République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Aboubekr BELKAÏD - TLEMCEN
Faculté des Sciences
Département d' Informatique

TITRE

EXERCICES DE BASES DE DONNÉES AVANCÉES

Adressé aux étudiants niveau : Master (M1)

Domaine : Mathématiques et Informatique
Filière : Systèmes Informatiques
Spécialité : SIC, GL, MID et RSD

Etabli Par :

Amal HALFAOUI ép. GHERNAOUT

Année 2020/2021

Table des matières

Table des figures

Liste des tableaux

Liste des schémas

Listings

Avant-propos

Ce polycopié s'adresse aux étudiants de master de la filière informatique. Les exercices proposés concernent la matière "Bases de Données Avancées". Cette matière est enseignée aux premières années master des quatre spécialités du département d'informatique de l'université de Tlemcen : Systèmes d'Information et Connaissance (SIC), Génie Logiciel (GL), Réseausx et Systèmes Distribués (RSD) et Modèles Intelligents et Décision (MID).
Il est à noter que les exercices présentés dans ces séries viennent compléter les notions déjà acquises durant le cycle de la licence Informatique, à savoir :

1. Les notions de bases sur le modèle relationnel (SQL, Dépendances fonctionnelles et formes normales) qui ont été abordées en deuxième année ;
2. Les notions avancées sur le modèle relationnel (Optimisation, vues et procédure stockées) traitées en troisième année.

Néanmoins, les exercices restent aussi accessibles aux étudiants de licence informatique ou des filières qui ont de bonnes connaissances en bases de données relationnelles et qui souhaitent prolonger et approfondir leurs connaissances en base de données.
L'aspect avancé des bases de données sera traité dans ce polycopié avec des rappels et des exercices qui peuvent être regroupés en trois catégories :

1. **Bases de données actives** : En premier lieu, nous aborderons, dans la première série d'exercices, l'une des fonctionnalités avancées des bases de données relationnelles à savoir les déclencheurs (l'aspect actif des bases).
   Les déclencheurs (trigger) sont incontournables dans la gestion d'une base de données. Ils permettent de gérer des contraintes dynamiques de manière automatique.
   Tous les logiciels de bases de données n'implémentent pas de la même façon le principe des déclencheurs. De plus, ils ne réagissent pas tous pareils face à certains

problèmes connus dans les déclencheurs comme le principe de la table mutante ; ou encore l'ordre d'exécution de plusieurs déclencheurs appliqués à une même table. Nous utiliserons, dans la résolution de nos exercices la définition des déclencheurs propre au SGBD Oracle (pl/sql).

2. **Extension du modèle relationnel** : En second lieu. Nous traiterons dans la série d'exercices 2 et 3 l'extension des bases relationnelles pour prendre en compte l'aspect objet introduit à partir de la norme (SQL3) ; qui permet de définir et d'interroger efficacement les données complexes et palier les limites du schéma relationnel.
   En effet, si le modèle logique relationnel a prouvé sa puissance et sa fiabilité depuis les années 1980, de nouveaux besoins en informatique industrielle ont vu l'émergence de structures de données complexes mal adaptées à une gestion relationnelle.
   Nous verrons, dans cette partie, des exercices de modélisation dans la série d'exercices 2 et d'interrogation dans la série d'exercices 3 qui montrent :
   a) comment l'aspect objet est appréhendé dans des bases relationnelles ;
   b) comment faire évoluer une base relationnelle classique dans un contexte objet.
      Le principe de l'objet a été introduit, par l'enrichissement du langage SQL, à partir de SQL3. Là encore, les SGBD n'implémentent pas tous la norme de la même manière. En plus, certains SGBD ne le prennent même pas en charge. Les solutions d'exercices de cette série ont été réalisées en utilisant la syntaxe SQL3 du SGBD Oracle.
3. **Modèles non relationnels** : Ces deux dernières décennies, nous avons assisté à une évolution logicielle importante qui est étroitement liée à l'évolution matérielle. Les SGBD actuels ne sont plus contraints par la capacité de stockage contrairement aux premiers SGBD qui dépendaient des capacités de stockage très réduites.
   En effet, de nouvelles possibilités ont vu le jour, dans le domaine de l'informatique distribuée et de la virtualisation. Ceci a remis en cause la prédominance du modèle relationnel qui permet une optimisation de stockage, grâce à son modèle normalisé qui assure la réduction de la redondance des données et garantit la cohérence de données.
   Un volume important de données hétérogènes, de différents domaines, doit être pris en charge et traité par les SGBD. Ce volume de données qui se compte maintenant en pétaoctets n'arrive plus à être traité de manière efficace avec les bases de don-

nées relationnelles qui sont hautement transactionnelles et fortement normalisées. De nouveaux modèles de SGBD ont vu le jour qui, contrairement au relationnel, sont dénormalisés ; la mouvance NoSQL (Not Only SQL) est lancée.

Les exercices de la série d'exercices 4 et 5 présentent cette nouvelle façon de modéliser les données avec ces nouveaux schémas où quatre modèles sont prédominants à savoir : clé-valeur, colonne, graphe et document. Nous mettrons l'accent sur le modèle orienté document en utilisant le SGBD Mongodb, vu son actuel succès et sa forte utilisation.

Organisation du manuel d'exercices

Le reste de ce manuel est constitué de cinq séries d'exercices qui contiennent des rappels et des exercices avec corrigés, un ensemble d'exercices supplémentaires avec corrigés est rajouté à certaines séries.

Les séries d'exercices sont répertoriées dans le tableau 1

Concepts traités	**Séries d'exercices**
Bases de données actives	Série 1 : Déclencheurs
Bases Relationnelles Objets (RO)	Série 2 : Modélisation et définition du schéma RO
	Série 3 : Manipulation des données du modèle RO
Bases NoSQL(Mongodb)	Série 4 : Modélisation des bases NoSQL (Mongodb)
	Série 5 : Commandes Mongodb

Tableau 1: *Correspondance entre concepts traités ainsi que leurs séries d'exercices*

Série TD 1

Les déclencheurs

Sommaire

1.1 Objectifs et Rappels

Les SGBD actifs [1], réactifs ou intelligents sont des SGBD capables de réagir à des événements afin de contrôler l'intégrité, gérer des redondances, autoriser ou interdire des accès, alerter des utilisateurs, et plus généralement gérer le comportement réactif des applications. Les déclencheurs sont un des mécanismes permettant d'incorporer dans la base de données elle-même (coté serveur) des composants actifs qui d'ordinaire sont inclus dans les programmes d'application.

Les déclencheurs sont tout simplement des programmes qui définissent une action qui doit s'exécuter automatiquement quand survient un évènement dans la base de données (séquence événement-Condition-Action (ECA)) voir figure 1.1.

Les évènements sont les instructions qui modifient la base (insertion, modification d'une ou de plusieurs colonnes, suppression) sur une table (ou une vue).

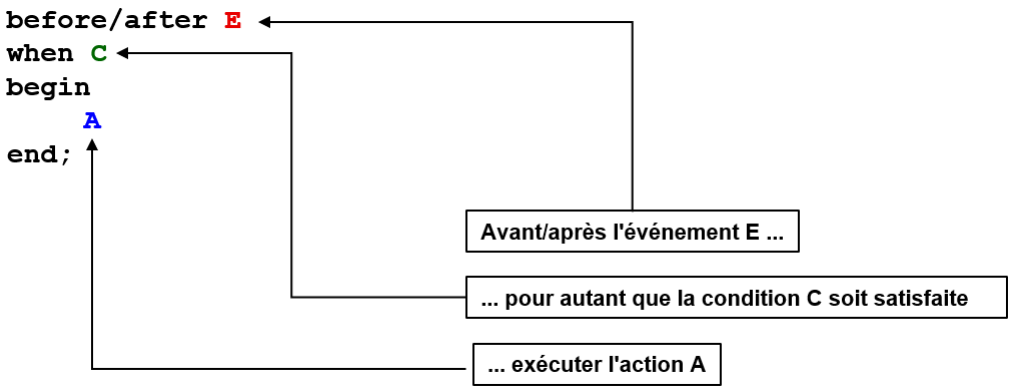

FIGURE *1.1:* *Séquence événement-condition-Action*

Les déclencheurs permettent de :

- Définir des contraintes d'intégrité complexes : les règles de gestion qui n'ont pas pu être mises en place par des contraintes statiques au niveau des tables ;
- Définir des comportements particuliers en cas de violation de contraintes ;
- Alléger le programme de l'application : car les contraintes définies par les déclencheurs sont au niveau du serveur.
- Émettre des alertes.

1.1.1 Syntaxe et principe

Listing 1.1: *Syntaxe de définition d'un déclencheur*

```
 CREATE [OR REPLACE] TRIGGER [schema.] nomDeclencheur
{ BEFORE | AFTER | INSTEAD OF }
{ { DELETE | INSERT | UPDATE [OF col1 [,col2]...] }
[OR { DELETE | INSERT | UPDATE [OF col1 [,col2]...] }]...
ON { [schema.] nomTable | nomVue }
[FOR EACH ROW] }
[WHEN ( condition ) ]
{    Bloc PL/SQL (
BEGIN
[IF INSERTING THEN ..... END IF;]
[IF UPDATING THEN ..... END IF;]
 END ; )
```

- Le nom du déclencheur :
  - doit être unique dans un même schéma ;
  - peut être le nom d'un autre objet (table, vue, procédure) mais à éviter.
- **BEFORE** ou **AFTER** : précise le quand (la chronologie entre l'action à réaliser par le déclencheur et la réalisation de l'événement.
- **DELETE**, **UPDATE** ou **INSERT** : spécifie l'événement. Si l'événement est UPDATE on peut préciser les attributs concernés par la modification.
- **ON** : précise le champ d'application du déclencheur.
- **WHEN** conditionne l'exécution du déclencheur.
- **FOR EACH ROW** est optionnelle. si elle est spécifiée, c'est un trigger ligne, sinon c'est un trigger global.
- Les triggers **lignes** se déclenchent individuellement pour chaque ligne de la table affectée par le trigger.
- Les triggers **globaux** sont déclenchés une seule fois.
- Les prédicats conditionnels **INSERTING**, **DELETING** et **UPDATING** peuvent être utilisés quand un trigger comporte plusieurs instructions de déclenchement (INSERT OR DELETE OR UPDATE). Un bloc de code spécifique pour chaque instruction est exécuté.

1.1.2 Les valeurs :NEW et :OLD

Deux valeurs sont manipulées dans les déclencheurs type ligne :

- La nouvelle valeur avant ajout ou modification est appelée :new.colonne
- L'ancienne valeur avant suppression ou modification est appelée :OLD.colonne

	:OLD	:NEW
INSERT	null	Valeurs champs ligne insérée
DELETE	Valeurs champs ligne supprimée	null
UPDATE	Valeurs champs ligne avant modification	Valeurs champs ligne après modification

Tableau 1.1: *Valeurs :NEW et :OLD selon les 3 opérations*

1.1.3 La table Mutante dans Oracle

Il est, généralement, interdit de manipuler la table sur laquelle se porte le déclencheur dans le corps du déclencheur lui-même [2]. C'est le principe de la table mutante. Dans Oracle, cette restriction ne concerne que les déclencheurs de lignes (FOR EACH ROW) par rapport aux évènements AFTER. Dans ce cas, l'erreur n'est pas soulignée à la compilation mais est soulevée dès la première opération (erreur : ORA-04091 : table ... en mutation, déclencheur/fonction ne peut la voir). Oracle autorise des requêtes de sélection sur des déclencheurs BEFORE. Par contre, la ligne manipulée n'est pas prise en compte, il faudra donc rajouter les valeurs des champs de la ligne dans le cas des requêtes d'agrégations.

1.2 Exercices

1.2.1 Exercice 1 : Transformation, vérification et mise à jour automatique des données

Soit la table "Citoyen" (Voir figure 1.2) qui présente les informations d'une personne et le numéro son conjoint.

Citoyen (NumC, Nom, Prenom, NomJeuneF, numConjoint*)

1. Transformer, au moment de l'insertion en majuscule, la valeur du nom du citoyen quel que soit son format d'origine (utilisation de UPPER()).
2. Vérifier, au moment de l'insertion ou modification, que le nom des deux conjoints est le même.

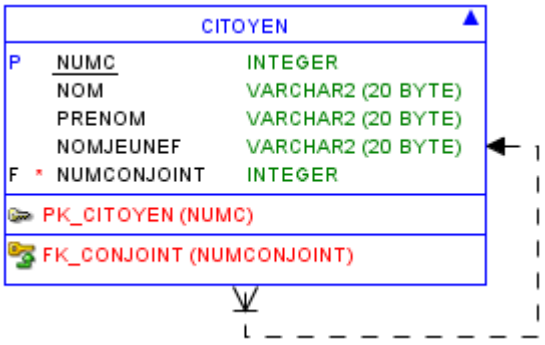

FIGURE *1.2:* *Schéma Citoyen*

3. Incrémenter la valeur de la clé NumC automatiquement.

1.2.2 Exercice 2 : Problème d'interblocage

Soient les deux triggers "T1" et "T2" sur la table "tab1" et la table "tab2" de la même base de données

1. Expliquer le problème que pose le déclenchement du Trigger T1.

```
CREATE TRIGGER "T1"
AFTER DELETE ON tab1
FOR EACH ROW
BEGIN
        UPDATE tab2
        SET attribut1='A'
END ;
```

```
CREATE TRIGGER "T2"
AFTER UPDATE of attribut1 on tab2
FOR EACH ROW
BEGIN
        INSERT INTO tab1 Values (1,'A');
END ;
```

2. Corriger les erreurs de syntaxe des deux déclencheurs suivants

```
CREATE OR REPLACE TRIGGER "T2"
AFTER UPDATE OF attribut1 on tableT2
FOR EACH ROW
BEGIN
        :NEW.attribute1 := 'A';
END ;
```

```
CREATE OR REPLACE TRIGGER "TRIGGER1"
AFTER insert ON tab
DECLARE
```

```
  total INTEGER;
BEGIN
 Select count(*) into total from tab where num := :NEW.num;
 if total > 10 then
  RAISE_APPLICATION_ERROR(-20006, 'le nombre max de lignes est
      atteint');
 END IF;
END;
```

1.2.3 Exercice 3

Soit la base de données pour la gestion d'évènements sportifs "DZZumbaDays". Cet évènement est organisé le 15 octobre de chaque année dans un lieu différent, en deux sessions : session "matin" et "après-midi"

Evenement (NumEven, LieuEven, adresse, capacité).

Participation (NumPart, numPersonne*, NumEven*, Session)

Personne (numPersonne, nom, âge, téléphone)

capacité : nombre de participants que peut accueillir le lieu où est organisé l'évènement : par exemple l'évènement "DZZumbaDays2019" sera organisé dans le stade "Frère Zerga Tlemcen". La capacité est de 100 participants ; deux sessions seront organisées : matin et après-midi donc le champ "session" peut prendre deux valeurs : matin , après-midi.

- Définir un déclencheur qui permet d'interdire une participation si l'évènement affiche complet (la capacité est atteinte).

1.2.4 Exercice 4 : Exercice complet

La société "InterTlemcen" est une entreprise de construction qui gère de nombreux projets. Chacun des employés de la société peut participer simultanément sur plusieurs projets en même temps.

Lorsque l'entreprise démarre un projet, il est indispensable d'estimer le temps nécessaire pour le réaliser et de connaitre son budget. Dans notre cas, cette estimation est faite en nombre d'heures.

L'employé est payé par tâche selon le nombre d'heures de la réalisation de la tâche, l'heure est tarifiée à 600 da. La tâche est affectée à une semaine qui est ordonnée selon le nombre de semaines de réalisation du projet. Le schéma relationnel relatif à cette gestion est donné ci-dessous et est schématisé dans la figure 1.3.

