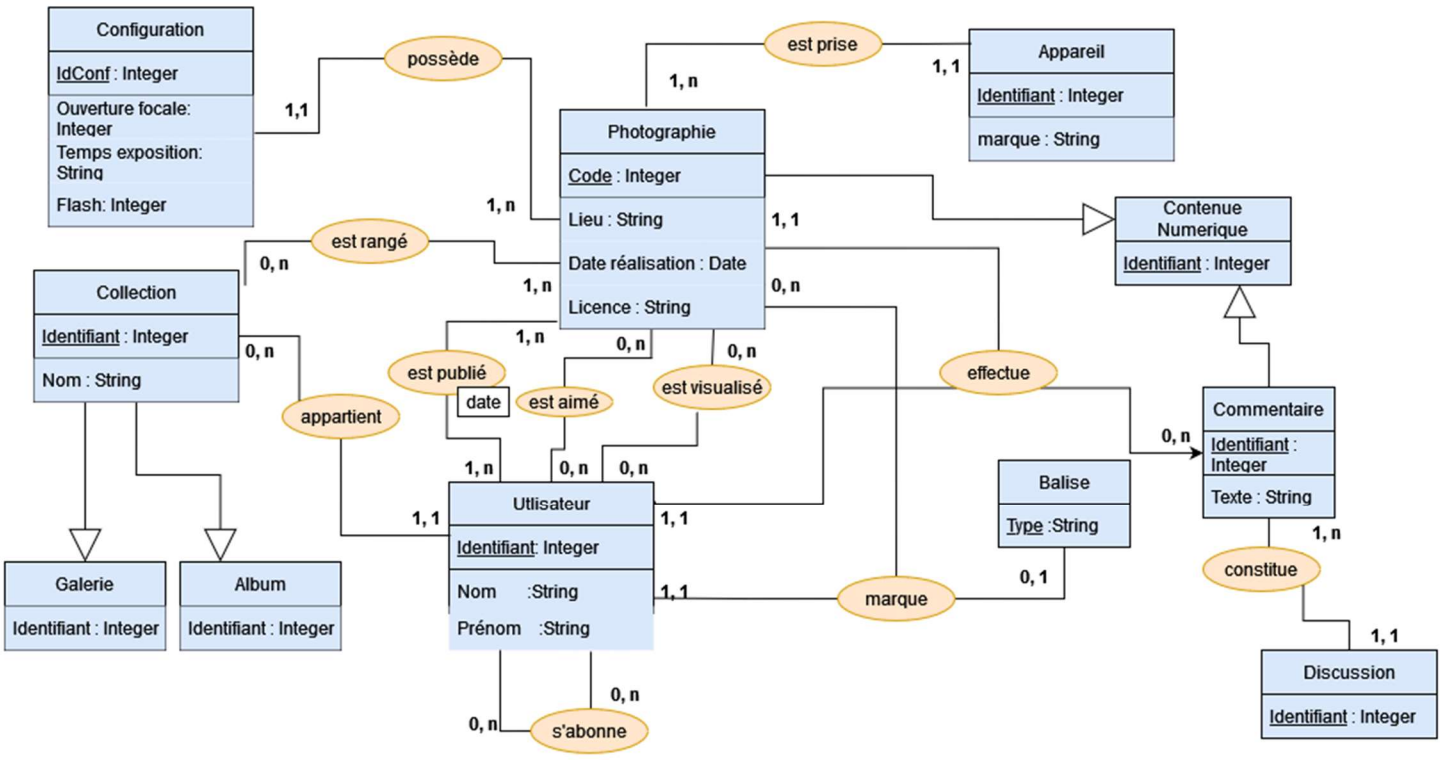
- La classe **COLLECTION** représentant les collections et incluent les albums et les galeries qui seront intégrées dans le système pour permettre aux utilisateurs de regrouper et d'organiser leurs photographies de manière plus structurée. Ainsi cette table a été modélisé pour représenter les collections d'images créées par les utilisateurs. Chaque collection a un nom et est associée à un utilisateur étant propriétaire de l'album.
-Etant donné que chaque collection a un propriétaire, on a la relation **APPARTIENT** qui relie une collection à un utilisateur.
- La classe **ALBUM** : Les albums sont des collections spéciales où les utilisateurs peuvent ranger leurs propres photographies. Etant donné que les albums sont des collections, la table ALBUM hérite de la classe** COLLECTION** et possède un identifiant associé à la table **COLLECTION**.

- La classe **GALERIE** : Les galeries sont des collections plus flexibles qui à la différence des albums permettent aux utilisateurs de rassembler des photographies, y compris celles d'autres utilisateurs. Ainsi cette dernière hérite aussi de la classe **COLLECTION** et possède un identifiant associé à la table **COLLECTION**.

La classe **COLLECTION** est associé à la classe **PHOTOGRAPHIE** par la relation **EST RANGE**.

Pour garantir la contrainte liée à l'album, l'implémentation des tables contiendrait un trigger pour assurer que seuls les propriétaires des photographies peuvent ranger dans la classe **ALBUM.**

1-3 Cahier des charges 3



Explication de la modélisation du cahier de charge 3

Le troisième cahier des charges élargit la plateforme Flickr en introduisant de nouvelles fonctionnalités pour les interactions des utilisateurs et la gestion du contenu. Ce cahier de charges se concentre sur les discussions autour des photographies, les commentaires, les marquages, et bien d'autres éléments.

Cette extension du système est mise en œuvre comme suit :

- L’association **EST AIME ** est introduite pour permettre aux utilisateurs d'exprimer leur appréciation pour une photographie, similaire à la fonction "Like" sur Facebook. Cette association relie un utilisateur à une photographie.

