Jallais Bastien	Gesnel Kerrian	Fumeron-Lecomte Baptiste	Ratovobodo Nicka

Rapport Sujet BD

Question 1)

a. Dictionnaire des données utilisées dans Touiteur

Table envisagée	nom champ	type	longueur max / format	définition du champ
Utilisateur	emailUtil	Texte	25	email d'un utilisateur
Utilisateur	nomUtil	Texte	15	nom d'un utilisateur
Utilisateur	prenomUtil	Texte	15	prénom d'un utilisateur
Utilisateur	password	Texte	20	mot de passe de l'utilisateur
Utilisateur	rôle	Texte	15	rôle d'un utilisateur : admin…
Touite	id_touite	Numérique	5	id D'un touite unique
Touite	texte	Texte	235	texte du touite d'une longueur limitée à 235 caractères
Touite	note	Numérique	4	note d'un touite
Touite	date	date		date de publication d'u touite
Touite	cheminIm		50	chemin vers l'image
aTouite	emailUtil	Texte	25	email d'un utilisateur

aTouite	date	date	YYYY-MM-DD HH:MM:SS	date de publication d'u touite
aTouite	id_touite	Numérique	5	id d'un touite unique
Tag	id_tag	Numérique	5	id d'un tag
Tag	libelle_tag	Texte	25	nom du tag
Tag	desc_Tag	Texte	100	description du tag
aTouite	emailUtil	Texte	25	email de l'utilisateur qui touite
aTouite	date	date	YYYY-MM-DD HH:MM:SS	date d'un touite
aTouite	idTouite	Numérique	5	id d'un touite unique
Suivis	emailAbb	texte	25	email de l'utilisateur qui suis un compte
Suivis	emailSuiv	texte	25	email de l'utilisateur qui se fait suivre
TouiteParTag	id_tag	Numérique	5	id d'un tag
TouiteParTag	id_touite	Numétique	5	id d'un touite
Image	cheminIm	Texte	100	chemin du fichier de l'image
Image	descIm	Texte	100	description de l'image

Question 2)

Rappel des dépendances fonctionnelles :

1FN → Attributs de forme atomique

2FN → tous les attributs non clés sont pleinement dépendants des clés + 1FN

3FN → tous les attributs non clés sont directement et pleinement dépendants des clés+ 2FN

a. Ensemble des dépendances fonctionnelles valides

{emailUtil} -> nomUtil, prenomUtil, password, rôle (1)

{id_touite} -> texte, note, date, cheminIm, descIm (2)

{emailUtil, date} -> id_touite (3)

{id_tag} -> libelle_Tag, descTag (4)

{id_touite}->id_tag (5)

(cheminIm) -> descIm (6)

Question 3)

Liste des clés minimales :

1) Décomposition des dépendances focntionnnelles pour avoir les formes canoniques

emailUtil -> nomUtil (1') emailUtil -> prenomUtil (1'') emailUtil -> password (1''') emailUtil -> rôle(1'''')	id_touite -> texte (2') id_touite -> note (2'') id_touite -> date (2''') id_touite -> cheminIm (2'''') id_touite -> descIm (2''''')
(emailUtil, date) -> id_touite (3')	id_tag -> libelle_Tag (4') id_tag -> descTagi (4'')
id_touite ->id_tag (5')	cheminIm -> descIm (6')

On identifie 6 tables différentes.

2) Identification des DF que l'on peut supprimer

emailUtil -> nomUtil (1') emailUtil -> prenomUtil (1'') emailUtil -> password (1''') emailUtil -> rôle(1'''')	id_touite -> texte (2') id_touite -> note (2'') id_touite -> date (2''') id_touite -> cheminIm (2'''') id_touite -> descIm (2''''') redondant avec (6')
(emailUtil, date) -> id_touite (3')	id_tag -> libelle_Tag (4') id_tag -> descTagi (4'')
id_touite ->id_tag (5')	cheminIm -> descIm (6')

3) Identification des clés

On supprime les attributs que l'on peut obtenir à partir d'autres attributs (clés):

- **NomUtil, prenomUtil, password** sont obtenus à partir d'un **emailUtil.**
- **Id_touite** est obtenu à partir de **emailUtil et date**
- **Texte, note, cheminIm, descIm** sont obtenus à partir de l'**id_touite.**
- **Libelle_tag, descTag** sont obtenus à partir de l'**id_tag**.

emailUtil car à partir de cet attribut on peut avoir le nom,le prénom ainsi que le mot de passe de l'utilisateur

Date car la combinaison avec emailUtil permet de retrouver la liste des touites d'un utilisateur et l'id_touite donnent les informations sur un touite.

id_Tag car les informations d'un tag comme son libellé ou sa description dépend de son id.