EMPLOYES(refemp, nom, salTot, NbProjet) ;

PROJET(refproj, refdir* ,titre, nb_heures_prevues, nb_heures_effectuees, budjet) ;

PARTICIPE(refproj*,refemp*, dateP) ;

TACHE(refproj*,refemp*,semaine ,nb_heures) ; semaine représente le numéro de semaine.

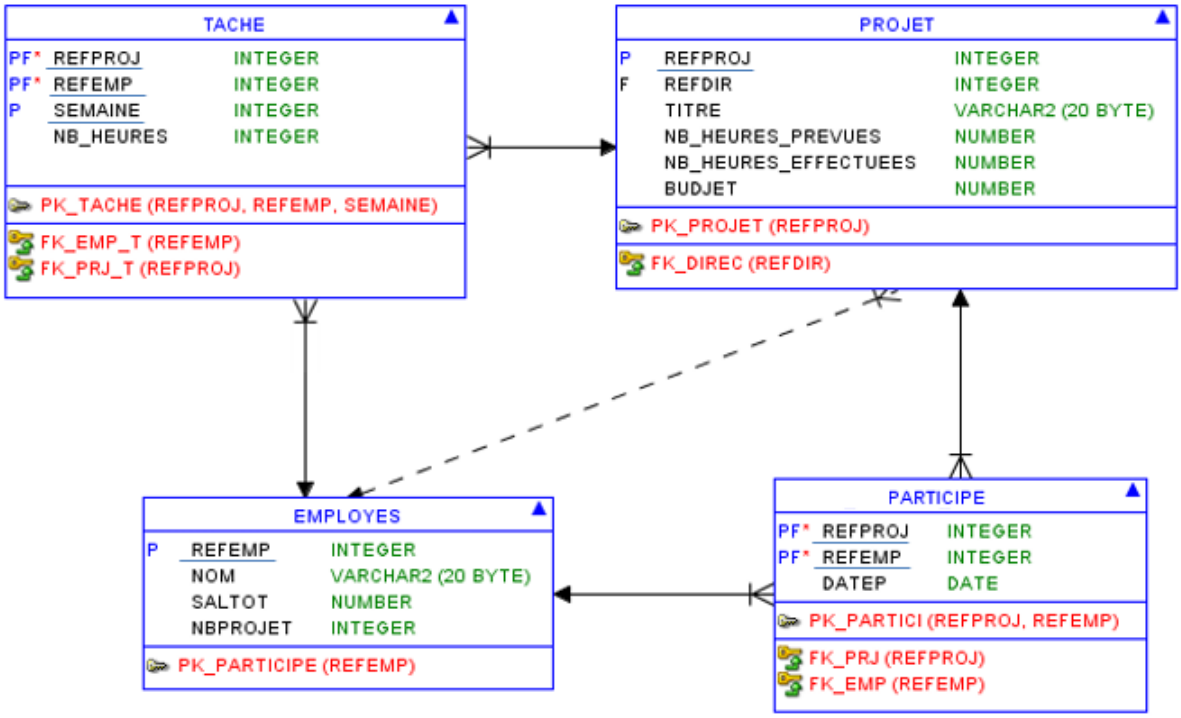

FIGURE *1.3:* *Schéma relationnel de la base 'InetrTlemcen'*

Questions : Définir les règles suivantes avec des déclencheurs :

(a) **Propager des mises à jour sur des données**

1. Mettre à jour le **salTot** d'un employé (cas d'insertion seulement).
2. Mettre à jour le champ **nb_heures_effectuees**.
3. Mettre à jour automatiquement le champ **NbProjet** ;

(b) **Empêcher les mises à jour de la base qui altèreraient l'intégrité des données.**

4. Au moment de l'affectation d'un responsable à un projet, vérifier qu'il y participe, sinon le mettre comme participant aussi.
5. Limiter le nombre de projets d'un employé, dans une même semaine, à 3.
6. Interdire au directeur d'un projet d'effectuer des tâches sur son propre projet.

(c) **Historiser et auditer les opérations sur une table**

7. Nous désirons vérifier que le nombre d'heures effectuées ne dépasse pas le nombre d'heures prévues pour la réalisation d'un projet. En cas de dépassement, nous écrivons un message dans la table **TAB_LOG** prévue pour recueillir les anomalies constatées lors de la gestion des projets. Le schéma de la table **TAB_LOG** est **TAB_LOG**(Num_Pb, date_pb, refproj, message). Le champ Num_Pb est un champ qui doit se remplir et s'incrémenter automatiquement.

   - Écrire le déclencheur qui permet de répertorier l'information de dépassement

8. La TAB_LOG est une table qui ne doit pas être modifiée, en cas de modification de celle-ci, on veut connaitre celui qu'il a modifiée (l'utilisateur courant USER) et la date de cette modification. Ces informations seront automatiquement insérées dans la table **AuditTable_Log**(Num_Pb, Modifie_par, Date_Modif)

   - Écrire le déclencheur qui permet d'inscrire ces modifications dans la table.

(d) **Question supplémentaire**

Supposons que nous avons ajouté à la table tâche les champs : ordre_tache (integer), Date_Debut et Date_Fin.

Nous voulons, à présent, avoir un programme des tâches de chaque employé. Pour cela, chaque tâche à un numéro séquentiel qui est affecté au champ ordre_tache. Ce dernier, permet d'ordonner les tâches d'un employé.

Les dates de tâches d'un employé ne doivent pas se chevaucher, la date de début d'une tâche commence le jour qui suit la fin de la tâche qui la précède. Un jeu de données est représenté dans la figure 1.4

REFPROJ	REFEMP	SEMAINE	NB_HEURES	ORDRE_TACHE	DATE_DEBUT	DATE_FIN
1	1	1	48	1	12/03/18	14/03/18
2	1	2	24	2	15/03/18	16/03/18
1	2	1	100	1	12/03/18	16/03/18
2	2	1	78	2	17/03/18	21/03/18

FIGURE *1.4:* *Exemple de jeux de données de l'ordre des taches et dates de la table Tache*

- Créer le déclencheur qui permet d'affecter automatiquement les valeurs de ces trois champs.

Indication sur l'addition des dates : on peut tout simplement additionner ou soustraire une valeur à une date, cette dernière est considérée comme jours.

Si on veut manipuler des heures, il suffit de diviser par 24

Date + nbrjours -> Exp : '11/03/18'+ 3 = '14/03/18'

Date+ (nbr d'heures/24) -> Exp :'11/03/18'+ (48/24) = '13/03/18'

1.3 Exercices supplémentaires

1.3.1 Exercice supplémentaire 1

Soit la base de données de gestion des réservations aux restaurants :

Restaurant (NumResto, Nom, adresse, capacité) ;

Réservation (NumRes, numPersonne*, NumResto*, nombre_personnes, dateRes, service)

Personne (numPersonne, nom, âge, téléphone)

"capacité" : nombre de places que peut accueillir le restaurant dans un service. Par exemple 100 couverts.

Deux services sont présentés par jour : déjeuner et diner. Donc, service peut prendre deux valeurs : "déjeuner", "diner".

- Définir un déclencheur qui permet d'interdire une réservation si le restaurant affiche complet (la capacité est atteinte)

1.3.2 Exercice supplémentaire 2

Soit la base de données suivante pour la réalisation de recettes de gâteaux, les clés primaires sont soulignées.

recette (NumRecette, nomGateau, NbrIngredient)

Ingredient (NumIngr, NomIngr, QteStock, PrixIngr)

RealisGateau (NumReal, NumGateau*, NumIngr *, QteUtilise)

Assurer les contraintes suivantes :

1. Mettre à jour la QteStock automatiquement.
2. Quelle est la contrainte qui permet d'assurer qu'un gâteau ne contienne pas plusieurs fois le même ingrédient.

1.3.3 Exercice supplémentaire 3

Soit le schéma relationnel suivant :

CLIENT(NumCli, CINCli, NomCli, AdrCli, TelCli, NumConjoint*, NbCptE)

OPERATION(NumOp, TypeOP, MtOp, NumCpt*, DateOp) TypeOp='Depot' ou 'Retrait'

COMPTE(NumCpt, SoldeCpt, TypeCpt, NumCli*) TypeCpt='CCourant' ou 'CEpargne'

EMPRUNT (NumE, NumCli *, DateE, MtE)

Assurer l'ensemble des contraintes suivantes en définissant des déclencheurs

1. Vérifier que le numéro de client d'un compte existe bien (Contrainte Foreign key).
2. Mettre à jour le solde du compte client après chaque opération effectuée.
3. Interdire aux clients de retirer un montant supérieur au solde du compte courant.
4. Interdire à un client d'avoir plusieurs comptes courants.
5. Mettre à jour automatiquement le champ NbCptE.
6. Modifier le déclencheur 5 pour limiter le nombre de comptes épargne à 10.
7. Interdire à un client de faire plus de 2 opérations de retrait par jour sur le même. compte.
8. Les clients ayant un solde inférieur à 40 000 da ne peuvent pas bénéficier d'un emprunt.

1.4 Solutions d'exercices

1.4.1 Solution de l'exercice 1

1. Transformer au moment de l'insertion en majuscule la valeur du nom

```
CREATE OR REPLACE TRIGGER  'UPPER_NOMP'
BEFORE INSERT ON CITOYEN
FOR EACH ROW
BEGIN
   :NEW.NOM := upper(:NEW.NOM) ;
END ;
```

Attention : l'utilisation de AFTER ne fonctionne pas ici; car au moment AFTER le déclencheur sauvegarde les valeurs saisies par l'utilisateur et ne permet plus de les modifier. Quand on veut affecter des valeurs aux champs au moment des deux opérations INSERT ou UPDATE, il faudra impérativement utiliser un déclencheur BEFORE.

2. Vérification des noms des deux conjoints

```
CREATE OR REPLACE TRIGGER  'CHECK_NOM_CONJ'
BEFORE INSERT OR UPDATE ON CITOYEN
FOR EACH ROW
DECLARE nomconj VARCHAR2(20);
BEGIN
   Select Nom Into Nomconj From CITOYEN
   Where NumC = :NEW.Numconjoint;
   If (Nomconj <> :NEW.Nom) Then
     RAISE_APPLICATION_ERROR(-20300, 'Oops !le nom de la personne et
          le nom de son conjoint ne sont pas identiques');
  END IF ;
END;
```

3. Incrémenter la valeur de la clé NumC automatiquement

```
CREATE OR REPLACE 'TRIGGER  AUTO_INC_NUMP'
BEFORE INSERT ON CITOYEN
FOR EACH ROW
DECLARE
MAX_VAL NUMBER(5);
BEGIN
          Select Max(NumC) Into Max_Val From CITOYEN;
          if Max_Val is not null then
          :NEW.NumC := Max_Val + 1;
          else
          :NEW.NumC := 1;
          END IF;
END;
```

Solution avec utilisation de séquence

```
CREATE SEQUENCE   "SeqCitoyen"  MINVALUE 1 MAXVALUE 1000000
    INCREMENT BY 1;
CREATE OR REPLACE TRIGGER  "TRIG_CLE_PRIM_PER"
BEFORE INSERT on "personne"
FOR EACH ROW
BEGIN
  select SeqCitoyen.nextval into :NEW.NumC from dual;
END;
```

1.4.2 Solution de l'exercice 2

1. L'exécution de déclencheur 1 conduit à une situation d'interblocage car le déclencheur T1 effectue une modification sur la table tab1 sur laquelle est mis un déclencheur en modification qui lui-même effectue une opération d'insertion sur une table qui est verrouillée
2. Correction des deux déclencheurs

   :NEW ne peut pas être utilisé au moment de la suppression donc remplacer DELETE par UPDATE

```
CREATE OR REPLACE TRIGGER T2
BEFORE UPDATE ON TABLET2
FOR EACH ROW
BEGIN
:NEW.attribute1 := 'A';
END ;
```

```
CREATE OR REPLACE TRIGGER TRIGGER1
Before INSERT ON tab
FOR EACH ROW
DECLARE
   total INTEGER;
BEGIN
 Total := 0;
 Select count(*) into total from tab ;
 if total +1 > 10 then
  RAISE_APPLICATION_ERROR(-20006, 'le nombre max de lignes est
      atteint');
 END IF;
END;
```

1.4.3 Solution de l'exercice 3

```
CREATE OR REPLACE TRIGGER Evn
BEFORE INSERT ON Participation
FOR EACH ROW
DECLARE
   NombreParticip INTEGER;
   NbCapacite INTEGER;
BEGIN
 NombreParticip:= 0 ;
 Select count(*) into NombreParticip from Participation where
 NumEven = :NEW.NumEven and Session = :NEW.Session; // le total de
     participant par session
 Select capacite into NbCapacite from Evenement where NumEven = :NEW.
     NumEven
 if NombreParticip + 1 > capacite then
  RAISE_APPLICATION_ERROR(-20006, 'L événement est au complet');
END IF;
END;
```

1.4.4 Solution de l'exercice 4

(a) **Propager des mises à jour sur des données**

1. Mettre à jour le **salTot** d'un employé (cas d'insertion seulement)

```
CREATE OR REPLACE TRIGGER  "MAJ_SalTot"
AFTER INSERT ON TACHE
FOR EACH ROW
BEGIN
```

```
 IF INSERTING then
          update EMPLOYES set SALTOT = SALTOT+(:NEW.NB_HEURES *600)
          where REFEMP = :NEW.REFEMP;
 END IF;
END;
```

2. Mettre à jour le champ **nb_heures_effectuees**.

```
CREATE OR REPLACE TRIGGER  "MAJ_ nb_heures_effectuees "
AFTER INSERT OR DELETE OR UPDATE ON TACHE
FOR EACH ROW
BEGIN
IF INSERTING THEN
    update projet
    set NB_HEURES_EFFECTUEES = NB_HEURES_EFFECTUEES + :NEW.NB_HEURES
    where REFPROJ= :NEW.REFPROJ;
END IF;
IF DELETING THEN
    update projet set nb_heures_effectuees= nb_heures_effectuees- :
       old.NB_HEURES where       REFPROJ= :OLD.REFPROJ;
end if;
IF UPDATING THEN
 if :NEW.NB_HEURES > :OLD.NB_HEURES  then update projet set
     nb_heures_effectuees= nb_heures_effectuees+ (:NEW.NB_HEURES-:
     OLD.NB_HEURES) where REFPROJ= :NEW.REFPROJ;
 else update projet set nb_heures_effectuees= nb_heures_effectuees-
      (:OLD.NB_HEURES-:NEW.NB_HEURES) where REFPROJ= :NEW.REFPROJ;
 END IF;
END IF;
END;
```

3. Mettre à jour automatiquement le champ **NbProjet**

```
CREATE OR REPLACE TRIGGER "MAJ_NbProjet"
BEFORE INSERT ON TACHE
FOR EACH ROW
DECLARE
   nb   INTEGER;
BEGIN
 nb :=0;
select count(*) into nb from TACHE where REFEMP = :NEW.REFEMP and
    refproj=:NEW.refproj;
if nb=0 then
 UPDATE EMPLOYES SET NBPROJET = NBPROJET +1
 WHERE REFEMP = :NEW.REFEMP;
end if;
END;
```

(b) **Empêcher les mises à jour de la base qui altèreraient l'intégrité des données.**

4. Au moment de l'affectation d'un responsable à un projet, vérifier qu'il y participe, sinon le mettre comme participant aussi.

```
CREATE OR REPLACE TRIGGER "AFFECT_RESP_PROJ"
BEFORE insert ON PROJET
```

```
FOR EACH ROW
DECLARE NEmp number;
BEGIN
  NEmp := 0;
  select count(*)  into NEmp from participe where REFPROJ = :NEW.
      REFPROJ and refemp = :NEW.REFDIR;
  if  NEmp = 0 then
  insert into participe values(:NEW.REFPROJ, :NEW.REFDIR,sysdate);
  END IF;
  END;
```