- Les utilisateurs ont la possibilité de s'abonner aux publications d'autres utilisateurs, créant ainsi un système de "follow" à la manière de Twitter. Cette fonctionnalité est mise en œuvre à travers l’association **S’ABONNE**qui relie un utilisateur à un autre utilisateur.

- La table **COMMENTAIRE** est utilisée pour enregistrer les commentaires laissés par les utilisateurs dans les discussions autour des photographies. Chaque commentaire possède un identifiant et un texte et est associé à un utilisateur et à une discussion spécifique.

-Un commentaire est lié à une photographie et à l’utilisateur qui l’a effectué par la relation **EFFECTUE**.

- Les discussions autour des photographies sont structurées à l'aide de la table **DISCUSSION**, qui possède un identifiant et regroupe les commentaires associés à des photographie.

-Un commentaire est lié à une discussion par l’association **CONSTITUE**.

- Les photographies peuvent être marquées par les utilisateurs avec des balises (tags ou mots-clés) pour faciliter leur recherche et leur classification. La classe **BALISE** est utilisée pour gérer ces balises associées aux photographies avec une chaine de caractère représentant le nom de la balise de marquage.

-Une photographie est lié à une balise de marquage et à un utilisateur par l’association **MARQUER**.

- Le nombre de fois qu'une photographie a été visualisée est enregistré et suivi pour fournir des données statistiques aux utilisateurs. De ce fait, on a l’association **EST VISUALISE** qui relie une photographie à un utilisateur.

- Enfin, la classe **CONTENU_NUMERIQUE** qui joue un rôle central dans la gestion de tous les types de contenu au sein de la plateforme, y compris les commentaires et les photographies. Cette classe possède un identifiant et constitue une super classe pour les classes COMMENTAIRE et PHOTOGRAPHIE.

Ce troisième cahier des charges vise à rendre la plateforme Flickr plus interactive et à offrir aux utilisateurs de nouvelles façons d'interagir avec les photographies, de partager leurs opinions et de créer des discussions enrichissantes autour du contenu visuel.

2 Modélisation Logique : schéma relationnel

Suite à la modélisation des spécifications de la plateforme Flickr, nous avons procédé à la transformation de ces spécifications en un modèle relationnel cohérent et structuré. Cette étape de modélisation relationnelle a permis de traduire les entités, les associations et les attributs définis précédemment en tables et en schémas relationnels avec des clés primaires, et secondaires de contraintes.

Ce modèle relationnel servira de base pour la création des tables, des clés et des contraintes nécessaires à la mise en place de la base de données de la plateforme Flickr.

Voici le modèle entité association associé à la modélisation précédente

Entités :

1. **Utilisateur**
   - Attributs : IDUTIL (clé primaire), NOM, PRENOM

2. **Appareil**
   - Attributs : IDAPP (clé primaire), MARQUE

3. **Configuration**
   - Attributs : IDCONF (clé primaire), OUVERTURE_FOCALE, TEMPS_EXPO, FLASH

4. **Contenu Numérique**
   - Attributs : IDCONTENU (clé primaire)

5. **Photographie**
   - Attributs : CODE (clé primaire), LIEU_REALISATION, DATE_REALISATION, LICENCE
   - Clés étrangères : IDUTILISATEUR_PHOTO (référence à Utilisateur qui a publié la photo), IDAPPAREIL (référence à l'identifiant de l'appareil qui a servi à la prise de la photo), IDCONF (référence à l'identifiant de la Configuration utilisée pour la photo, IDCONTENU (référence au Contenu Numérique de la photo)

6. **Balise**
   - Attributs : TYPE_BALISE (clé primaire) représente le texte ou symbole d'une balise

7. **Marquer**
   - Attributs : BALISE_MARQUER (clé étrangère référençant Balise utilisé pour le marquage), CODEPHOTOMARQUER (clé étrangère référençant la Photographie marqué), IDUTIL_MARQUER (clé étrangère référençant Utilisateur ayant marqué la photo)
   - Clé primaire composite : (BALISE_MARQUER, CODEPHOTOMARQUER, IDUTIL_MARQUER)

8. **Commentaire**
   - Attributs : IDCOMMENTAIRE (clé primaire), TEXTE de commentaire
   - Clés étrangères : DIS_COM (référence à Discussion auquel appartient un commentaire s'il appartient à un commentaire), CODEPHOTOCOM (référence à Photographie), IDUTIL_COM (référence à Utilisateur)

9. **Collect**
   - Attributs : IDCOL (clé primaire), NOM (nom de la collection)
   - Clé étrangère : IDPROPRIETAIRE_COLLECT (référence à Utilisateur auquel appartient la collection)

10. **Album**
    - Attributs : IDALB (clé primaire)

11. **Galerie**
    - Attributs : IDGAL (clé primaire)

12. **Ranger**
    - Attributs : IDCOLLECTION (clé étrangère référençant soit album ou galerie), CODEPHOTO (clé étrangère référençant la Photographie à ranger)
    - Clé primaire composite : (IDCOLLECTION, CODEPHOTO)

**Cardinalités **

- ** (Utilisateur - Photographie) ** : Un utilisateur peut publier plusieurs photographies.

- ** (Appareil - Photographie) ** : Un appareil peut être utilisé pour prendre plusieurs photographies.

- **(Configuration - Photographie) ** : Une configuration d'appareil peut être utilisée pour prendre plusieurs photographies.

- ** (Photographie - Balise) ** : Une photographie peut posséder plusieurs balises
.
- ** (Balise - Photographie - Utilisateur) ** : Une balise peut être associée à plusieurs photographies par différents utilisateurs.