Question 4)

La
TOUITEUR(EmailUtil, nomUtil, prenomUtil, password, role, id_touite, texte, date, note, cheminIm, id_tag, libelleTag, descIm)

On a :

- 1FN car les attributs sont sous forme canonique
- 2FN car 1FN et il n'y a pas de DF entre les attributs non-clés et une partie de la clé
- 3FN car 2FN et il n'y a pas de DF entre les attributs non-clés.

Question 5)

Rappel de la 3FN : il faut qu'il n'y ait aucune dépendance fonctionnelle entre attributs non clés. tous les attributs non clés sont directement et pleinement dépendants des clés + 2FN

1) Redéfinition des relations entre les attributs

- UTILISATEUR(emailUtil, nomUtil, prenomUtil, password, rôle)
- TOUITE(id_touite, texte, date, note, cheminIm)
- TAG(id_tag, libelleTag, descTag)
- ATOUITE(emailUtil, date, id_Touite)
- SUIVIS(emailUtilA, emailUtilB)
- TOUITEPARTAG(id_touite, id_tag)
- IMAGE(cheminIm, descIm)

Question 6)

1) Création des tables et insertion des valeurs

TABLE TOUITE : définit le contenu d'un touite.

```
CREATE TABLE TOUITE
(
   id_touite INTEGER AUTO_INCREMENT,
   texte     TEXT     NOT NULL,
   dateTouite DATETIME NOT NULL,
   note      INTEGER  NOT NULL,
   cheminIm  VARCHAR(256) NOT NULL,
   PRIMARY KEY (id_touite, dateTouite)
);
```

INSERTION DES VALEURS

```
INSERT INTO TOUITE (texte, dateTouite, note) VALUES ('Yop, ça boum
?', '2023-11-08 8:30:23', 0);
INSERT INTO TOUITE (texte, dateTouite, note) VALUES ('Les débuts',
'2023-11-06 15:11:01', 0);
```

TABLE UTILISATEUR : permet de connaître les informations concernant un utilisateur.

```
CREATE TABLE UTILISATEUR
(
   emailUtil  VARCHAR(60) PRIMARY KEY,
   nomUtil    TEXT NOT NULL,
   prenomUtil TEXT NOT NULL,
   password   TEXT NOT NULL,
   role       TEXT NOT NULL
);
```

INSERTION DES VALEURS

```
INSERT INTO UTILISATEUR (emailUtil, nomUtil, prenomUtil, password,
role) VALUES ('user1@mail.com', 'User', 'Un',
'f9777362314a88ba0d7df701ed6bebd0a3577be33befe72babc40c39de65ab1f'
, 'user');
mot de passe : user1user1
INSERT INTO UTILISATEUR (emailUtil, nomUtil, prenomUtil, password,
role) VALUES
('user2@mail.com','User','Deux','$2y$10$EflBj0wr6KbTZbEVxA64T.p267
JNkTQVbVf4/V5AqbNTFRoRHceJq','user');
mot de passe : user2user2
```

TABLE IMAGE : table contenant la description de l'image et son chemin.

```
CREATE TABLE IMAGE
(
   cheminIm VARCHAR(256) PRIMARY KEY,
   descIm   TEXT NOT NULL
);
```

INSERTION DES VALEURS

Ce n'est pas possible d'insérer les valeurs pour les images car le dossier de stockage est dans le gitignore ce qui veut dire que l'on ne pourra pas retrouver les images en question.