5. Limiter le nombre de projets d'un employé, dans une même semaine, à 3.

```
CREATE OR REPLACE TRIGGER  "MAJ_NbProjet_Max3"
BEFORE INSERT ON TACHE
FOR EACH ROW
DECLARE
 nbprj number;
 nbprjS NUMBER;
BEGIN
nb := 0
nbprjS:= 0;

select count(*) into nbprjS from TACHE where REFEMP = :NEW.REFEMP
    and semaine =:NEW.semaine ;

IF (nbprjS > 3) THEN
RAISE_APPLICATION_ERROR(-20006, 'LE NBRE MAX DE PROJET EST 3 PAR
    SEMAINE');
Else
nb :=0;
select count(*) into nb from TACHE where REFEMP = :NEW.REFEMP and
    refproj=:NEW.refproj;
if nb=0 then
 UPDATE EMPLOYES SET NBPROJET = NBPROJET +1
 WHERE REFEMP = :NEW.REFEMP;
end if;
end if;
END;
```

6. Interdire au directeur d'un projet d'effectuer des tâches sur son propre projet

```
CREATE OR REPLACE TRIGGER  "Inter_Direct"
AFTER INSERT ON tache
FOR EACH ROW
DECLARE
 Nbr Number;
BEGIN
Nbr := 0;
select count(*) into Nbr from projet where refdir = :NEW.refemp and
     refproj =:NEW.refproj;
 IF (Nbr > 0) THEN
RAISE_APPLICATION_ERROR(-20006, 'Vous ne pouvez pas avoir d''autres
    tâches dans ce projet car vous  êtes son directeur');
END IF;
END;
```

(C) **Historiser et auditer les opérations sur une table**

7. Le déclencheur qui permet de répertorier l'information de dépassement

```
Create sequence SEQ;
CREATE OR REPLACE TRIGGER  "TRIG_CLE_PRIM_LOG"
BEFORE insert on "TAB_LOG"
FOR EACH ROW
BEGIN
 select SEQ.nextval into :NEW.Num_Pb from dual;
END;
```

```
CREATE OR REPLACE TRIGGER  "Depassement"
AFTER UPDATE on Projet
FOR EACH ROW
BEGIN
 if  :NEW.nb_heures_effectuees > :NEW.NB_HEURES_PREVUES then insert
     into TAB_LOG (DATE_PB,REFPROJ,MESSAGE) values (sysdate,:NEW.
    refproj,'probleme de depassement');
END IF;
END;
```

8. Le déclencheur qui permet d'inscrire les modifications dans TAB_LOG.

```
CREATE OR REPLACE TRIGGER  "User_Modif"
BEFORE UPDATE on "TAB_LOG"
FOR EACH ROW
BEGIN
Insert into AUDITTABLE_LOG  values ( :NEW.Num_Pb, USER, sysdate);
END;
```

(d) **Questions supplémentaires**

Le déclencheur qui permet d'affecter automatiquement les valeurs aux champs rajoutés à la table Tache : Ordre_Tache, date_debut, date_fin

```
CREATE OR REPLACE TRIGGER "rdreTache"
BEFORE INSERT ON TACHE
FOR EACH ROW
DECLARE ordre INTEGER;
        nb INTEGER;
        datef date;
BEGIN
nb := 0;
select count(*) into nb from tache where REFEMP = :NEW.REFEMP;
if nb = 0 then
   :NEW.Date_debut := sysdate + 1;
   :NEW.ORDRE_TACHE := 1;
else
   select max(ORDRE_TACHE) into ordre from TACHE
   where REFEMP = :NEW.REFEMP;
   :NEW.ORDRE_TACHE := ordre+1;
   select date_fin into datef from tache where ordre_tache = ordre
   and REFEMP = :NEW.REFEMP;
```

```
    :NEW.Date_debut := dateF+1;
END IF;
:NEW.date_fin := :NEW.Date_debut+(:NEW.NB_HEURES/24);
END
```

1.5 Solutions d'exercices supplémentaires

1.5.1 Solution de l'exercice supplémentaire 1

```
CREATE OR REPLACE TRIGGER "Resr"
BEFORE INSERT ON reservation
FOR EACH ROW
DECLARE
   Nombreplace INTEGER;
   NbCapacite INTEGER;
BEGIN
 Nombreplaces := 0 ;
 Select sum(nombre_personnes) into Nombreplaces from reservation where
 date= :NEW.date and service= :NEW.service ;

 select capacite into NbCapacite from restaurant where NumResto = :NEW.
     NumResto
if Nombreplaces + :NEW.nbPersonne > capacite then
  RAISE_APPLICATION_ERROR(-20006, 'le Restaurant est au complet');
END IF;
END;
```

1.5.2 Solution de l'exercice supplémentaire 2

1. Mettre à jour la QteStock automatiquement.

```
CREATE OR REPLACE TRIGGER "majQteStock"
BEFORE INSERT ON realisgateau
FOR EACH ROW
DECLARE q number;
BEGIN
 select qtestock into q
 from ingredient
 where numingr = :NEW.numingr;
 IF q<:NEW.qteutilise then RAISE_APPLICATION_ERROR(-20203,'stock
     inferieur');
 END IF;
 IF q>=:NEW. qteutilise then update produit set qstock=qstock-:NEW.
     qteutilise where nprod = :NEW.nprod;
 END IF;
END;
```

2. Il suffit de mettre les trois champs NumReal, NumGateau, NumIngr , comme clé primaire composée

1.5.3 Solution de l'exercice supplémentaire 3

1. Vérifier que le numéro de client d'un compte existe bien (Contrainte Foreign key).

```
CREATE or REPLACE "TRIGGER FK_CONSTRAINT"
AFTER INSERT OR UPDATE OF NUMCLI ON COMPTE
FOR EACH ROW
DECLARE
existe number(10);
gerer_excep EXCEPTION;
BEGIN
  SELECT COUNT(NUMCLI) into existe from CLIENT where
  NUMCLI = :NEW.NUMCLI;
  if existe = 0 then
  raise gerer_excep;
  end if;
  exception
  when gerer_excep then
  RAISE_APPLICATION_ERROR(-20300, 'vilolation de fk');
END;
```

2. Mettre à jour le solde du compte client après chaque opération effectuée.

```
CREATE OR REPLACE "TRIGGER UPDATE_SOLDE"
AFTER INSERT OR UPDATE OF MTOP ON OPERTATION
FOR EACH ROW
DECLARE
t_op VARCHAR2(20);
BEGIN
  t_op := :NEW.TYPEOP;
  if t_op = 'D' then
    UPDate COMPTE set
    SOLDECPT = SOLDECPT + :NEW.MTOP
    where NUMCPT = :NEW.NUMCPT;
    else
     UPDate COMPTE set
    SOLDECPT = SOLDECPT - :NEW.MTOP
    where NUMCPT = :NEW.NUMCPT;
   end if;
END;
```

3. Interdire aux clients de retirer un montant supérieur au solde du compte courant.

```
CREATE OR REPLACE TRIGGER "CHECK_MONTANT_R"
BEFORE INSERT OR UPDATE OF MTOP ON OPERATION
FOR EACH ROW
DECLARE
SoldeC  number;
BEGIN
   select soldecpt into SoldeC from COMPTE c where c.TYPECPT = '
      CCourant' and c.NUMCPT = :NEW.NUMCPT;
   if ( :NEW.typeop = 'r' and :NEW.MTOP > SoldeC ) then
      RAISE_APPLICATION_ERROR(-20300, 'solde insuffisant');
   end if;
END;
```

4. Interdire à un client d'avoir plusieurs comptes courants.

```
CREATE OR REPLACE TRIGGER "check_multiple_cc"
BEFORE INSERT OR UPDATE OF TYPECPT ON COMPTE
FOR EACH ROW
DECLARE
   nbr_cc number(20);
   check_multiple_cc exception;
BEGIN
   nbr_cc :=0;
   SELECT count(NUMCPT) INTO
   nbr_cc FROM COMPTE
   where NUMCLI = :NEW.NUMCLI and TYPECPT = 'CCOURANT';
   if :NEW.TYPECPT = 'CCOURANT' and nbr_cc >= 1 then
   raise check_multiple_cc;
   end if;
   EXCEPTION
   when check_multiple_cc then
   RAISE_APPLICATION_ERROR(-20300,'Oops ! vous ne pouvez avoir qu un
       seul ccourant');
END;
```

5. Mettre à jour automatiquement le champ NbCptE.

```
CREATE OR REPLACE TRIGGER "AUTOINC_NbCptE"
AFTER INSERT OR UPDATE OF TYPECPT ON COMPTE
FOR EACH ROW
BEGIN
   if :NEW.TYPECPT = 'CEPARGNE' then
   UPDATE CLIENT SET NbCptE = NbCptE + 1
   WHERE NUMCLI = :NEW.NUMCLI;
   end if;
END;
```

6. Modifier le déclencheur 5 pour limiter le nombre de comptes épargne à 10.

```
CREATE OR REPLACE "TRIGGER LIMIT_CE"
BEFORE INSERT ON COMPTE
FOR EACH ROW
DECLARE
   nbr_ce number(20);
   check_nbr_ce exception;
BEGIN
   nbr_ce :=0;
   SELECT NBCPTEV INTO
   nbr_ce FROM CLIENT
   where NUMCLI = :NEW.NUMCLI;
   if :NEW.TYPECPT = 'CEPARGNE' and nbr_ce >= 10 then
     raise check_nbr_ce;
   end if;
   EXCEPTION
   when check_nbr_ce then
     RAISE_APPLICATION_ERROR(-20300,'Oops le nbr de compte épargne
         est limite a 10');
END;
```

7. Interdire à un client de faire plus de 2 opérations de retrait par jour sur le même compte.

```
CREATE OR REPLACE TRIGGER LIMIT_NBR_OPERATION
BEFORE INSERT ON OPERTATION
FOR EACH ROW
DECLARE nbr_op number(20);
exc exception;
BEGIN
   nbr_op := 0;
   SELECT count(NUMOP) into nbr_op from OPERTATION
   where NUMCPT = :NEW.NUMCPT and DATEOP = :NEW.DATEOP; //(:NEW.
      DATEOP ou SYSDATE)
   if nbr_op >=2 and :NEW.TYPEOP = 'R' then
   raise exc;
   end if;
   exception
   when exc then
   RAISE_APPLICATION_ERROR(-20300, 'vous ne pouvez pas effectuer
      plus de 2 operations par jour sur le meme compte !');
END;
```

8. Les clients ayant un solde inférieur à 40 000 da ne peuvent pas bénéficier d'un emprunt.

```
CREATE OR REPLACE "TRIGGER BENIF_EMPRUNTE"
BEFORE INSERT ON EMPRUNT
FOR EACH ROW
DECLARE
   solde number(20);
BEGIN
   SELECT sum(soldecpt) INTO solde from CLIENT clt ,COMPTE cpt
   where clt.NUMCLI = cpt.NUMCLI
   and
   Cpt.Numcli = :NEW.Numcli;
   if solde<40000 then
     RAISE_APPLICATION_ERROR(-20300, 'desole ! vous ne pouvez pas
        beneficier d un emprunt');
   END IF;
END;
```

Partie sur :

Les bases Relationnelles Objets (RO)

Série TD 2

Modélisation et définition du schéma RO

Sommaire

2.1 Objectifs et Rappels

L'objectif principal de cette série d'exercices est de modéliser et créer une base de données Relationnelle Objet. Il s'agit de reprendre le modèle relationnel et de le transformer en bénéficiant des apports de l'objet, on verra entre autres comment :

- Définir de nouveaux types complexes,
- Dénormaliser les données en utilisant le principe d'imbrication de collections : tableaux et tables imbriquées,
- Utiliser les références : à la différence du relationnel, l'enregistrement dans une table objet est identifié par son OID ce qui permet de le référencer par un pointeur et d'accéder à toute la ligne référencée,

- Utiliser le principe de l'objet comme l'héritage.

2.1.1 Syntaxe LDD

2.1.1.1 Création de nouveaux types (CRAETE TYPE)

Dans le RO, les utilisateurs ont, en plus des domaines usuels de SQL tels que VarChar, Number, etc., la possibilité de définir de nouveaux types propres à leur base de données grâce au constructeur TYPE dont il existe plusieurs variantes selon le type utilisé : le type abstrait de données ou le type ensembliste (Collections imbriquées). La figure 2.1 présente les différentes formes du Constructeur TYPE et la syntaxe Oracle générale de création de chacun d'entre eux.

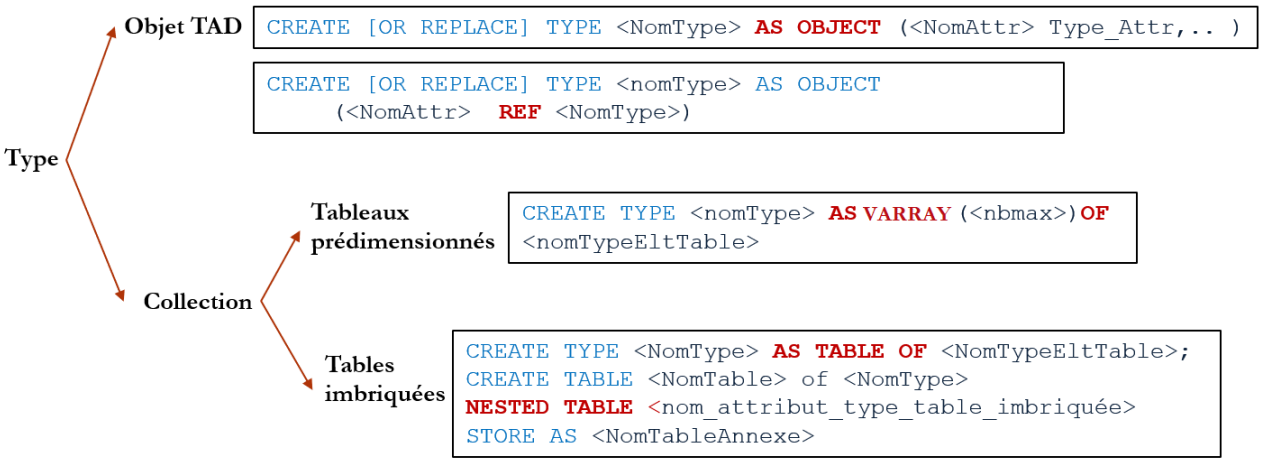

FIGURE 2.1: *Les différentes variantes (avec syntaxe de création) du constructeur "TYPE"*

a Le type abstrait de données TAD

Appelé aussi type objet il permet de définir de nouveaux types par l'utilisateur. Il peut correspondre à une structure de données partagées ; dans ce cas, il peut être utilisé dans une ou plusieurs tables et entrer dans la composition d'un ou plusieurs autres types. Il peut également décrire la structure et le comportement des objets stockés dans des tables objet-relationnelles. La syntaxe de création est définie à l'aide du mot clé AS OBJECT (figure 2.1)

- **Utilisation de pointeurs (REF)** : un attribut peut avoir comme type de définition une référence à un autre type (utilisation du mot clé REF). Cette définition par référence correspond à un objet de type pointeur sur l'objet référencé. Les références peuvent être vues comme une extension de la notion de clé étrangère, et elles constituent le principal moyen pour représenter l'intégrité référentielle dans les données objets comme les types définis par l'utilisateur.