- ** (Utilisateur - Photographie - Commentaire) ** : Un utilisateur peut laisser plusieurs commentaires sur différentes photographies.

- **(Utilisateur - Collect) ** : Un utilisateur peut être propriétaire de plusieurs collections.

- ** (Album - Collect) ** : Un album peut être une collection.

- * (Photographie - Galerie) ** : Une photographie peut être placée dans plusieurs galeries.

Ce modèle entité-association détaille toutes les entités, les attributs associés à chacune d'elles, ainsi que les relations entre les entités.

3 Dictionnaire de données

Pour assurer une documentation complète et précise, nous avons élaboré un dictionnaire de données qui décrit en détail chaque table, chaque attribut et chaque contrainte au sein de notre modèle relationnel.

1. ****UTILISATEUR****
   - `IDUTIL` (NUMERIC (4,0)): Identifiant de l'utilisateur (clé primaire).
   - `NOM` (VARCHAR (30)): Nom de l'utilisateur.
   - `PRENOM` (VARCHAR (30)): Prénom de l'utilisateur.

2**. ****APPAREIL****
   - `IDAPP` (NUMERIC (4,0)): Identifiant de l'appareil (clé primaire).
   - `MARQUE` (VARCHAR (30)): Marque de l'appareil.

3. ****CONFIG****
   - `IDCONF` (NUMERIC (4,0)): Identifiant de la configuration (clé primaire).
   - `OUVERTURE_FOCALE` (VARCHAR (30)): Ouverture focale de la configuration.
   - `TEMPS_EXPO` (VARCHAR (30)): Temps d'exposition de la configuration.
   - `FLASH` (NUMERIC (4,0)): Indicateur de flash (0 ou 1).

4**. ****CONTENU_NUMERIQUE****
   - `IDCONTENU` (NUMERIC (4,0)): Identifiant du contenu numérique (clé primaire).

5. **PHOTOGRAPHIE**
- `CODE` (NUMERIC (4,0)): Code de la photographie (clé primaire).
- `LIEU_REALISATION` (VARCHAR (30)): Lieu de réalisation de la photographie.
- `IDUTILISATEUR_PHOTO` (VARCHAR (30)): ID de l'utilisateur ayant pris la photo.
- `DATE_REALISATION` (DATE): Date de réalisation de la photographie.
- `IDAPPAREIL` (NUMERIC (4,0)): ID de l'appareil utilisé pour la photographie.
- `IDCONF` (NUMERIC (4,0)): ID de la configuration de l'appareil pour la photographie.
- `LICENCE` (VARCHAR (50)): Licence de la photographie.

6. **BALISE**
- `TYPE_BALISE` (VARCHAR (40)): Type de balise (clé primaire).

7. **MARQUER**
- `BALISE_MARQUER` (VARCHAR (40)): Balise associée à la photographie (clé étrangère vers BALISE).
- `CODEPHOTOMARQUER` (NUMERIC(4,0)): Code de la photographie associée (clé étrangère vers PHOTOGRAPHIE).
- `IDUTIL_MARQUER` (NUMERIC(4,0)): ID de l'utilisateur associé (clé étrangère vers UTILISATEUR).

8. **COMMENTAIRE**
- `IDCOMMENTAIRE` (NUMERIC(4,0)): Identifiant du commentaire (clé primaire).
- `DIS_COM` (NUMERIC(4,0)): Identifiant de la discussion.
- `TEXTE` (VARCHAR(30)): Texte du commentaire.
- `CODEPHOTOCOM` (NUMERIC(4,0)): Code de la photographie associée (clé étrangère vers PHOTOGRAPHIE).
- `IDUTIL_COM` (NUMERIC(4,0)): ID de l'utilisateur ayant écrit le commentaire (clé étrangère vers UTILISATEUR).

9. **COLLECT**
- `IDCOL` (NUMERIC(4,0)): Identifiant de la collection (clé primaire).
- `NOM` (VARCHAR(30)): Nom de la collection.
- `IDPROPRIETAIRE_COLLECT` (VARCHAR(30)): ID de l'utilisateur propriétaire de la collection (clé étrangère vers UTILISATEUR).

10. **ALBUM**
- `IDALB` (NUMERIC(4,0)): Identifiant de l'album (clé primaire).

11. **GALERIE**
- `IDGAL` (NUMERIC(4,0)): Identifiant de la galerie (clé primaire).

12. **RANGER**
- `IDCOLLECTION` (NUMERIC(4,0)): Identifiant de la collection (clé étrangère vers COLLECT).
- `CODEPHOTO` (NUMERIC(4,0)): Code de la photographie associée (clé étrangère vers PHOTOGRAPHIE).

4 Modélisation physique :Schéma Physique

Après avoir défini le modèle conceptuel de données et transformé nos spécifications en un modèle relationnel cohérent, nous avons procédé à la conception du schéma physique de notre base de données en sql. Ci-dessous l'implémentation du troisième modèle dans l'SGBD ORACLE.