TABLE TAG : table qui associe un id_tag à un libellé et une description.

```
CREATE TABLE TAG
(
   id_tag     INTEGER PRIMARY KEY AUTO_INCREMENT,
   libelleTag TEXT NOT NULL,
   descTag    TEXT NOT NULL
);
```

INSERTION DES VALEURS

```
INSERT INTO TAG (libelleTag, descTag) VALUES ('Politique', 'Touite
sur la politique');
INSERT INTO TAG (libelleTag, descTag) VALUES ('Sport', 'Touite sur
le sport');
```

TABLE ATOUITE : contient la liste des utilisateurs qui ont publié un touite en associant un email et une date à un id_touite.

```
CREATE TABLE ATOUITE
(
   emailUtil VARCHAR(60) NOT NULL,
   dateTouite      DATETIME NOT NULL,
   id_touite INTEGER  NOT NULL,
   PRIMARY KEY (emailUtil, dateTouite)
);
```

INSERTION DES VALEURS

```
INSERT INTO ATOUITE (emailUtil, dateTouite, id_touite) VALUES
('user1@mail.com', '2023-11-06 15:11:01', 2);
INSERT INTO ATOUITE (emailUtil, dateTouite, id_touite) VALUES
('user2@mail.com', '2023-11-08 8:30:23', 1);
```

TABLE TOUITEPARTAG : contient la liste des touites qui contiennent un tag, les id_touites sont liés à un id_tag.

```
CREATE TABLE TOUITEPARTAG
(
   id_touite INTEGER NOT NULL,
   id_tag    INTEGER NOT NULL,
   PRIMARY KEY (id_touite, id_tag)
);
```

INSERTION DES VALEURS

```
INSERT INTO TOUITEPARTAG (id_touite, id_tag) VALUES (2, 1);
```

TABLE SUIVI: table dans laquelle on enregistre les personnes suivies par un utilisateur.

```
CREATE TABLE SUIVIS
(
   emailUtil      VARCHAR(60) NOT NULL,
   emailUtilSuivi VARCHAR(60) NOT NULL,
   PRIMARY KEY (emailUtil, emailUtilSuivi)
);
```

INSERTION DES VALEURS

```
INSERT INTO SUIVIS (emailUtil, emailUtilSuivi) VALUES
('user1@mail.com','user2@mail.com');
```

---

2) Ajout des contraintes de clés étrangères

TABLE ATOUITE

```
ALTER TABLE ATOUITE
   ADD CONSTRAINT fk_atouite1 foreign key (emailUtil) REFERENCES
utilisateur (emailUtil);
ALTER TABLE ATOUITE
   ADD CONSTRAINT fk_atouite2 foreign key (id_touite) REFERENCES
touite (id_touite);
```

TABLE SUIVIS

```
ALTER TABLE SUIVIS
   ADD CONSTRAINT fk_suivis1 FOREIGN KEY (emailUtil) REFERENCES
UTILISATEUR (emailUtil);
ALTER TABLE SUIVIS
   ADD CONSTRAINT fk_suivis2 FOREIGN KEY (emailUtilSuivi)
REFERENCES UTILISATEUR (emailUtil);
```

TABLE TOUITEPARTAG

```
ALTER TABLE TOUITEPARTAG
   ADD CONSTRAINT fk_suivis foreign key (id_touite) REFERENCES
touite (id_touite);
```

3) Création des tables pour les fonctionnalités

Pour la fonctionnalité s'abonner à un tag nous avons ajouté la table tagSuivi(emailUtil, id_tag) où un utilisateur est associé à un tag qu'il suit:

```
CREATE TABLE TAGSUIVI (
      emailUtil varchar(50),
      id_tag int,
      PRIMARY KEY(emailUtil, id_tag)
);
```

Pour la fonctionnalité du like, nous avons ajouté la table aLike(emailUtil, id_touite)

```
CREATE TABLE ALIKE (
      emailUtil varchar(50),
      id_touite INT,
      PRIMARY KEY(emailUtil, id_touite),
      FOREIGN KEY (emailUtil) REFERENCES UTILISATEUR(emailUtil),
      FOREIGN KEY (id_touite) REFERENCES TOUITE(id_touite)
);
```

et la table aDislike(emailUtil, id_touite) :

```
CREATE TABLE ADISLIKE (
      emailUtil varchar(50),
      id_touite INT,
      PRIMARY KEY(emailUtil, id_touite),
      FOREIGN KEY (emailUtil) REFERENCES UTILISATEUR(emailUtil),
      FOREIGN KEY (id_touite) REFERENCES TOUITE(id_touite)
);
```

Ces tables permettent d'enregistrer les personnes qui ont liké et qui ont disliké, et permet de limiter le nombre de like et de dislike par personne à un. Elles sont mises à jour après l'action d'un utilisateur.

Pour ne pas avoir de problème de caractères non reconnus :

```
ALTER DATABASE sae CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
```