- **Type vide** : appelé aussi incomplet est un type créé sans définition d'attributs (CREATE TYPE NomTYPE;). Ce type vide est généralement utilisé dans le cas de références croisées, il sera complété par la suite.
- **Héritage entre types** : l'héritage est défini en utilisant le mot clé UNDER. Le type qui a servi à l'héritage doit contenir à la fin de sa définition le mot clé NOT FINAL;

```
CREATE [or REPLACE] TYPE NomType1 AS OBJECT (..) NOT FINAL;
CREATE [or REPLACE] TYPE UNDER NomType2 (..);
```

b Le type ensembliste (Collections)

- **Tableaux** : c'est une collection avec un nombre de données ordonnées et limitées. il est défini avec le mot clé AS VARRAY (voir figure 2.1)
- **La table imbriquée** : à la différence du tableau c'est une collection non ordonnée et non limitée en nombre d'éléments. Quand on crée la table imbriquée (NESTED TABLE) le système Oracle crée physiquement une table annexe dans laquelle il stockera les enregistrements. Cette table doit être nommée dans la définition de la requête (voir syntaxe figure 2.1). A noter que cette table physique ne sera jamais utilisée dans les requêtes LMD.

2.1.1.2 Modification de types (ALTER TYPE)

La syntaxe simplifiée pour modifier les attributs du type objet est la suivante :

```
ALTER TYPE NomType
ADD|MODIFY attribute
{NmAttr [Type_attr]|(NommAttr type_attr [, NomAttr type_attri]...)}
|DROP attribute {NomAttr[, NomAttr]..)}
}
CASCADE [[NOT] including table data]
```

La clause CASCADE permet de propager les changements aux types et tables dépendantes (qui utilisent le type modifié). L'instruction de modification est annulée si une erreur est détectée.
Avec l'option INCLUDING TABLE DATA, les modifications sont répercutées sur les objets; Sinon avec l'option NOT les modifications sont prises au niveau du méta-schéma (dictionnaire de données), mais elles ne s'impliquent pas sur les données physiquement.

2.1.1.3 Suppression de types (DROP TYPE)

```
DROP TYPE NomType [FORCE]
```

L'option FORCE permet, quand il y a utilisation de référencement de types, d'éviter le problème d'incohérences de pointeurs (un pointeur qui référence le type supprimé).

2.1.2 Modélisation et passage du relationnel vers RO

La possibilité de création de nouveaux types complexes dans le relationnel objet permet de revoir la modélisation du relationnel [3] en dénormalisant [4] la première forme normale 1NF. En effet, les attributs peuvent être d'un type complexe (non atomique ou simple) ou encore multivalués. Il s'agit alors d'imbriquer la relation fils dans un attribut de la relation père en utilisant les collections : un tableau si le nombre d'enregistrement (tuples) est fixé ou, dans le cas contraire, une table imbriquée.

- Cas d'une association 1-n : Si le n est limité, par exemple : une personne peut avoir au maximum 3 prénoms, 5 adresses, 4 numéros de téléphone. On utilisera un attribut de type tableau (VARRAY) dans la définition de la table père.
  Si le n n'est pas limité, alors il faudra soit utiliser une table imbriquée ou bien utiliser la référence.
- Cas d'une association n-n : soit la relation L entre les deux relations A et B
  - Utilisation des références (pas d'imbrications) : comme dans le relationnel, une table association est rajoutée avec des références. Par exemple, si on a une association L entre deux tables A et B, trois tables A, B et L seront créées avec les attributs de type REF dans la table L. Cette solution est moins optimale que l'utilisation de l'imbrication car elle nécessite plusieurs accès pour retourner le résultat d'une requête de recherche.
  - Utilisation des imbrications : dans ce cas, l'imbrication ne doit concerner que l'association. il faudra imbriquer L (Nested TABLE) dans l'une des deux tables soit A soit B [4]. La table choisie pour l'imbrication est selon l'utilisation de l'application à développer. La table la plus utilisée sera choisie pour l'imbrication

2.2 Exercices

2.2.1 Exercice 1 : Du modèle RO vers la syntaxe SQL3

Soit le schéma conceptuel de données de la base de données RO (figure 2.2) relatif aux déplacements des véhicules (camions) d'une entreprise entre le dépôt (stock des matériaux comme : gravier, ciment, etc.) et les différents chantiers de construction.
Donner les différentes commandes (syntaxe SQL3) de création de ce schéma.

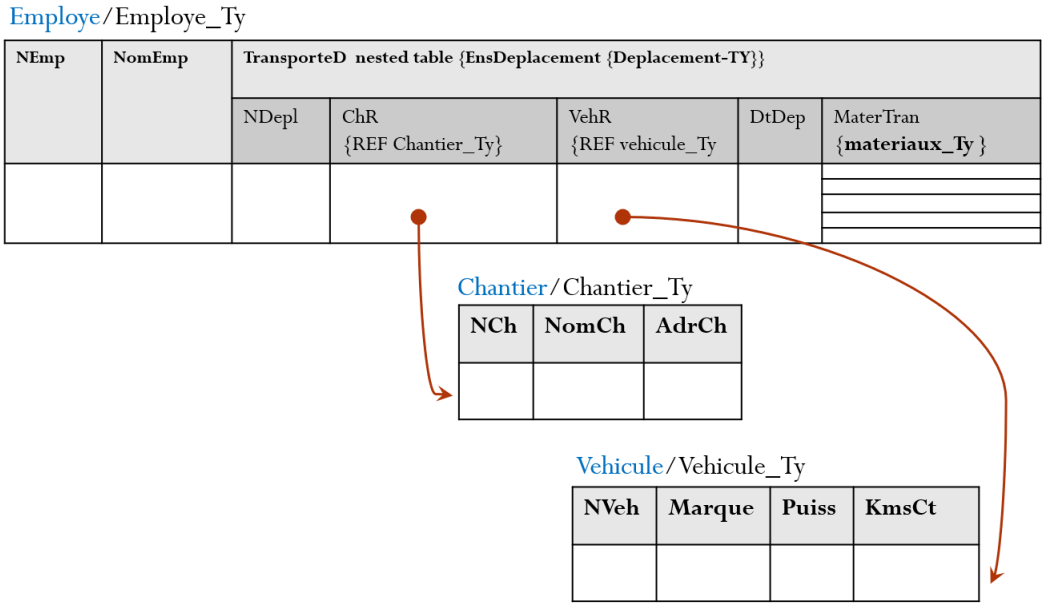

FIGURE 2.2: *Schéma conceptuel RO de la base 'Déplacements_Chantier'*

2.2.2 Exercice 2 : Transformation d'un schéma relationnel vers RO

Soit le diagramme UML (2.3) relatif à la gestion des conducteurs de voitures et de leurs propriétaires.

Le propriétaire est la personne qui possède la voiture, elle n'a pas forcément un permis.

Les conducteurs sont des personnes qui possèdent un permis et sont autorisés à conduire une voiture.

Le modèle relationnel logique de données correspondant au schéma UML (figure 2.3) est comme suit :

Personne(NumP, Nom, Prénom1, Prénom2, Prénom3) ;

Conducteur(NumP, Nom, Prénom1, Prénom2, Prénom3, NumPermis ,DatePermis) ;

Voiture(immat, TypeV,Nbportes,NumP*) ;

Conduire(LeConducteur* ,LaVoiture*) ;

Donner les différentes conceptions de données (avec script SQL3 de création du schéma) correspondantes à la transformation de ce schéma en Relationnel Objet ; équivalentes aux trois scénarios suivants :

1. La conception du modèle RO est la plus proche du modèle relationnel ;
2. En utilisant les références, la solution proposée doit être centrée sur la relation "Voitures". La solution doit comporter trois tables : "Personnes", "conducteurs" et "Voitures". Donner une solution équivalente, en favorisant la table "Conducteurs" (solution centrée sur les conduc-

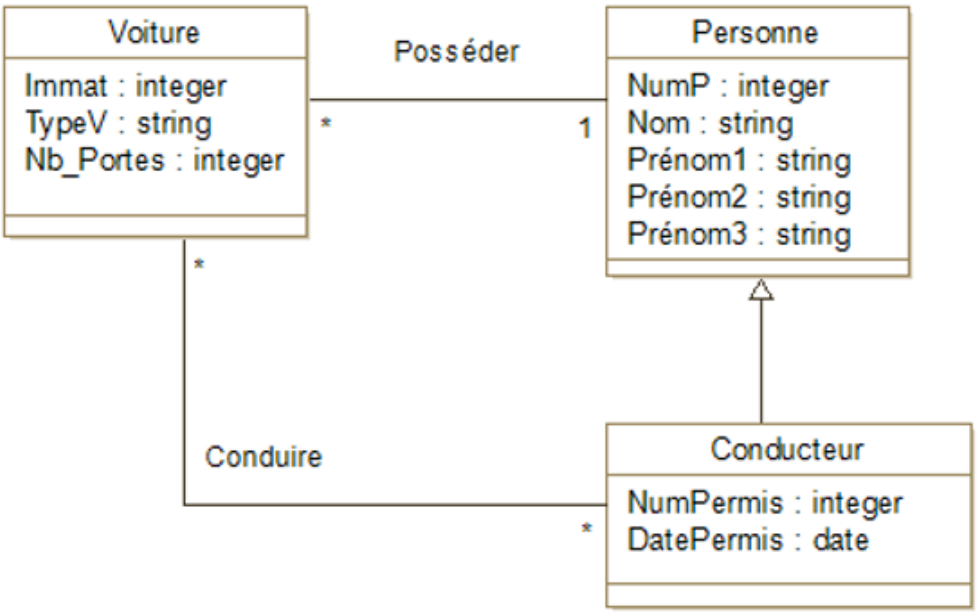

FIGURE ***2.3:*** *Schéma UML de la base "Conducteurs_Voitures"*

teurs) ;

3. Sans utilisation des références. La solution doit comporter qu'une seule table "Voitures".

2.3 Solutions d'exercices

2.3.1 Solution de l'exercice 1

La Syntaxe SQL de création de la base correspondante au schéma conceptuel RO de la base 'Déplacements_Chantiers' (figure 2.2) est donnée au listing 2.1.

Le schéma contient trois types abstraits de données qui sont : *Employe_Ty*, *Chantier_Ty* et *Vehicule_Ty*.

La syntaxe de création de ces trois types se fait par l'utilisation de la clause **AS OBJECT** de la commande **CREATE TYPE** (voir ligne 1,6 et 14 du listing 2.1).

La relation transporter les matériaux à un chantier avec un véhicule est matérialisée par une table imbriquée (clause **NESTED TABLE** de la commande **CREATE TABLE** : voir ligne 30 du listing 2.1).

Le chantier et le véhicule sont sous forme de références (Ligne 16 et 17) vers types *Chantier_TY* et *Vehicule_TY)*. L'utilisation de la référence exige la création de la table à partir du type référencé d'où la création des deux tables *vehicule* et *chantier* (lignes 28, 29).

D'après le schéma, un véhicule ne peut pas transporter plus de 5 matériaux en même temps car le

champ *materiaux_TY* est schématisé par un tableau de 5 éléments. Ceci est défini par la clause **AS VARRAY (5)** de la commande **CREATE TYPE** (ligne 12).

Listing 2.1: *Commandes SQL3 de création du schéma : DéplacementsChantiers*

```
1  CREATE TYPE Chantier_Ty AS OBJECT (
2  NCh number,
3  NomCh varchar(20),
4  AdrCh varchar (50)
5  );
6  CREATE TYPE Vehicule_Ty AS OBJECT (
7   NVeh number,
8   Marque number,
9   Puiss number,
10  KmsCt number
11 );
12 CREATE TYPE  materiaux_Ty AS VARRAY (5) OF varchar (30);
13
14 CREATE TYPE Deplacement_Ty AS OBJECT
15 (NDepl number,
16  ChR REF Chantier_Ty
17  VehR REF Vehicule_Ty,
18  DtDep date,
19  MaterTran materiaux_Ty
20 );
21 CREATE TYPE  EnsDeplacement AS TABLE OF Deplacement_Ty;
22
23 CREATE TYPE Employe_Ty AS OBJECT (
24 NEmp number,
25 NomEmp varchar(20),
26 TransporteD EnsDeplacement
27 );
28 CREATE TABLE vehicule OF  Vehicule_ty;
29 CREATE TABLE chantier OF Chantier_Ty;
30 CREATE TABLE Employe OF Employe_Ty NESTED TABLE TransportD STORE AS
       TTrans;
```

2.3.2 Solution de l'exercice 2

1. **Solution proche du modèle relationnel** :

   Nous aurons exactement 4 tables comme dans le modèle relationnel.

   Une représentation graphique du schéma Relationnel Objet avec les types et tables est donnée dans la figure 2.4.

   Les références sont représentées par des flèches. Nous rappelons que l'utilisation d'une référence implique la création de la table à partir du type référencé.

   Un exemple d'un jeu de données est présenté dans la figure2.5.

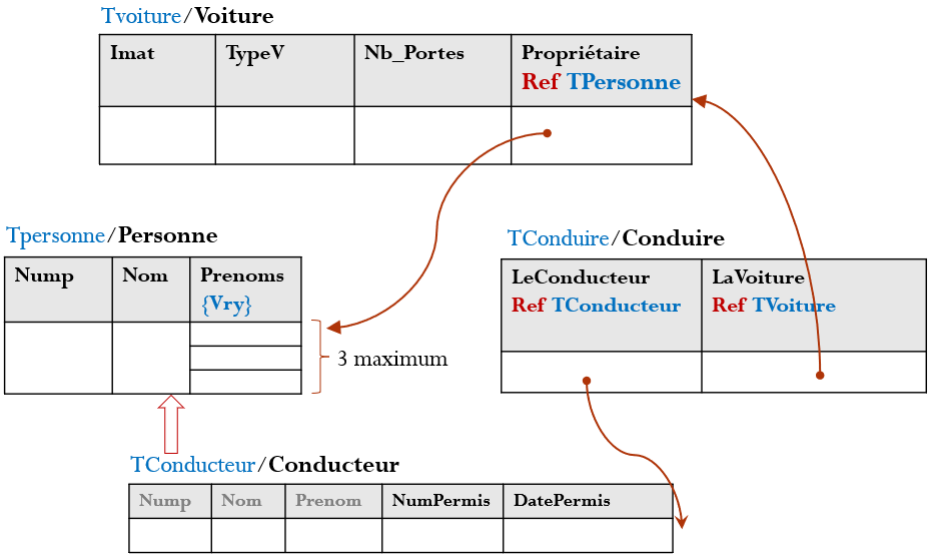

FIGURE 2.4: *Transformation 1 en RO de la base "Conducteurs_Voitures"*

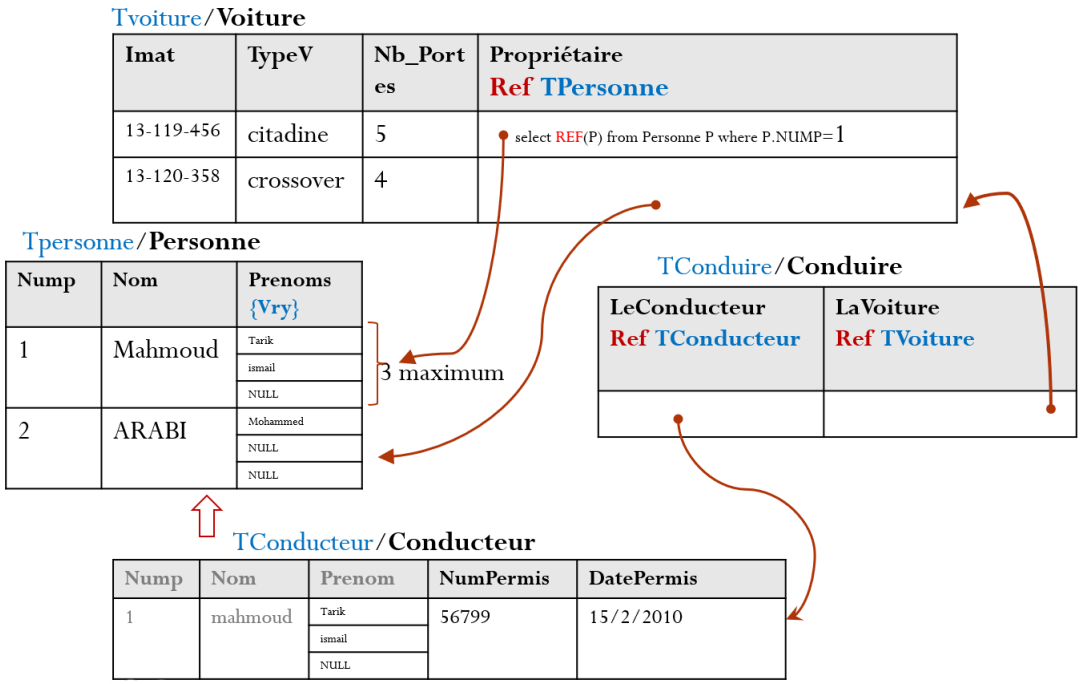

FIGURE 2.5: *Exemple d'un jeu de données de la Transformation 1 en RO de la base "Conducteurs_Voitures"*