4-1Création des tables

```
CREATE TABLE UTILISATEUR (
IDUTIL NUMERIC (4,0),
NOM VARCHAR(30) CONSTRAINT NN_NOMUTIL NOT NULL,
PRENOM VARCHAR(30) CONSTRAINT NN_PRENOMUTIL NOT NULL,
   CONSTRAINT PK_UTILISATEUR PRIMARY KEY(IDUTIL)
);


CREATE TABLE APPAREIL (
IDAPP NUMERIC (4,0),
MARQUE VARCHAR(30) CONSTRAINT NN_MARQUEAPP NOT NULL,
   CONSTRAINT PK_APPAREIL PRIMARY KEY(IDAPP)
);


CREATE TABLE CONFIG (
IDCONF NUMERIC (4,0),
OUVERTURE_FOCALE VARCHAR(30) CONSTRAINT NN_OUVERTURE NOT NULL,
TEMPS_EXPO VARCHAR(30) CONSTRAINT NN_EXPO   NOT NULL ,
FLASH NUMERIC(4,0) CONSTRAINT CK_FLASH CHECK (FLASH =0 OR FLASH = 1),
   CONSTRAINT PK_CONFIGURATION PRIMARY KEY(IDCONF)
);


CREATE TABLE CONTENU_NUMERIQUE(
IDCONTENU NUMERIC (4,0),
   CONSTRAINT PK_CONTENU_NUM PRIMARY KEY(IDCONTENU)
);


CREATE TABLE PHOTOGRAPHIE (
CODE NUMERIC (4,0),
LIEU_REALISATION VARCHAR(30) CONSTRAINT NN_LIEU NOT NULL,
DATE_REALISATION DATE CONSTRAINT NN_DATE NOT NULL,
IDAPPAREIL NUMERIC (4,0) CONSTRAINT NN_APPAREIL_PHOTO NOT NULL,
IDCONF NUMERIC (4,0) CONSTRAINT NN_CONFIG_PHOTO NOT NULL,
LICENCE VARCHAR(50) CONSTRAINT CK_LICENCE CHECK(LICENCE IN('tous droits reserves','utilisation commerciale autorisee','modifications de image autorisees')),
   CONSTRAINT PK_PHOTO PRIMARY KEY(CODE),
   CONSTRAINT FK_CONTENU_NUM FOREIGN KEY(CODE) REFERENCES CONTENU_NUMERIQUE(IDCONTENU) ON DELETE CASCADE,
   CONSTRAINT FK_APPAREIL FOREIGN KEY(IDAPPAREIL) REFERENCES APPAREIL(IDAPP) ON DELETE CASCADE,
   CONSTRAINT FK_CONFIGURATION FOREIGN KEY(IDCONF) REFERENCES CONFIG(IDCONF) ON
```

```
DELETE CASCADE
);

CREATE TABLE COLLECT (
IDCOL NUMERIC (4,0),
NOM VARCHAR(30) CONSTRAINT NN_NOMCOLLECT NOT NULL,
IDPROPRIETAIRE_COLLECT NUMERIC (4,0) CONSTRAINT NN_PROPRIETAIRECOLLECT NOT NULL,
   CONSTRAINT PK_COLLECT PRIMARY KEY(IDCOL),
CONSTRAINTFK_PROPRIETAIRE_COLLECT FOREIGN KEY(IDPROPRIETAIRE_COLLECT)REFERENCES UTILISATEUR(IDUTIL) ON DELETE CASCADE
);

CREATE TABLE ALBUM (
IDALB NUMERIC (4,0),
   CONSTRAINT PK_ALBUM PRIMARY KEY(IDALB),
   CONSTRAINT FK_ALBUM FOREIGN KEY(IDALB)REFERENCES COLLECT(IDCOL) ON DELETE CASCADE
);

CREATE TABLE GALERIE (
IDGAL NUMERIC (4,0),
   CONSTRAINT PK_GALERIE PRIMARY KEY(IDGAL),
   CONSTRAINT FK_GALERIE FOREIGN KEY(IDGAL)REFERENCES COLLECT(IDCOL) ON DELETE CASCADE
);

CREATE TABLE RANGER (
IDCOLLECTION NUMERIC (4,0),
CODEPHOTO NUMERIC (4,0),
   CONSTRAINT PK_RANGER PRIMARY KEY(IDCOLLECTION,CODEPHOTO),
   CONSTRAINT FK_COLLECT FOREIGN KEY(IDCOLLECTION)REFERENCES COLLECT(IDCOL) ON DELETE CASCADE,
   CONSTRAINT FK_PHOTOGRAPHIE FOREIGN KEY(CODEPHOTO)REFERENCES PHOTOGRAPHIE(CODE) ON DELETE CASCADE
);

CREATE TABLE DISCUSSION (
IDDIS NUMERIC (4,0),
   CONSTRAINT PK_DISCUSSION PRIMARY KEY(IDDIS)
);

CREATE TABLE AIME (
CODEPHOTOLIKE NUMERIC (4,0),
ID_UTIL_LIKE NUMERIC (4,0) CONSTRAINT NN_IDUTILISATEURLIKE NOT NULL,
   CONSTRAINT PK_LIKE PRIMARY KEY(CODEPHOTOLIKE,ID_UTIL_LIKE),
   CONSTRAINT FK_LIKE_PHOTO FOREIGN KEY(CODEPHOTOLIKE)REFERENCES PHOTOGRAPHIE(CODE) ON DELETE CASCADE,