La syntaxe SQL 3 de la création du schéma RO de cette solution (voir figure 2.5) est donnée dans le listing 2.2

Listing 2.2: *Code SQL3 du schéma de la transformation 1 : Schéma RO proche du schéma R*

```
CREATE TYPE TListePrenoms AS VARRAY(3) OF varchar(15);

CREATE TYPE TPersonne  AS OBJECT (
Nump number,
nom varchar(20),
prenom TListePrenoms
) NOT FINAL;

CREATE TYPE TConducteur UNDER  Tpersonne (
NumPermis number,
datePermis date);

CREATE TYPE TVoiture AS OBJECT (
   Immat varchar(20),
   TypeV varchar(20),
   Nb_Portes number,
   Proprietaire REF TPersonne
);

CREATE TYPE TConduire AS OBJECT (
LeConducteur REF Tconducteur,
LAVoiture REF TVoiture
)

CREATE TABLE Personne OF TPersonne ;
CREATE TABLE Conducteur OF Tconducteur ;
CREATE TABLE Voiture OF TVoiture;
CREATE TABLE Conduire OF TConduire;
```

2. **Solution centrée "Voiture" :**

   Dans ce cas, la table "Voitures" doit être créée. La table "Voitures" utilise des références vers les tables "Personnes" et "Conducteurs" pour matérialiser les relations : "posséder" et "conduire".

   La figure 2.6 présente le schéma avec les types abstraits de données et les tables correspondantes.

   Nous remarquons que cette solution est optimisée par rapport à la solution précédente (voir figure2.4). En effet, elle ne contient que trois tables, la relation conduire est représentée sous forme d'une table imbriquée de références.

   La syntaxe SQL3 de la création du schéma RO de la solution de cette transformation est donnée dans le listing 2.3

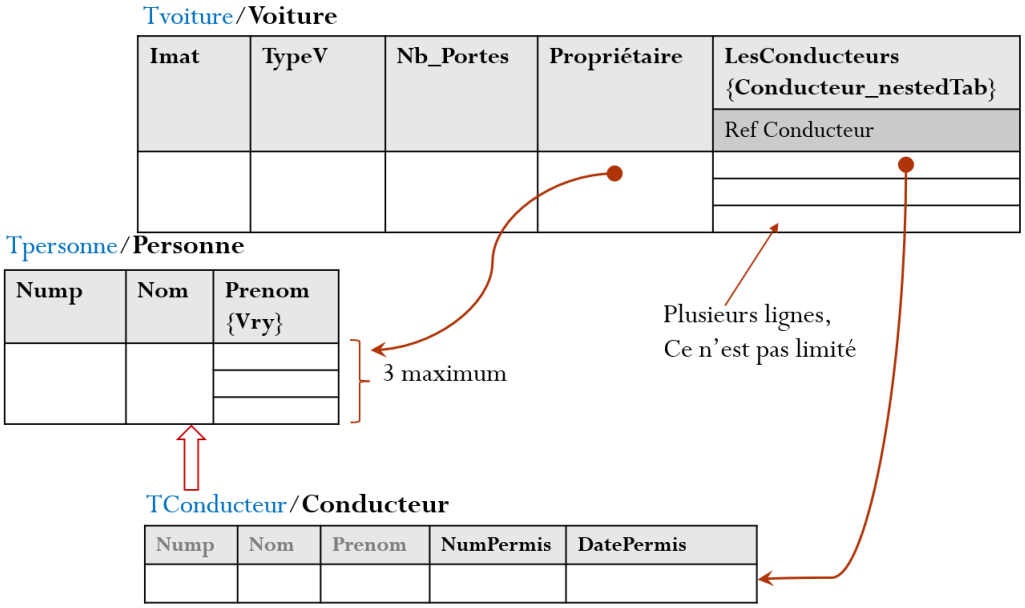

FIGURE 2.6: *Transformation 2 en RO de la base "Conducteurs_Voitures" centrée Voitures avec références*

Listing 2.3: *Code SQL3 du schéma de la transformation 2 (figure 2.6)*

```
CREATE TYPE TListePrenoms AS VARRAY(3) OF varchar(15);

CREATE TYPE TPersonne  AS OBJECT (
Nump number,
nom varchar(20),
prenom TListePrenoms
) NOT FINAL;

CREATE TYPE TConducteur UNDER  Tpersonne (
NumPermis number,
datePermis date) ;

CREATE TYPE TListeConducteurs AS TABLE  OF REF  TConducteur;

CREATE TYPE TVoiture AS OBJECT (
   Immat varchar(20),
   TypeV varchar(20),
   Nb_Portes number,
   Proprietaire REF TPersonne,
   LesConducteurs TListeConducteurs) ;

CREATE TABLE Personne OF TPersonne ;
CREATE TABLE Conducteur OF Tconducteur ;
CREATE TABLE Voiture OF TVoiture NESTED TABLE LesConducteurs STORE
    AS TabConducteurs;
```

Notons qu'une transformation similaire à cette solution peut être proposée mais centrée sur la table "conducteurs". Dans ce cas, la table "Voitures" sera une table imbriquée de références dans la table "conducteurs". Cette solution est présentée dans la figure 2.7

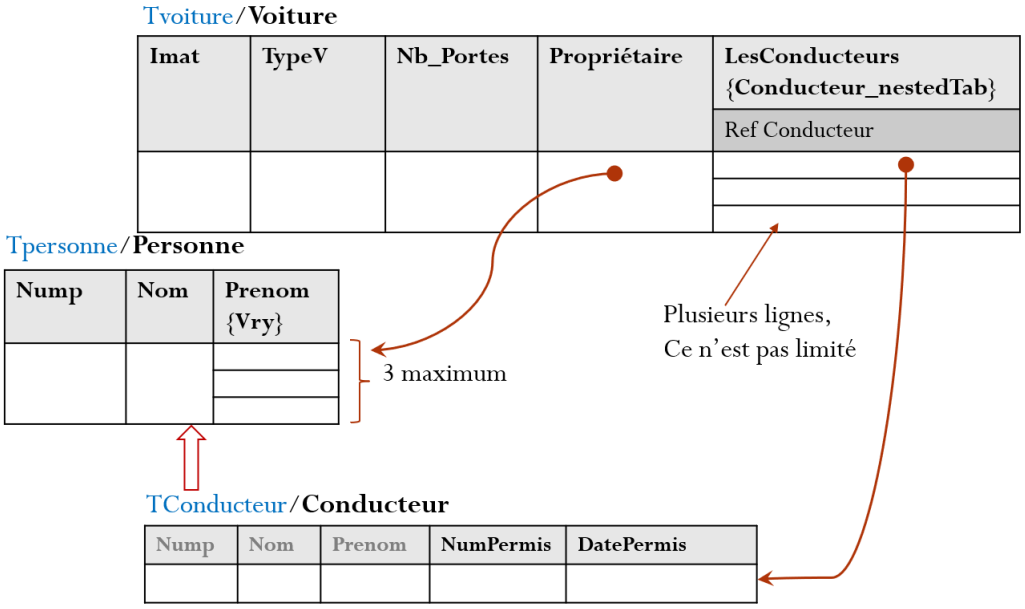

FIGURE 2.7: *Transformation 2 en RO de la base "Conducteurs_Voitures" centrée Conducteurs*

3. **La transformation 3 : sans références** :

   C'est une solution sans l'utilisation de références. Dans ce cas, toutes les informations seront incluses dans la table "Voitures". De ce fait, une seule table (Voitures) sera créée dans laquelle toutes les autres informations des autres tables (Propriétaire et Conducteurs) seront imbriquées (Voir figure 2.8).

   La syntaxe SQL 3 de la création du schéma RO de la transformation 3 (figure 2.8) est présentée dans le listing 2.4

Listing 2.4: *Code SQL3 du schéma de la transformation 3 : solution sans références orientée "Voitures"*

```
CREATE TYPE TListePrenoms AS VARRAY(3) OF varchar(15);

CREATE TYPE TPersonne  AS OBJECT (
Nump number,
nom varchar(20),
prenom TListePrenoms
)NOT FINAL;

CREATE TYPE TConducteur UNDER  Tpersonne (
NumPermis number,
datePermis date);

CREATE TYPE TListeConducteurs AS TABLE  OF  TConducteur;

CREATE TYPE TVoiture AS OBJECT (
   Immat varchar(20),
   TypeV varchar(20),
   Nb_Portes number,
   Proprietaire  TPersonne,
```

```
    LesConducteurs TListeConducteurs);

CREATE TABLE Voiture OF TVoiture NESTED TABLE LesConducteurs STORE
    AS TabConducteurs;
```

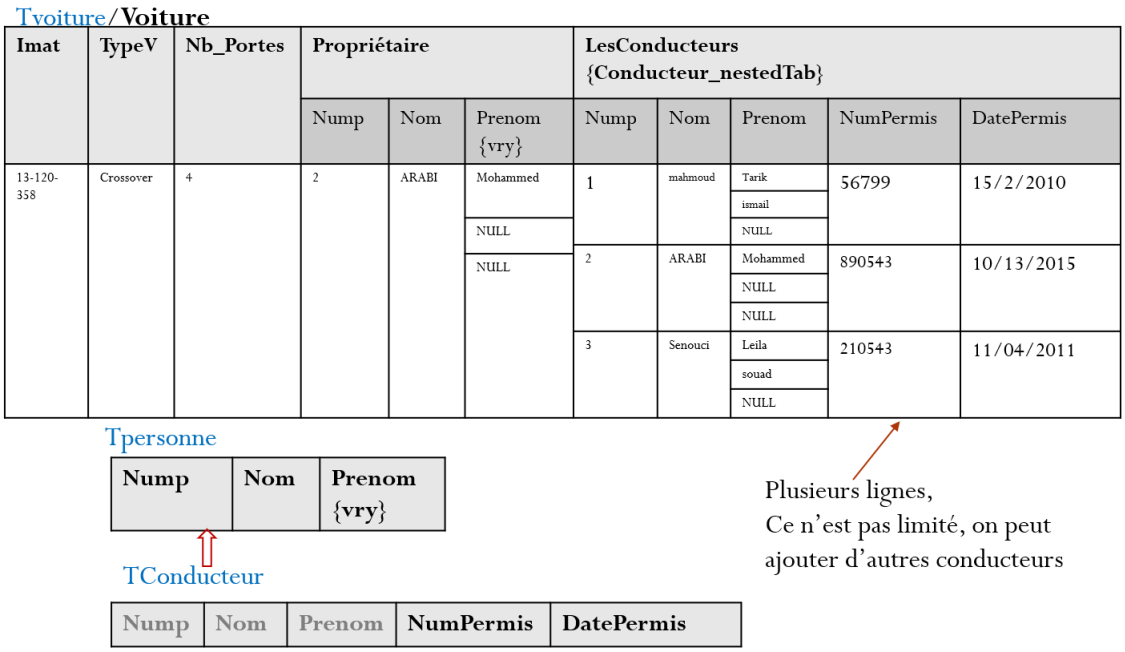

FIGURE 2.8: *Transformation 3 en RO de la base "Conducteurs_Voitures" (Solution sans références)*

Série TD 3

Manipulation des données du modèle RO

Sommaire

3.1 Objectifs et Rappels

Les exercices de cette série traitent de l'aspect LMD (Langage de Modélisation de Données) d'une base de données relationnelle objet. Nous verrons comment interroger une base de données dont le schéma de définition a été créé en SQL3 d'oracle (voir série de TD 2) où de nouveaux types complexes sont rajoutés.

Dans ce qui suit, nous donnerons la syntaxe générale pour manipuler les données de ces types complexes.

3.1.1 Type abstrait de données TAD

- **Insertion dans une table objet**

  Il est obligatoire d'utiliser le constructeur de chaque attribut de type objet.

```
INSERT INTO <NomTable> VALUES (Constructeur(att1 ,..))
```

  Dans le cas où l'attribut est de Type REF, il faudra dans ce cas, associer un pointeur en sélectionnant la référence de la ligne (objet) référencée. Cette ligne doit être obligatoirement insérée avant d'être référencée.

```
INSERT INTO <NomTable> VALUES (..., (SELECT REF(t) FROM TableRéfé
    rencée t WHERE t.NUM = ..), ..)
```

- **Modification**

  – Modification d'un attribut : utilisation de la notation pointée

```
UPDATE NomTable t
set t.NomAttr.attr= NouvelleValeur
where t.NomAttr1 = Valeur;
```

  – Affectation ou modification d'une valeur (pointeur) à un attribut de type REF

```
UPDATE NomTable t
set t.NomAttr.attr= (SELECT REF(t) FROM TableRéférencée t
   WHERE t.NUM = ..)
where t.NomAttr1 = Valeur;
```

  – Modification de l'objet complet : utilisation du constructeur. l'utilisation du constructeur oblige à renseigner tous les champs. Attention : si aucun champ n'est renseigné l'objet sera vide.

```
UPDATE NomTable t
set t.NomAttrComplexe= Constructeur(NouvelleValeurAttr1 ,
    NouvelleValeurAttr2 ,..)
where t.NomAttr = Valeur;
```

- **Suppression**

  La suppression de l'objet revient à le mettre à NULL

```
UPDATE NomTable t
set t.NomAttrComplexe= NULL
where t.NomAttr1 = Valeur;
```

- **Sélection**

  Pour la sélection, on utilise la notation pointée pour accéder à l'objet manipulé comme attribut d'un objet principal.

3.1.2 Les collections

3.1.2.1 Les tableaux

- **Insertion**

  elle se fait de la même manière que le type TAD. il faudra englober avec le constructeur.

- **Modification et suppression**

  Il n'existe pas de commandes prédéfinies pour la suppression ou la modification d'un enregistrement dans un VARRAR. Pour cela, il faudra écrire un programme en PL/SQL

- **Sélection**

  Il n'est pas possible d'accéder directement à un élément(enregistrement) d'un VARRAY. Un SELECT restitue le tableau dans son intégralité. Pour naviguer dans des collections, il faut désimbriquer (ou aplatir) la collection. L'expression TABLE permet d'interroger une collection dans la clause FROM comme une table.

```
SELECT ... FROM NomTable t , TABLE (t.attribut-multivalué) v ...
```

3.1.2.2 Les tables imbriquées

- **Insertion**

```
INSERT INTO TABLE( SELECT AttrTypeTableOf FROM
NomTable ... ) VALUES ( ... ) ;
```

  Le mot clé Table peut être remplacé par THE. La commande permet de stocker un enregistrement dans la table imbriquée désignée par THE ou Table. Le SELECT doit retourner un seul objet, ce qui permet de sélectionner la table imbriquée associée. Ces commandes ne peuvent être utilisées que si la table imbriquée a été initialisée Sinon il faudra l'initialiser et/ou insérer avec un UPDATE

- **Modification**

```
UPDATE TABLE( SELECT AttrTypeTableOf FROM NomTable ... ) t SET t.
    attr =.... ;
```

  Pour initialiser la table imbriquée, il faudra modifier la table qui contient le champ imbriqué, en affectant à l'attribut le constructeur vide.

```
UPDATE NomTable
SET AttrTypeTableOf = Constructeur();
```

- **Suppression**

  - Pour supprimer une ligne de la table imbriquée

```
DELETE from TABLE(ELECT AttrTypeTableOf FROM NomTable ... ) t WHERE
    t.attr = ...;
```

  - Pour supprimer la table imbriquée, il suffit de modifier la table qui contient le champ imbriqué, en affectant à l'attribut NULL.

- **Sélection**

  Pour interroger les données d'une table imbriquée, on utilise TABLE comme pour l'interrogation des VARRAY.

3.2 Exercices

3.2.1 Exercice 1

Reprenons le schéma de la base de l'exercice 1 de la série de TD 2 (voir section 2.2.1) et son code de création (Listing 2.1). Donnez les requêtes suivantes :

1.
   - Insérer les camions :(1,'iveco',4,195850) et (2,'renault',7,85630) et un chantier (1,'laboratoire','Universite','Tlemcen')
   - Insérer l'employé 1, 'Mohammed' avec un transport NULL
   - Insérer l'employé 2, 'Tariq' qui a effectué hier deux déplacements entre le dépôt et le chantier 1. Le premier avec le véhicule pour transporter du Gravier, le deuxième n'est pas encore renseigné.
2. L'employé 'Mohammed' a effectué aujourd'hui un transport des deux matériaux : sable et ciment pour le chantier 1 avec le véhicule 2, rajouter cette information dans la base.
3. Formuler la requête qui donne le nom des employés et le nom des chantiers dans lesquels ils se sont rendus à la date '10/05/2015'.
4. Spécifier la requête qui restitue pour chaque employé le nombre de déplacements effectués.

3.2.2 Exercice 2

Soit le schéma SQL3 (3.1) d'une base de données RO qui gère les rencontres sportifs entre les équipes. Réaliser les commandes SQL3 suivantes :

1. Créer les tables 'LesJoueurs' et 'LesEquipes'.

   - On suppose que la table joueur contient les joueurs (1, 'ABDELLAOUI', 28 ans) , (2, 'AISSAOUI', 19 ans)

2. Insérer l'équipe 'WAT' qui contient pour l'instant les deux joueurs 1 et 2. Le joueur 2 est aussi le capitaine de cette équipe, mettre les autres renseignements à NULL.
3. L'équipe de 'JSM Béjaïa', qui est déjà insérée et qui contient 10 joueurs, a recruté le joueur 'Aissaoui' dans son équipe, ce dernier a été transféré de l'équipe WAT. Faites ces mises à jour sur la base.
4. Ajouter une rencontre sportive à l'équipe WAT qui est prévue le 24/3/2016 avec l'équipe 'JSM Béjaïa' au state 'Akid Lotfi'
5. Donner l'équipe qui a comme capitaine le joueur 'AISSAOUI'.
6. Donner l'équipe qui a disputé le plus grand nombre de rencontres sportives

Listing 3.1: *Schéma SQL3 de la base 'Rencontres sportives'*

```
CREATE TYPE Equipe; /*type incomplet*/

CREATE TYPE Joueur AS OBJECT(
  num number,
  nom varchar2(30),
  age number(2)
  );
CREATE TYPE Ens_Joueurs AS TABLE OF REF Joueur;

CREATE TYPE rencontre AS OBJECT (
  equi  REF equipe ,
  datem date,
  lieu varchar(20)
  );
CREATE TYPE Ens_rencontres  AS TABLE OF rencontre;

CREATE TYPE Equipe AS OBJECT (
  nom varchar2(30),
  joueurs Ens_Joueurs,
  capitaine REF Joueur,
  lesRencontre Ens_rencontres
  );
```

3.3 Solutions d'exercices

3.3.1 Solution de l'exercice 1

1. Les insertions :
   - L'insertion des deux camions et du chantier sont des insertions avec des types simples prédéfinis comme dans le relationnel.

```
INSERT INTO Vehicule VALUES(1,'iveco',4,195850)
INSERT INTO Vehicule VALUES(2,'renault',7,85630)
```

```
INSERT INTO Chantier VALUES(1,'laboratoire','Universite','
    Tlemcen')
```

- Insérer l'employé 1, 'Mohammed' avec un transport NULL

```
INSERT INTO employe VALUES(1,'mohammed',NULL);
```

- Insérer l'employé 2, 'Tariq' qui a effectué hier deux déplacements entre le dépôt et le chantier 1. Le premier avec le véhicule pour transporter du Gravier, le deuxième n'est pas encore renseigné.

```
INSERT INTO employe VALUES(2,'Tariq', EnsDeplacement(
    Deplacement_Ty(1, (SELECT REF(c) FROM chantier c WHERE NCh
    =1),   (SELECT REF(v) FROM vehicule V WHERE NVeh=1),  '
    2/5/2020',  Materiaux_Ty('Gravier')), Deplacement_Ty()));
```

2. L'employé 'Mohammed' a effectué un transport du sable et du ciment aujourd'hui pour le chantier 1 avec le vehicule2, rajouter cette information dans la base.

   La table imbriquée ne contient aucun enregistrement pour l'instant, elle n'est même pas encore créée donc il faudra d'abord exécuter une modification sur la colonne 'TransporteD' de la table 'employé' pour initialiser la table imbriquée. On aurait pu, à ce niveau, ajouter la première insertion dans la table imbriquée au moment de son initialisation.

```
UPDATE table Employe e
SET e.TransporteD = EnsDeplacement ()
WHERE e.NEmp= 2;

INSERT INTO TABLE  ( SELECT TransporteD FROM Employer  WHERE e.NEmp
    = 2) VALUES
(Deplacement_Ty(1, (SELECT REF(c) FROM chantier c  WHERE Ch=1), (
    SELECT REF(v)
  FROM vehicule V WHERE NVeh=2),'03/05/2020', Materiaux_Ty('sable',
      'ciment'))));
```

3. Formuler la requête qui donne le nom des employés et le nom des chantiers dans lesquels ils se sont rendus à la date '10/05/2015'

```
SELECT t.NChr,  e.nomEmpl, t.ChR.nomCh,
FROM employe e, TABLE(e.TransporteD) t
WHERE t.DtDep = '10/15/2015'
```

4. Spécifier la requête qui restitue pour chaque employé le nombre de déplacements effectués.

```
SELECT e.Nemp, e.NomEmp, COUNT(d.nDepl)
FROM employe e, TABLE(e.transporte) d
GROUP BY e.Nemp, e.NomEmp
```

3.3.2 Solution de l'exercice 2

1. Créer les tables LesJoueurs et LesEquipes

```
CREATE TABLE lesjoueurs OF joueur;
CREATE TABLE lesequipes OF equipe nested table joueurs,  nested
    table lesRencontres
```

2. Insérer l'équipe 'WAT' qui contient pour l'instant les deux joueurs 1, et 2. Le deuxième joueur est aussi le capitaine de cette équipe, mettre les autres renseignements à NULL.

```
INSERT INTO lesEquipes VALUES ('WAT',Ens_joueurs( (select REF(j)
    FROM lesjoueurs WHERE j.num=1), SELECT REF(j) FROM Lesjoueurs
    WHERE j.num=2)), SELECT REF(j) FROM joueur WHERE j.num=2),NULL)
    ;
```

3. l'équipe de 'JSM Béjaïa', qui est déjà insérée et qui contient 10 joueurs, a recruté le joueur 'Aissaoui' dans son équipe, ce dernier a été transféré de l'équipe WAT. Faites ces mises à jour sur la base.

```
DELETE FROM TABLE (SELECT joueurs FROM Equipe WHERE nom ='WAT') t
    WHERE  t.num=2;
UPDATE equipe e SET capitaine=null WHERE  e.num=2;
INSERT INTO TABLE (SELECT joueurs FROM Equipe WHERE nom =''JSM
    Bejaia) VALUES (select REF(j) FROM Lesjoueurs WHERE j.num=2);
```

4. Ajouter une rencontre sportive à l'équipe WAT qui est prévue le 24/3/2016 avec l'équipe 'JSM Béjaïa' au state 'Akid Lotfi'

```
UPDATE equipe  SET lesRencontre=EnsRencontre(SELECT REF(e) FROM
    equipe e WHERE e.nom = 'JsmBejaya', 24/03/2016, 'Akid Lotfi')
```

5. Donner l'équipe qui a comme capitaine le joueur AISSAOUI.

```
SELECT  e.nom
From lesequipes e
WHERE e.capitaine.nom='aissaoui'
```

6. Donner l'équipe qui a disputé le plus grand nombre de rencontres sportives

```
SELECT  e.nom, MAX(count(r))
FROM lesequipes e,  TABLE(e.rencontre) r
GROUP BY e.nom;
```

Partie sur :

Les Bases NoSQL

Série TD 4

Modélisation des bases NoSQL (Mongodb)

Sommaire

4.1 Objectifs et Rappels

L'objectif principal de cette série d'exercices est de se familiariser avec la nouvelle façon de concevoir et modéliser les bases de données NoSQL. Ces bases sont très différentes des bases relationnelles où l'aspect normalisation n'est plus respecté, ce qui offre plusieurs alternatives de modélisation. Il s'agit de voir à travers les exercices proposés comment dénormaliser une base relationnelle et structurer les données selon les familles de schéma NoSQL [5] (voir image 4.1). Nous nous intéresserons plus particulièrement à MongoDB qui est, actuellement, la base la plus utilisée dans la famille des bases orientées documents.

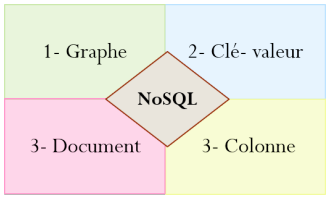

FIGURE *4.1:* *Familles de schéma de données NoSQL*

Il existe plusieurs différences entre le relationnel et NoSQL [5]. Par exemple, un champ avec une valeur NULL dans le relationnel, n'est simplement pas représenté dans le NoSQL. Les bases NoSQL sont dites schemaless à l'instar du relationnel qui se base sur un schéma rigide qui est défini au départ et qui est le même pour tous les enregistrements de la table. Les principales différences sont présentées dans le tableau 4.1

Relationnel	**NoSQL**
Schéma défini au départ	Schémaless
Structure rigide	Structure dynamique
Gestion des valeurs Null	Pas de valeurs Null
Scalabilité Verticale	Scalabilité Horizontale sans limites
Transactions ACID	Transactions BASE
Cohérence des données	Disponibilité et la Performance
Mauvaise gestion de gros volumes de données	Traite un gros volume de données
Langage SQL	Différents Langages selon le modèle utilisé

Tableau 4.1: *SGBD Relationel Vs. NoSQL.*

4.1.1 NoSQL et les différents schémas de SGBD

Dans ce qui suit, nous donnerons un rappel de quelques caractéristiques des 4 schémas de données (voir figure 4.1) des bases NoSQL. L'exercice 1 (section 4.2.1) se chargera de détailler la transformation d'un modèle relationnel vers les 4 Types avec un exemple d'un SGBD pour chacun des 4 types de schémas. Il est à noter qu'il existe une autre classification [5] (catégorisation des SGBD) par performance et non par schéma de données.

4.1.1.1 Orienté graphe

Les SGBD de schéma orienté graphe se basent sur la théorie des graphes, ils se caractérisent par :

- Des nœuds qui ont chacun leur propre structure ;
- Des relations entre les nœuds ;
- Des propriétés (de nœuds ou de relations).

4.1.1.2 Orienté clé - valeur

Les données sont sous forme d'une grosse HashMap : un simple tableau associatif unidimensionnel avec des millions voire des milliards d'entrées.
Une entrée est un couple Clé+Valeur :

- Pas de schéma pour la valeur (chaine, objet, entier, binaire. . .)
- Pas d'expressivité d'interrogation (pré/post traitement pour manipuler concrètement les données)

4.1.1.3 Orienté colonne

Les informations sont stockées colonne par colonne. Elles se rapprochent le plus des bases relationnelles (principe de tables) mais beaucoup plus souples.

- Les colonnes sont dynamiques : Les lignes peuvent avoir des colonnes différentes avec un nombre diffèrent
- Pas de champ «NULL» contrairement à une table relationnelle (pas de colonne inutile dans une ligne)
- L'historisation des données se fait à la valeur et non pas à la ligne comme dans les SGBDR

4.1.1.4 Orienté document

- Basé sur le modèle clé-valeur où la valeur = document
- Le document a une structure arborescente : les données sont semi structurées (JSON ou XML)
- Les documents sont structurés mais aucune définition de structure préalable n'est nécessaire.
- Un document est composé de clés/valeurs
- Le type de données peut être simple (Int, String, Date) ou sous forme d'une liste de données.
- Les données peuvent être imbriquées (schéma arborescent)

4.1.2 Caractéristiques d'une base Mongodb

MongoDB est un SGBD open source qui a été développé en C++ par la compagnie 10gen en 2007, et rebaptisé MongoDB, Inc. en 2013 [6]. C'est une base de données orientée document. En effet, le stockage/récupération des données est sous forme de documents au format BSON (Binary JSON).

Les types de données BSON data types disponibles dans MongoDB sont : double, string, object (un autre document), array, binary data, ObjectID, Boolean, date, null, regular expression, JavaScript, and integer (32-bit and 64-bit).

Les caractéristiques principales de Mongodb et son analogie avec le relationnel sont :

- La base de données est une collection de collections.
- La collection est l'équivalent d'une table dans le relationnel. La collection contient un ensemble de documents.
- Un document est l'équivalent d'une ligne/tuple ou enregistrement dans le relationnel. Tous les documents d'une collection ne sont pas nécessairement identiques. Les mêmes champs peuvent avoir un ordre différent dans les documents ou encore un champ peut exister dans un document et pas dans un autre, ce qui est inconcevable dans les bases relationnelles.
- Chaque document est identifié de manière unique dans une collection. La clé d'un document porte un nom fixe :**_id**. Elle peut être fournie par l'utilisateur ou générée automatiquement par le SGBD (**ObjectId**). C'est l'équivalent de la clé primaire dans le relationnel.
- Les documents BSON ont une taille de stockage limitée à 16 Mo. Pour les documents de plus grande taille mongoDb utilise GridFS.

4.1.3 Templates de modélisation d'une base Mongodb

La normalisation [1] est un principe fondamental dans les bases relationnelles, elle permet d'assurer la non redondance des données en éclatant les relations. Le lien entre ces relations est assuré avec la contrainte d'intégrité référentielle (clé étrangère) afin de pouvoir retrouver les informations à partir de la clé primaire. Ce principe couteux en temps d'exécution est généralement inexistant dans les SGBD NoSQL (dénormalisation), bien qu'il soit possible de le simuler en Mongodb avec $Lookup [7]. Pour pouvoir modéliser une base Mongodb, il faudra remettre en cause toutes les habitudes et adopter une nouvelle façon de concevoir la structure de données, tout en gardant à l'esprit, qu'il n'est pas optimal de tenter de faire du relationnel avec une base MongoDb (nombre de collections = nombre de tables)
Deux alternatives de modélisation [8] sont proposées par Mongodb : l'imbrication des données et les références. Le choix du type de modélisation adéquat est très important et dépend de la future application à construire. Dans ce sens, il faudra :

- Connaitre comment l'application produira, utilisera et traitera les données (documents). Comment l'application va-t-elle récupérer et interroger des données ?
- Pour orienter l'application vers la collection la plus sollicitée et imbriquer les autres il faudra savoir :
  - Quelles sont les données fréquemment interrogées conjointement ?
  - Si l'application exige de nombreuses lectures (read heavy) ou de nombreuses écritures (write heavy) ;
- Les données qui sont le moins dépendantes des autres ;
- Le taux de mise à jours de vos données.