   CONSTRAINT FK_LIKE_UTILISATEUR FOREIGN KEY(ID_UTIL_LIKE)REFERENCES UTILISATEUR(IDUTIL) ON DELETE CASCADE
);
```

```
CREATE TABLE ABONNE (
ID_UTIL1 NUMERIC (4,0) CONSTRAINT NN_IDUTILISATEUR_ABON_1 NOT NULL,
ID_UTIL2 NUMERIC (4,0) CONSTRAINT NN_IDUTILISATEUR_ABON_2 NOT NULL,
   CONSTRAINT PK_ABONNE PRIMARY KEY(ID_UTIL1,ID_UTIL2),
   CONSTRAINT FK_ABONNE_UTILISATEUR1 FOREIGN KEY(ID_UTIL1)REFERENCES
UTILISATEUR(IDUTIL) ON DELETE CASCADE,
   CONSTRAINT FK_ABONNE_UTILISATEUR2 FOREIGN KEY(ID_UTIL2)REFERENCES
    UTILISATEUR(IDUTIL) ON DELETE CASCADE
);


CREATE TABLE VISUALISE (
CODEPHOTOVIS NUMERIC (4,0) CONSTRAINT NN_CODEPHOTOVIS NOT NULL,
IDUTIL_VIS NUMERIC (4,0) CONSTRAINT NN_IDUTILISATEURVIS NOT NULL,
   CONSTRAINT PK_VISUALISE PRIMARY KEY(CODEPHOTOVIS,IDUTIL_VIS),
   CONSTRAINT FK_VISUALISE_UTILISATEUR FOREIGN KEY(IDUTIL_VIS)REFERENCES
UTILISATEUR(IDUTIL) ON DELETE CASCADE,
   CONSTRAINT FK_VISUALISE_PHOTO FOREIGN KEY(CODEPHOTOVIS)REFERENCES
PHOTOGRAPHIE(CODE) ON DELETE CASCADE
);


CREATE TABLE PUBLIE  (
CODEPHOTOPUB NUMERIC (4,0) CONSTRAINT NN_CODEPHOTOPUB NOT NULL,
DATEPUB DATE  CONSTRAINT NN_DATEPUB NOT NULL,
IDUTIL_PUB NUMERIC (4,0) CONSTRAINT NN_IDUTILISATEURPUB NOT NULL,
   CONSTRAINT PK_PUBLIE PRIMARY KEY(CODEPHOTOPUB,IDUTIL_PUB),
   CONSTRAINT FK_PUB_UTILISATEUR FOREIGN KEY(IDUTIL_PUB)REFERENCES
UTILISATEUR(IDUTIL) ON DELETE CASCADE,
   CONSTRAINT FK_PUB_PHOTO FOREIGN KEY(CODEPHOTOPUB)REFERENCES
PHOTOGRAPHIE(CODE) ON DELETE CASCADE
);


CREATE TABLE COMMENTAIRE (
IDCOMMENTAIRE NUMERIC (4,0),
DIS_COM NUMERIC (4,0),
TEXTE VARCHAR(50) CONSTRAINT NN_TEXTECOMMENTAIRE NOT NULL,
CODEPHOTOCOM NUMERIC (4,0) CONSTRAINT NN_CODEPHOTOCOM NOT NULL,
IDUTIL_COM NUMERIC (4,0) CONSTRAINT NN_IDUTILISATEURCOM NOT NULL,
   CONSTRAINT PK_COMMENTAIRE PRIMARY KEY(IDCOMMENTAIRE),
   CONSTRAINT FK_CONTENU_NUM_COM FOREIGN KEY(IDCOMMENTAIRE)
   REFERENCES CONTENU_NUMERIQUE(IDCONTENU) ON DELETE CASCADE,
   CONSTRAINT FK_DISCUSSION_COM FOREIGN KEY(DIS_COM)REFERENCES DISCUSSION(IDDIS)
ON DELETE CASCADE,
   CONSTRAINT FK_COMMENTAIRE_PHOTO FOREIGN KEY(CODEPHOTOCOM)REFERENCES
PHOTOGRAPHIE(CODE) ON DELETE CASCADE,
   CONSTRAINT FK_COMMENTAIRE_UTILISATEUR FOREIGN KEY(IDUTIL_COM)REFERENCES
UTILISATEUR(IDUTIL) ON DELETE CASCADE
);


CREATE TABLE BALISE (
TYPE_BALISE VARCHAR(40),
```

```
   CONSTRAINT PK_BALISE PRIMARY KEY(TYPE_BALISE)
);


CREATE TABLE MARQUER (
BALISE_MARQUER VARCHAR(40),
CODEPHOTOMARQUER NUMERIC (4,0) CONSTRAINT NN_CODEPHOTOMAR NOT NULL,
IDUTIL_MARQUER NUMERIC (4,0) CONSTRAINT NN_IDUTILISATEURMAR NOT NULL,
   CONSTRAINT PK_MARQUER PRIMARY KEY(BALISE_MARQUER,CODEPHOTOMARQUER,IDUTIL_MARQUER),
   CONSTRAINT FK_MAR_UTILISATEUR FOREIGN KEY(IDUTIL_MARQUER)REFERENCES UTILISATEUR(IDUTIL) ON DELETE CASCADE,
   CONSTRAINT FK_MAR_PHOTO FOREIGN KEY(CODEPHOTOMARQUER)REFERENCES PHOTOGRAPHIE(CODE) ON DELETE CASCADE,
   CONSTRAINT FK_MAR_BALISE FOREIGN KEY(BALISE_MARQUER)REFERENCES BALISE(TYPE_BALISE) ON DELETE CASCADE
);
```

4-2 Triggers

Trigger check_album

Ce trigger nommé "check_album", est conçu pour être activé avant qu'une insertion ne soit effectuée dans la table "RANGER" afin de garantir que seul le propriétaire d'un album est autorisé à ranger des photos dans cet album spécifique.
Voici une explication détaillée du code :
BEFORE INSERT ON RANGER: Cela signifie que le déclencheur sera exécuté avant qu'une nouvelle ligne ne soit insérée dans la table "RANGER".
FOR EACH ROW: Cette clause indique que le déclencheur s'exécute pour insertion ou modification de chaque ligne dans la table "RANGER".
DECLARE: Déclaration de variables locales: "is_album" pour stocker le résultat de la vérification de l'existence de l'album, et "album_owner_id" pour stocker l'identifiant du propriétaire de l'album.
BEGIN:
Le déclencheur commence par vérifier si l'ID de collection (NEW.IDCOLLECTION) correspond à un album existant. Il effectue cette vérification en comptant le nombre d'occurrences de l'ID de l'album dans la table "ALBUM". Si "is_album" est égal à 1, cela signifie qu'un album correspondant existe.
Vérification du propriétaire de l'album: Si l'ID correspond à un album, le déclencheur vérifie ensuite si l'utilisateur (NEW.IDUTILISATEUR) qui souhaite ranger une photo dans cet album est également le propriétaire de l'album. Pour ce faire, il sélectionne l'ID du propriétaire de l'album à partir de la table "COLLECT" en utilisant l'ID de la collection (NEW.IDCOLLECTION). Si l'utilisateur n'est pas le propriétaire de l'album (c'est-à-dire si :NEW.IDUTILISATEUR <> album_owner_id), le déclencheur génère une erreur personnalisée à l'aide de "RAISE_APPLICATION_ERROR" informant que seuls les propriétaires d'albums sont autorisés à ranger des photos dans cet album.

4-3 Insertion des tuples

```
INSERT INTO UTILISATEUR VALUES (0001 , 'DURAND' , 'Lucas' ) ;
INSERT INTO UTILISATEUR VALUES (0002 , 'TURING' , 'Alan' ) ;
INSERT INTO UTILISATEUR VALUES (0003 , 'MARIZ' , 'Julie' ) ;
INSERT INTO UTILISATEUR VALUES (0004 , 'MARIA' , 'Rose' ) ;
INSERT INTO UTILISATEUR VALUES (0005 , 'TOTO' , 'PAUL' ) ;


INSERT INTO APPAREIL VALUES (1000 , 'HP' ) ;
INSERT INTO APPAREIL VALUES (1001 , 'APPLE' ) ;
INSERT INTO APPAREIL VALUES (1002 , 'KANON' ) ;
INSERT INTO APPAREIL VALUES (1003 , 'NOKIA' ) ;
INSERT INTO APPAREIL VALUES (1004 , 'ASUS' ) ;


INSERT INTO CONFIG VALUES (2000 , '1234mm' , '0.1s' , 0) ;
INSERT INTO CONFIG VALUES (2001 , '9876mm' , '2.0s' , 0) ;
INSERT INTO CONFIG VALUES (2002 , '6547mm' , '0.1s' , 1) ;
INSERT INTO CONFIG VALUES (2003 , '0324mm' , '2.5s' , 1) ;
INSERT INTO CONFIG VALUES (2004 , '4710mm' , '1.6s' , 1) ;


INSERT INTO CONTENU_NUMERIQUE VALUES (3000  ) ;
INSERT INTO CONTENU_NUMERIQUE VALUES (3001  ) ;
INSERT INTO CONTENU_NUMERIQUE VALUES (3002  ) ;
INSERT INTO CONTENU_NUMERIQUE VALUES (3003  ) ;
INSERT INTO CONTENU_NUMERIQUE VALUES (3004  ) ;
INSERT INTO CONTENU_NUMERIQUE VALUES (3005  ) ;

INSERT INTO CONTENU_NUMERIQUE VALUES (3100  ) ;
INSERT INTO CONTENU_NUMERIQUE VALUES (3101  ) ;
INSERT INTO CONTENU_NUMERIQUE VALUES (3102 ) ;
INSERT INTO CONTENU_NUMERIQUE VALUES (3103 ) ;
INSERT INTO CONTENU_NUMERIQUE VALUES (3104  ) ;
INSERT INTO CONTENU_NUMERIQUE VALUES (3105  ) ;


INSERT INTO PHOTOGRAPHIE VALUES (3000 , 'Montpellier' , '19/09/2022' ,1000, 2000 ,'tous droits reserves') ;
INSERT INTO PHOTOGRAPHIE VALUES (3001 , 'Toulouse' , '29/10/2023' ,1000, 2000 ,'modifications de image autorisees') ;
INSERT INTO PHOTOGRAPHIE VALUES (3002 , 'Marseille' ,'12/01/2023' ,1000, 2001 ,'utilisation commerciale autorisee') ;
INSERT INTO PHOTOGRAPHIE VALUES (3003 , 'Montpellier' ,'14/02/2023' ,1002, 2004 ,'tous droits reserves') ;