4.1.3.1 Imbrication

L'imbrication est une dénormalisation du modèle de données. Par exemple, la représentation d'un modèle relationnel, où une table 'A' référence une table 'B' est représentée en mongodb par une seule collection 'A'.
Chaque document de la collection 'A' contient une paire clé-valeur où :
- La clé représente le nom ou la sémantique du nom de la table référencée 'B' ;
- La valeur est un objet (document imbriqué) qui contient toutes les données (champs- valeurs) de la ligne référencée de la table 'B'.
En général, l'utilisation de l'imbrication est préconisée quand :

- Les données imbriquées sont une partie intégrante des données d'un document (cas d'une relation 1-1) ;

- Il existe des relations de type "contient" entre des entités ;
- Il existe des relations de type "un-à-plusieurs" entre des entités ; par contre, les données incorporées changent rarement sinon la mise à jour est difficile du fait de la duplication des données.
- Les données incorporées ne croîtront pas sans limite.

4.1.3.2 Références

L'utilisation de références est proche du modèle de données normalisé. Un document d'une collection pourra faire référence à un autre document d'une autre collection via son _id. Cependant, mongodb ne supporte pas la jointure comme dans le relationnel. En effet, la lecture et l'écriture ne sont pas des opérations atomiques et nécessitent plusieurs allers retours.
Les références sont utilisées quand l'imbrication n'est pas avantageuse par rapport à la nature de l'application (Heavy write) c à d :

- Le nombre de données imbriquées est illimité : l'augmentation de la taille du document a une incidence sur les possibilités de lecture et de mise à jour du document à l'échelle.
- Les données imbriquées sont souvent utilisées dans d'autres documents et changent fréquemment.

4.1.3.3 Hybride

Les solutions hybrides offrent un bon compromis entre dénormalisation et redondances de données où les données sont imbriquées sous forme de document imbriqué qui contient la référence vers l' id du document en plus des informations les plus utilisées.

4.2 Exercices

4.2.1 Exercice 1 : Transformation du relationnel (1-N) vers les 4 types de schémas NoSQL

Soit le schéma relationnel (voir figure 4.2) de la base "Livres_auteurs". Nous supposons, dans un premier temps, qu'un livre n'est écrit que par un seul auteur, c.à.d, la relation écrire est de type 1-n au lieu de n-n.

- Donner la transformation de ce schéma dans les 4 types de schéma NoSql : Orienté Graphe, Clé-valeur, Orienté colonne et Orienté Document en donnant un exemple de SGBD et de

création de chacun d'eux.

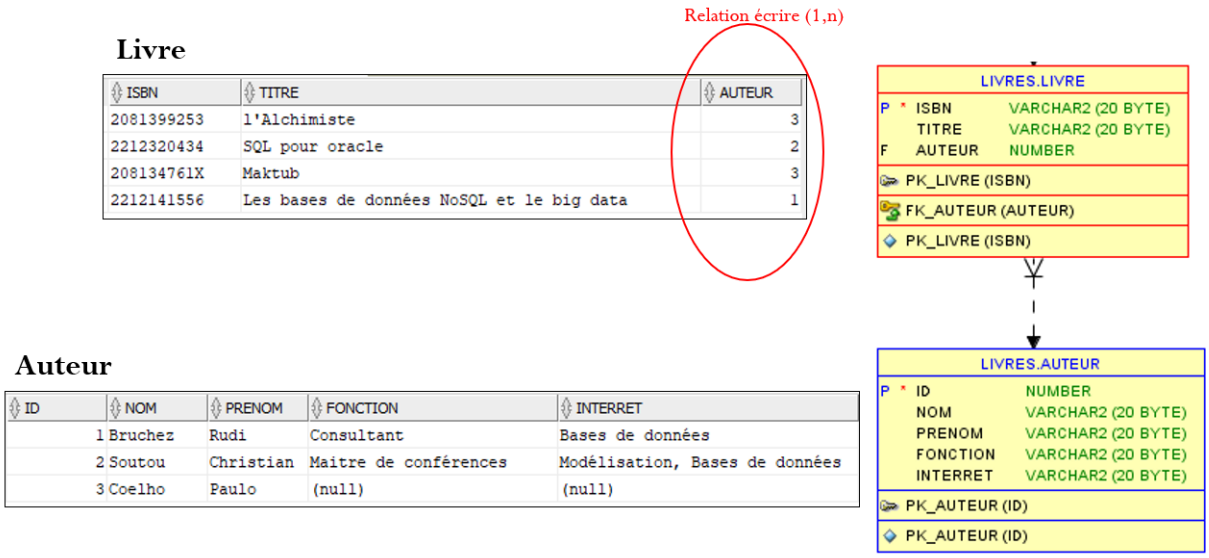

FIGURE *4.2:* *Schéma relationnel de la base "Livres_auteurs"*

4.2.2 Exercice 2 : Format des documents Mongodb (Json)

Soit le document JSON Film suivant :

```
{
"_id": 1,
"title": "Taxi driver",
"year": 1976,
"genre": "drama",
"summary": 'Vétéran de la Guerre du Vietnam, Travis Bickle est chauffeur
    de taxi dans la ville de New York. La violence quotidienne l'
   affecte peu à peu.',
"country": "USA",
"director": {
"last_name": "Scorcese",
first_name: "Martin",
"birth_date": "1962"
},
"actors": [
{
first_name: "Jodie",
"last_name": "Foster",
"birth_date": null,
"role": "1962"
}
{
first_name: "Robert",
```

```
"last_name": "De Niro",
"birth_date": "1943",
"role": "Travis Bickle ",
}
}
```

1. Le document JSON est-il bien formé ? relevez les erreurs.
2. Donnez une représentation arborescente du document
3. Que représente le champs _id.

4.2.3 Exercice 3 : Modélisation Mongodb : Relation n-n

Reprenons l'énoncé de l'exercice 1 du schéma "Livres_auteurs" en considérant, cette fois ci, une relation "plusieurs à plusieurs" (n-n). Une première transformation de ce schéma en une base orientée documents Mongodb a été faite.
Le Schéma 1 présente une vue sur les collections créées de la base avec un jeu de données.

Schéma 1: Collections de la base "Livres_auteurs" : Modélisation n-n

Collection auteur :

```
{"id": "a1", "nom": "Soutou", "prenom":"christian" }
{"id": "a2", "nom": "Bruchez", "prenom":"Rudi" }
{"id": "a3", "nom": "Andersen", "prenom":"Thomas" }
{"id": "a4", "nom":"Brouard", "prenom ":"Frédéric"}
```

collection livre :

```
{"id": "b1", "isbn":"2212320434", "titre": "Sql pour Oracle" }
{"id": "b2", "isbn": "2212141556", "titre":"Les bases de données
   NoSQL et le big data" }
{"id": "b3", "titre": "Azure Cosmos DB for RDBMS Users" }
{"id": "b4", "titre": "Taking over the world one JSON doc at a
   time" }
```

Collection auteur_livre :

```
\begin{lstlisting}[language=json]
{"authorId": "a1", "bookId": "b1" }
{"authorId": "a2", "bookId": "b2" }
{"authorId": "a3", "bookId": "b4" }
{"authorId": "a3", "bookId": "b3" }
```

1. Que pensez-vous de cette modélisation ?. Donner une autre transformation en réduisant votre schéma à deux collections en utilisant les références.
2. Transformer le schéma de la base en une seule collection "Auteur"

3. Transformer le schéma de la base sachant que l'application à développer consiste à afficher toutes les informations des livres avec quelques informations de l'auteur. Un exemple de la page d'accueil est donné dans la figure 4.3. Les détails de l'auteur ne sont affichés qu'au besoin.

FIGURE *4.3:* *Exemple 1 de la page d'accueil de l'application "Livres_auteurs"*

4. Supposons maintenant que l'application à développer consiste à n'afficher que quelques informations sur le livre et quelques informations sur l'auteur (voir figure 4.4) et, au besoin, afficher le détail soit de l'auteur soit du livre (Voir figure 4.5).

FIGURE *4.5:* *Exemple 2 de la page détails Livre de l'application "Livres_auteurs"*

4.2.4 Exercice 4 :Modélisation Mongodb : Exemple complet

On souhaite réaliser une base de données orientée document pour gérer les évènements sportifs algériens (voir 4.6) :

- Un évènement est décrit par les attributs code, titre, description, prix d'inscription.
- Le Coach sportif de l'évènement est décrit par les attributs Num, nom, téléphone. Un coach peut animer plusieurs événements et un évènement peut être animé par plusieurs coachs à la fois.
- Établissement ou lieu d'événement est décrit par les attributs : Libellé, adresse, téléphone.
- Les adresses sont composées d'un numéro de rue, d'une ville et d'un code postal.
- Un établissement peut accueillir plusieurs événements à des dates différentes, un événement se déroule dans un seul établissement. Par exemple, l'événement DZ_Zumba_Day 2020 se déroulera dans le Stade Frères Zerga le 21/06/ 2020. Il sera animé par les coachs Fighouli et Melloul à 14h00.

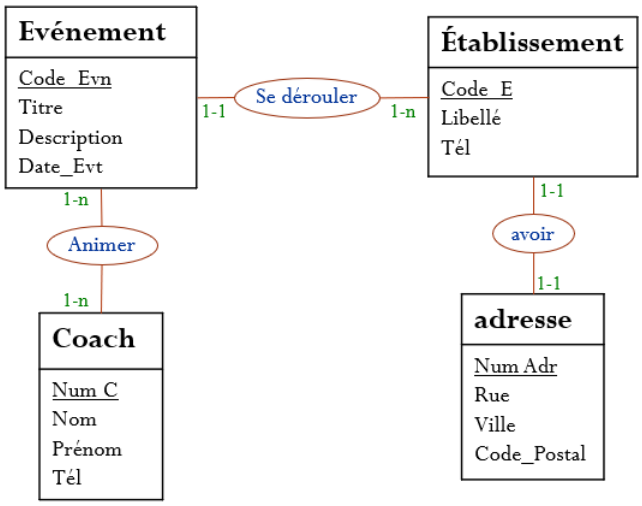

FIGURE *4.6:* *Schéma Entité-association de la base "Événements sportifs"*

1. Proposer une modélisation équivalente à un MLD relationnel normalisé, en donnant un exemple d'un document des collections de cette modélisation. Ce type est-il adapté aux BD orientées document ?
2. Proposer une modélisation basée sur l'imbrication. En favorisant, dans un premier temps, la collection "Évènement". Et dans un second temps, la collection "Établissement".
3. Proposer une modélisation basée sur les références.

4. Sachant que l'objectif de l'application est de visualiser une liste de noms (Événements, Établissement, Animateurs) et d'accéder aux détails lorsqu'on clique sur le nom de chacun. Proposer une solution adaptée à ce problème.

4.3 Solutions d'exercices

4.3.1 Solution de l'exercice 1

1. **Schéma orienté Graphe**

   Nous aurons deux types de nœuds : Type Livre et Auteur. Chaque enregistrement de la table est représenté par un nœud. La relation "écrire" est représentée par un arc qui relie l'enregistrement de la table "livre" à celui de la table "Auteur".

   (a) **Exemple de transformation du schéma**

   Un exemple de transformation du schéma est donné dans la figure 4.7

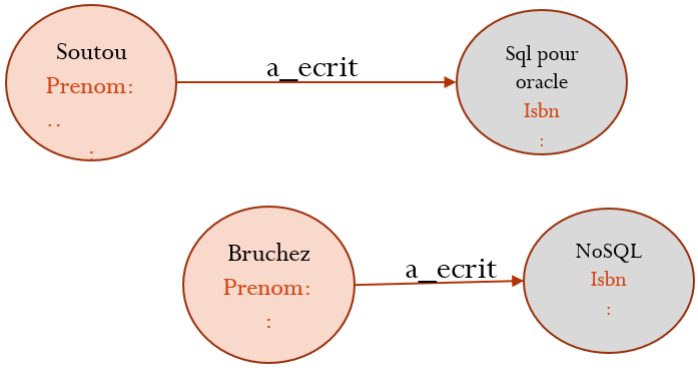

FIGURE 4.7: *Exemple de schéma Orienté Graphe de la base "Livres_Auteurs"*

   (b) **Exemple du SGBD et requêtes**

   Parmi les SGBD les plus utilisés Neo4j [1] avec le langage Cypher

   - Création des nœud auteurs

```
CREATE (Soutou:personne {nom:"Soutou",prenom:"Christian"})
CREATE (SQl_Oracle:livre {titre:"Sql pour oracle", isbn:"
    2212320434"})
```

   - Création de la relation écrire

```
CREATE (Soutou)-[:ecrit]->(SQl_Oracle)
RETURN Soutou,SQl_Oracle
```

1. https ://neo4j.com/

- Retourner un nœud avec une propriété

```
MATCH (n {isbn:'2212320434'}) RETURN n
```

2. **Schéma orienté clé valeur**

(a) **Exemple de transformation du schéma**

Il n'y pas une normalisation de transformation. Donc, plusieurs solutions de modélisation peuvent être envisagées.

- **Solution 1 :** On peut avoir un ensemble de clé valeurs uniques (figure 4.8) avec l'association de clé aux attributs : **SET** Nom1 Bruchez, **SET** prénom1 Rudi, **SET** fonction1 consultant, **SET** interert1 bdd. A noter que cette solution n'est pas intéressante.

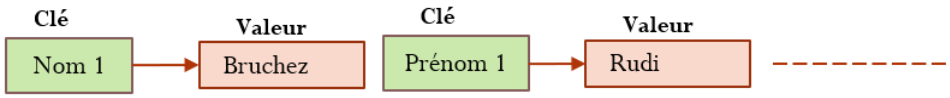

FIGURE *4.8:* *Exemple 1 du schéma clé valeur de la base "Livres_Auteurs"*

- **Solution 2 :** On peut utiliser (Figure 4.9) un **HSET** associé aux clés : **HSET** 1 Nom Bruchez, **HSET** 1 Prénom Rudi, **HSET** 1 Fonction : Consultant.

  Attention : précéder la clé par le nom de la table HSET auteur1 Nom Bruchez,..etc. Car, la clé de la HSET est unique et ne sera pas utilisable pour la clé 1 de la table Livre.

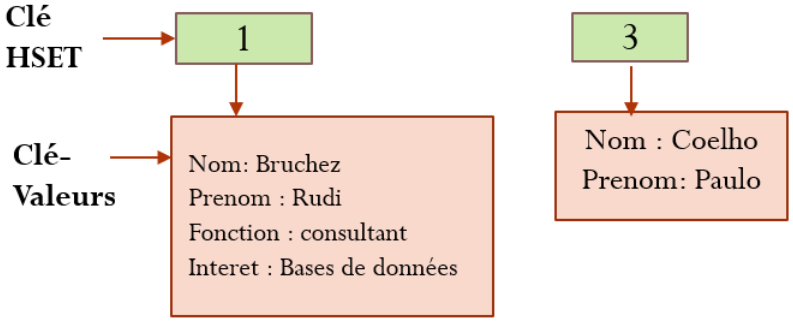

FIGURE *4.9:* *Exemple 2 du schéma clé valeur de la base "Livres_Auteurs"*

(b) **Solution 3 :** Nous pouvons utiliser un SET pour les valeurs de la table et un HSET [9] pour les associations clé auteur, nom auteur (pour rechercher par nom et retrouver la clé).