INSERT INTO PHOTOGRAPHIE VALUES (3004 , 'Montpellier' , '14/02/2022' ,1003, 2001 ,'tous droits reserves') ;
INSERT INTO PHOTOGRAPHIE VALUES (3005 , 'Montpellier' ,'27/07/2023' ,1002, 2003 ,'tous droits reserves') ;


INSERT INTO COLLECT VALUES (4000 , 'Animaux', 0001) ;
INSERT INTO COLLECT VALUES (4001 , 'Voitures',0001 ) ;
INSERT INTO COLLECT VALUES (4002 , 'Nature',0002 ) ;
INSERT INTO COLLECT VALUES (4003 , 'Mode',0003 ) ;
INSERT INTO COLLECT VALUES (4004 , 'Sport' ,0004) ;


INSERT INTO ALBUM VALUES (4000  ) ;
INSERT INTO ALBUM VALUES (4001  ) ;
INSERT INTO ALBUM VALUES (4002  ) ;
```

```
INSERT INTO ALBUM VALUES (4004  ) ;


INSERT INTO GALERIE VALUES (4000  ) ;
INSERT INTO GALERIE VALUES (4001  ) ;
INSERT INTO GALERIE VALUES (4002  ) ;
INSERT INTO GALERIE VALUES (4003  ) ;


INSERT INTO RANGER VALUES (4000, 3000  ) ;
INSERT INTO RANGER VALUES (4001, 3000  ) ;
INSERT INTO RANGER VALUES (4002, 3000  ) ;

INSERT INTO RANGER VALUES (4000, 3003  ) ;
INSERT INTO RANGER VALUES (4002, 3001  ) ;
INSERT INTO RANGER VALUES (4001, 3002  ) ;
INSERT INTO RANGER VALUES (4004, 3004  ) ;


INSERT INTO DISCUSSION VALUES (5000  ) ;
INSERT INTO DISCUSSION VALUES (5001  ) ;
INSERT INTO DISCUSSION VALUES (5002  ) ;
INSERT INTO DISCUSSION VALUES (5003  ) ;
INSERT INTO DISCUSSION VALUES (5004  ) ;
INSERT INTO DISCUSSION VALUES (5005 ) ;


INSERT INTO AIME VALUES (3000,0001  ) ;
INSERT INTO AIME VALUES (3000,0003  ) ;
INSERT INTO AIME VALUES (3000,0005  ) ;

INSERT INTO AIME VALUES (3002,0002  ) ;
INSERT INTO AIME VALUES (3002,0004  ) ;

INSERT INTO AIME VALUES (3004,0004  ) ;

INSERT INTO AIME VALUES (3005,0001  ) ;
INSERT INTO AIME VALUES (3005,0002  ) ;
INSERT INTO AIME VALUES (3005,0003  ) ;
INSERT INTO AIME VALUES (3005,0004  ) ;


INSERT INTO ABONNE VALUES (0001,0002  ) ;
INSERT INTO ABONNE VALUES (0001,0003  ) ;
INSERT INTO ABONNE VALUES (0001,0004  ) ;
INSERT INTO ABONNE VALUES (0001,0005  ) ;

INSERT INTO ABONNE VALUES (0002,0004  ) ;

INSERT INTO ABONNE VALUES (0004,0005  ) ;

INSERT INTO ABONNE VALUES (0003,0001  ) ;


INSERT INTO VISUALISE VALUES (3000,0001  ) ;
INSERT INTO VISUALISE VALUES (3000,0002  ) ;
```

```
INSERT INTO VISUALISE VALUES (3000,0003  ) ;
INSERT INTO VISUALISE VALUES (3000,0004  ) ;

INSERT INTO VISUALISE VALUES (3001,0001  ) ;
INSERT INTO VISUALISE VALUES (3001,0003  ) ;
INSERT INTO VISUALISE VALUES (3001,0004  ) ;

INSERT INTO VISUALISE VALUES (3002,0001  ) ;

INSERT INTO VISUALISE VALUES (3004,0002  ) ;
INSERT INTO VISUALISE VALUES (3004,0004  ) ;


INSERT INTO PUBLIE VALUES (3000,'11/02/2022',0001  ) ;
INSERT INTO PUBLIE VALUES (3001,'21/12/2020',0001  ) ;

INSERT INTO PUBLIE VALUES (3002,'07/07/2023',0002  ) ;

INSERT INTO PUBLIE VALUES (3003,'25/07/2022',0003  ) ;

INSERT INTO PUBLIE VALUES (3004,'30/11/2021',0004  ) ;

INSERT INTO PUBLIE VALUES (3005,'26/06/2022',0005  ) ;

INSERT INTO COMMENTAIRE VALUES (3100,5000,'jolie photo', 3000,0004  ) ;
INSERT INTO COMMENTAIRE VALUES (3105,5005,'sublime', 3000,0002  ) ;
INSERT INTO COMMENTAIRE VALUES (3101,5001,'une merveille la ville rose', 3001,0002  ) ;
INSERT INTO COMMENTAIRE VALUES (3102,5002,'vous auriez du utiliser le flash', 3002,0003  ) ;
INSERT INTO COMMENTAIRE VALUES (3103,5003,'très belle plage !!!!', 3003,0001  ) ;
INSERT INTO COMMENTAIRE VALUES (3104,5004,'quelle prise ', 3004,0004  ) ;


INSERT INTO BALISE VALUES ('plage' ) ;
INSERT INTO BALISE VALUES ('Toulouse' ) ;
INSERT INTO BALISE VALUES ('tigre' ) ;


INSERT INTO MARQUER VALUES ('plage',3003,0001 ) ;
INSERT INTO MARQUER VALUES ('Toulouse',3001,0001 ) ;
INSERT INTO MARQUER VALUES ('tigre',3004,0004)
```

5 Les requêtes de recherche et leurs explications

Ces requêtes ci-dessous représentent trois différentes recherches sur la plateforme Flick.

Requête 1 : Photos récentes marque avec la balise 'plage'

```
SELECT P.CODE, P.LIEU_REALISATION, P.DATE_REALISATION
  FROM PHOTOGRAPHIE P
```

```
INNER JOIN MARQUER M ON P.CODE = M.CODEPHOTOMARQUER
INNER JOIN BALISE B ON M.BALISE_MARQUER = B.TYPE_BALISE
WHERE B.TYPE_BALISE = 'plage'
ORDER BY P.DATE_REALISATION DESC;
```

Cette requête sélectionne les informations (code, lieu de réalisation et date de réalisation) des photos qui ont la balise 'plage'. Les résultats sont triés par ordre décroissant de la date de réalisation.

Résultat

```
Donner la/les photos avec la balise 'plage' trié par ordre décroissant

      CODE LIEU_REALISATION               DATE_REALI
---------- ------------------------------ ----------
      3003 Montpellier                    14/02/2023
```

Requête 2 : Photos prises avec le flash à Montpellier en 2023

```
SELECT P.CODE, P.LIEU_REALISATION, P.DATE_REALISATION
FROM PHOTOGRAPHIE P
INNER JOIN CONFIG C ON P.IDCONF = C.IDCONF
WHERE P.LIEU_REALISATION = 'Montpellier'
AND P.DATE_REALISATION >= '01/01/2023' AND P.DATE_REALISATION <= '31/12/2023'
AND C.FLASH = 1;
```

Cette requête sélectionne les photos prises avec le flash à Montpellier en 2023. Elle vérifie si le lieu de réalisation est 'Montpellier' et si la date de réalisation est comprise entre le 1er janvier 2023 et le 31 décembre 2023, en plus de vérifier la configuration avec le flash activé.

Résultat

```
Donner la/les photos prises avec le flash à Montpellier en 2023

      CODE LIEU_REALISATION               DATE_REALI
---------- ------------------------------ ----------
      3003 Montpellier                    14/02/2023
      3005 Montpellier                    27/07/2023
```

Requête 3 : Photos prise avec la même configuration et le même appareil que la photo n°3000

```
SELECT P.CODE, P.LIEU_REALISATION, P.DATE_REALISATION
FROM PHOTOGRAPHIE P
WHERE P.CODE <> 3000
AND P.IDAPPAREIL = (SELECT IDAPPAREIL FROM PHOTOGRAPHIE WHERE CODE = 3000)
AND P.IDCONF = (SELECT IDCONF FROM PHOTOGRAPHIE WHERE CODE = 3000);
```

Cette requête sélectionne les photos ayant la même configuration (IDCONF) et le même appareil (IDAPPAREIL) que la photo avec le code 3000, mais exclut la photo avec le code 3000 elle-même.

Résultat

```
Donner la/les photos prise avec la même configuration et le même appareil que la photo n°3000

      CODE LIEU_REALISATION              DATE_REALI
---------- ------------------------------ ----------
      3001 Toulouse                       29/10/2023
```

6 Les requêtes du sujet

Donner la requête qui permet de trouver les photos les plus appréciées avec la licence de distribution 'tous droits réservés'.

```
SELECT P.CODE, P.LIEU_REALISATION, P.DATE_REALISATION, P.LICENCE, COUNT(A.ID_UTIL_LIKE) AS
LIKE_COUNT
   FROM PHOTOGRAPHIE P
   INNER JOIN AIME A ON P.CODE = A.CODEPHOTOLIKE
   WHERE P.LICENCE = 'tous droits reserves'
   GROUP BY P.CODE, P.LIEU_REALISATION, P.DATE_REALISATION, P.LICENCE
   HAVING COUNT(A.ID_UTIL_LIKE) = (
      SELECT MAX(LIKE_COUNT)
         FROM (
         SELECT P2.CODE, COUNT(A2.ID_UTIL_LIKE) AS LIKE_COUNT
         FROM PHOTOGRAPHIE P2
         INNER JOIN AIME A2 ON P2.CODE = A2.CODEPHOTOLIKE
         WHERE P2.LICENCE = 'tous droits reserves'
         GROUP BY P2.CODE, P2.LICENCE
         )
      );
```

Cette requête trouve les photos avec la licence 'tous droits réservés' qui ont reçu le plus grand nombre de likes. Elle utilise une sous-requête pour déterminer le nombre maximum de likes et ensuite sélectionne les photos qui ont ce nombre de likes.

Résultat

```
Donner la requête qui permet de trouver les photos les plus appréciées avec la licence de distribution
SP2-0734: commande inconnue au debut de "'tous dr..." - le reste de la ligne est ignore.

      CODE LIEU_REALISATION              DATE_REALI LICENCE
---------- ------------------------------ ---------- --------------------
LIKE_COUNT
----------
      3005 Montpellier                    27/07/2023 tous droits reserves
         4
```

Donner la requête qui permet de trouver les photos incluses dans le plus grand nombre de galeries.

```
SELECT R.CODEPHOTO, P.LIEU_REALISATION, P.DATE_REALISATION, COUNT(*) AS GALLERY_COUNT
   FROM RANGER R
   INNER JOIN GALERIE G ON R.IDCOLLECTION = G.IDGAL
   INNER JOIN PHOTOGRAPHIE P ON R.CODEPHOTO = P.CODE
   GROUP BY R.CODEPHOTO, P.LIEU_REALISATION, P.DATE_REALISATION
   ORDER BY GALLERY_COUNT DESC;
```

Cette requête identifie les photos qui sont incluses dans le plus grand nombre de galeries en comptant le nombre de fois qu'une photo apparaît dans la table de liaison RANGER.

Résultat

```
Donner la requête qui permet de trouver les photos incluses dans le plus grande nombre de galeries.

 CODEPHOTO LIEU_REALISATION              DATE_REALI GALLERY_COUNT
---------- ------------------------------ ---------- -------------
      3000 Montpellier                    19/09/2022             3
      3002 Marseille                      12/01/2023             1
      3003 Montpellier                    14/02/2023             1
      3001 Toulouse                       29/10/2023             1
```

7 ReadMe

-Exécution du fichier suppression.sql
-Exécution du fichier creation.sql
-Exécution du fichier requete.sql

Conclusion

Dans le cadre de notre TP, nous avons expliqué le schéma de modélisation de la base de données de la plateforme de partage de photos. De plus, nous avons spécifié un trigger et des contraintes. En effet, ce tp a fait l'objet d'une expérience intéressante, il nous a permis de réaliser que le projet de modélisation d'une base de données est un ensemble de plusieurs actions planifiées et dépendantes les unes des autres. Toutes les étapes de ce tp nous ont permis d'enrichir notre expérience de modélisation de base de données en utilisant les trois modèles mentionnés au cours notamment le modèle conceptuel, logique et physique.