(c) **Exemple du SGBD et requêtes** Le SGBD REDIS [2]

- solution 1

```
SET 1 Bruchez
GET 1
```

- solution 2

```
HSET 1 Nom Bruchez
HSET 1 Prenom RUDI
HSET 1 Fonction Consultant
```

Pour voir toutes les autres commandes et types de stockage de Redis allez sur http ://redis.io/commands.

3. **Schéma orienté Document**

Chacune des deux tables sera représentée par une collection qui portera le même nom. Chaque enregistrement de la table correspond à un document dans la collection correspondante (voir figure 4.10).

(a) **Exemple de transformation du schéma** Un exemple de la transformation de la base relationnelle "Livres_Auteurs" vers un schéma orienté document est donné dans la figure 4.10

(b) **Exemple du SGBD et requêtes** L'un des SGBD le plus utilisé est mongodb [3].

Insérer un document dans la collection auteur :

```
db.auteur.insert({"_id": 3, "nom": "Coelho", "prenom": "
    Paulo" })
```

Retourner tous les documents de la collection :

```
db.auteur.find()
```

Retourner toutes les collections :

```
show collections
```

Restriction :

```
db.auteur.find({"nom":"coelo"})
```

2. redis.io
3. //www.mongodb.com/

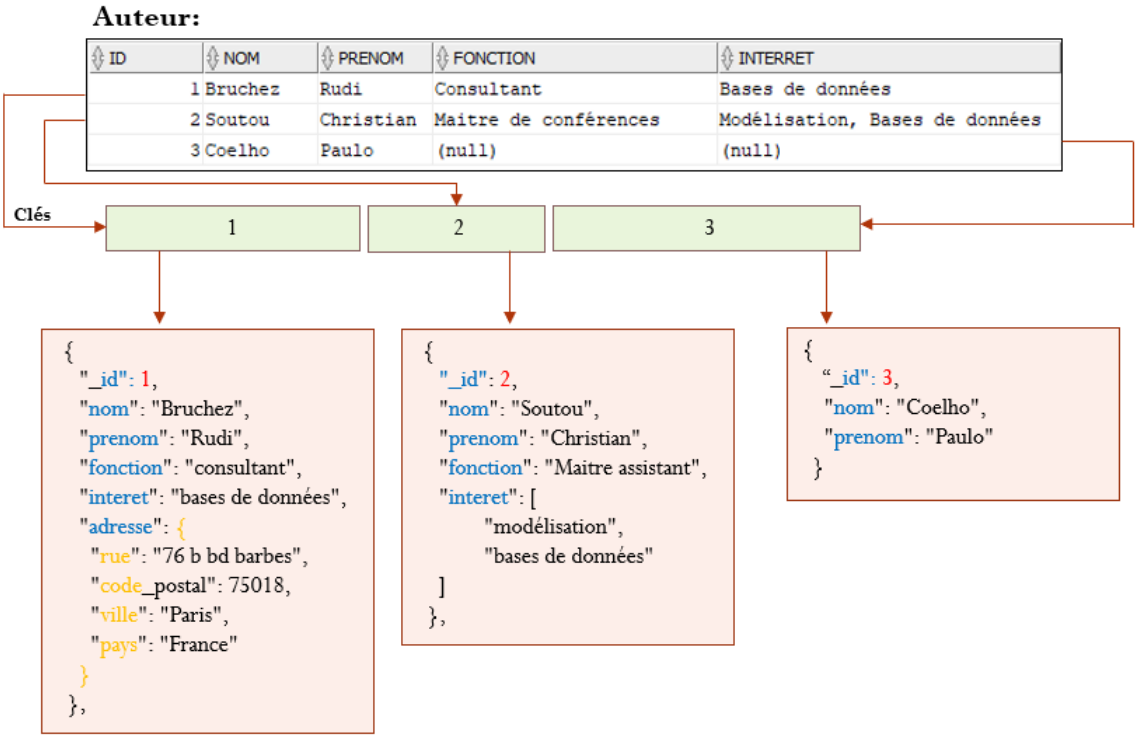

FIGURE ***4.10:*** *Exemple de schéma Orienté document de la table Auteur de la base "Livres_Auteurs"*

4. **Schéma orienté Colonne** :

   (a) **Exemple de transformation du schéma**

   Un exemple de la transformation de la base relationnelle "Livres_Auteurs" vers un schéma orienté Colonne est donné dans la figure 4.11

   (b) **Exemple de SGBD**

   - Hbase : http ://hbase.apache.org/

   - Google Big Table : https ://cloud.google.com/bigtable/

   - Hypertable : http ://www.hypertable.org/

   - Apache Parquet : https ://parquet.apache.org/

4.3.2 Solution de l'exercice 2

1. Le document n'est pas bien formé car il contient plusieurs erreurs à savoir : chaque élément (clé) doit être sous forme de string donc l'élément *first name* doit être entouré de ". Les séparateurs utilisés entre deux éléments sont des virgules sauf pour le dernier élément. Le document après correction est présenté dans le listing 4.1.

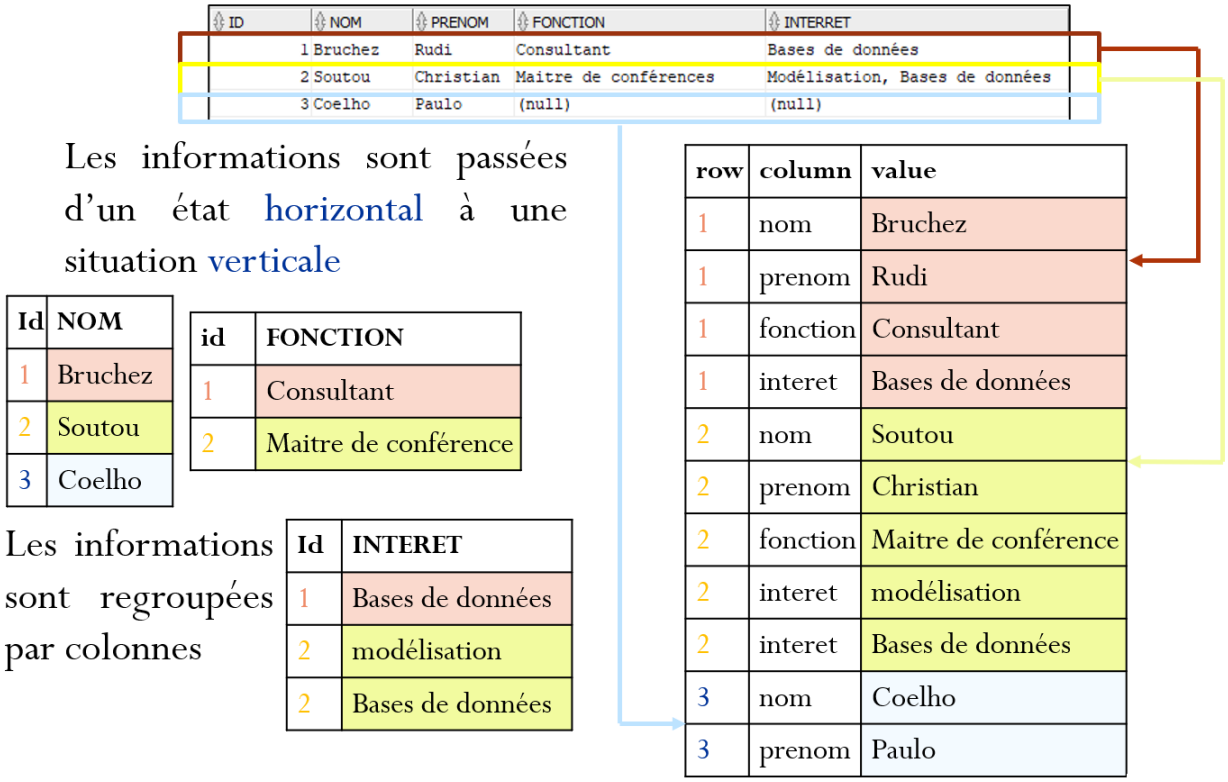

FIGURE *4.11:* *Exemple de schéma Orienté colonne de la base "Livres_Auteurs"*

Listing 4.1: *Document Json corrigé*

```
{
"title": "Taxi driver",
"year": 1976,
"genre": "drama",
"summary": "Vétéran de la Guerre du Vietnam, Travis Bickle est
    chauffeur de
 taxi dans la ville de New York. La violence quotidienne l'affecte
     peu à peu.",
"country": "USA",
"director": {
"last_name": "Scorcese",
"first_name": "Martin",
"birth_date": "1962"
},
"actors": [
{
"first_name": "Jodie",
"last_name": "Foster",
"birth_date": null,
"role": "1962"
},
{
"first_name": "Robert",
"last_name": "De Niro",
"birth_date": "1943",
"role": "Travis Bickle "
}]
}
```

2. L'arborescence du document est présentée dans la figure 4.12. L'arborescence a été réalisée par l'outil JsonViewer [4]

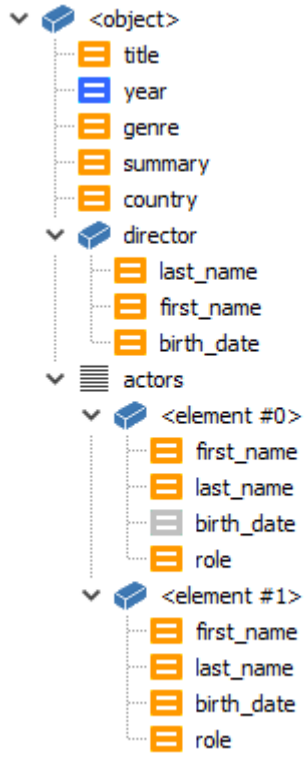

FIGURE *4.12:* *Arborescence du document JSon "Film"*

3. "_id" représente la clé unique de chaque document dans une collection. L'attribut "_id" peut être comparé à la notion de clé primaire d'une table relationnelle. La valeur de l'"id" peut être fournie par l'utilisateur au moment de l'insertion d'un document ou générée automatiquement (ObjectId ())

4. http ://www.mitec.cz/jsonv.html

4.3.3 Solution de l'exercice 3

1. **Solution avec références :**

   Cette solution (voir schéma 2) est déconseillée si l'application n'est pas centrée sur la relation auteur_livre. Le fait de charger soit un auteur avec ses livres soit un livre avec ses auteurs nécessite toujours au moins deux requêtes supplémentaires sur la base de données. De ce fait, la solution est non optimale car on n'a pas dénormalisé.

   Notons que les références peuvent êtres utilisées seulement dans l'une des deux collections, selon les besoins de l'application (orientée livres ou auteurs).

Schéma 2: Modélisation avec références de la base "Livres_auteurs"

Collection Auteur :

```
{"id": "a1", "nom": "Soutou", "prenom":"christian", "livre":
    [ "b1", "b4"]}}
{"id": "a2", "nom": "Bruchez", "prenom":"Rudi","livre": ["b2
    "] }
{"id": "a3", "nom": "Andersen", "prenom":"Thomas", "livre":
    ["b3"]} }
{"id": "a4", "nom":"Brouard", "prenom ":"Frédéric", "livre
    ":["b4"]
```

Collection Livre :

```
{"id": "b1", "isbn":"2212320434", "titre": "Sql pour Oracle",
     "auteur": ["a1"]}}
{"id": "b2", "titre": "Les bases de données NoSQL et le big
    data","auteur": ["a2"]  }
{"id": "b3", "titre": "Azure Cosmos DB for RDBMS Users","
    auteur": ["a3"] }
{"id": "b4", "isbn": "2212141556", "titre":"Modélisation des
    bases de données","auteur": ["a1, "a4"]}}
```

2. Si l'application manipule et affiche les livres, on dit que l'application est orientée collection livre.

   Dans ce cas, on ne crée que la collection "livre" et on imbrique les auteurs dans cette collection (voir Schéma 3).

Schéma 3: Modélisation de la base "Livres_auteurs" avec une seule Collection Livre (imbrication orientée collection "Livre")

Collection "Livre"

```
{"id": "b1", "isbn":"2212320434", "titre": "Sql pour Oracle",
    "auteur": [{"id": "a1", "nom": "Soutou", "prenom":"
    christian" }
]}}
{"id": "b2", "titre": "Les bases de données NoSQL et le big
    data","auteur": [{"id": "a2", "nom": "Bruchez", "prenom":"
    Rudi" }]  }
{"id": "b3", "titre": "Azure Cosmos DB for RDBMS Users","
    auteur": [{"id": "a3", "nom": "Andersen", "prenom":"Thomas
    " }
] }
{"id": "b4", "isbn": "2212141556", "titre":"Modélisation des
    bases de données","auteur": [{"id": "a1", "nom": "Soutou",
    "prenom":"christian" } , {"id": "a4", "nom":"Brouard", "
    prenom ":"Frédéric"}]}
```

Ce genre de solution est rapide en lecture (cas d'application heavy ready), mais contient beaucoup de redondances, il faudra aussi faire attention aux tableaux avec un nombre d'objets croissant très grand et sans limite.

3. La solution (voir schéma 4) est une modélisation Hybride Orientée collection "Livre".

Schéma 4: Modélisation hybride orientée collection "Livre"

Exemple d'un document de la collection "Livre" :

```
{   "id": "b2",
    "name": "Les bases de données NoSQL et le big data",
    "isbn": "",   "date_parrution":"24/04/2015",
    "couverture_livre":"http://....png"
    "authors": [
        {"id": "a1", "name": "Rudi_Bruchez", "thumbnailUrl":
            "http://....png"}   ]
}
```

Exemple d'un document de la collection auteur :

```
{  "id": "a2",   "nom": "Bruchez",   "prenom": "Rudi",   "
    fonction": "consultant",
  "interet": ["bases de données"],  "adresse": {"rue":  ,..}
 "Image_Auteur":"http://....png"
}
```

4. La solution est une modélisation hybride orientée sur la relation "auteur_livre". De ce fait, trois collections doivent êtres créées. La solution est présentée dans le schéma 5

Schéma 5: Modélisation hybride Orientée sur la relation "auteur_livre" de la base "Auteurs_livres"

Collection "auteur_livre" (qq infos du livre + ref livre, qq infos auteur + ref auteur) :

```
{    "couverture_livre":"http://....png"
     "titre": "Les bases de données NoSQL et le big data",
     "date_parrution":"24/04/2015",
     "détail_livre":"b2",
     "authors": [
         {"id": "a1", "name": "Rudi_Bruchez"}
             ]
   }
```

Collection auteur :

```
{
"id": "a2",    "nom": "Bruchez",    "prenom": "Rudi",    "
    fonction": "consultant",    "interet": ["bases de données
    "], "adresse": {"rue": "76 b bd barbes",..},     "
    Nombre_livre": 7, "Image_Auteur":"http://....png",  "
    livres": ["b2",...], }
```

Collection livre

```
{"id": "b2",
"titre": "Les bases de données NoSQL et le big data" , "Broch
    é": 322 , "Editeur" : "Eyrolles";
"Collection" : Blanche, "Langue" : Français, "ISBN-10":
    2212141556
"ISBN-13": 978-2212141559, "Dimensions du produit": "22,7 x 2
     x 19 cm " }
```

4.3.4 Solution de l'exercice 4

1. **Solution équivalente au relationnel.**

   La base à créer dans mongodb contient exactement les mêmes collections que les tables du schéma relationnel 4.13.

   Notons que la transformation de la relation "avoir adresse" de type 1-1 a été fusionnée avec la table établissement au lieu de créer une autre table adresse.

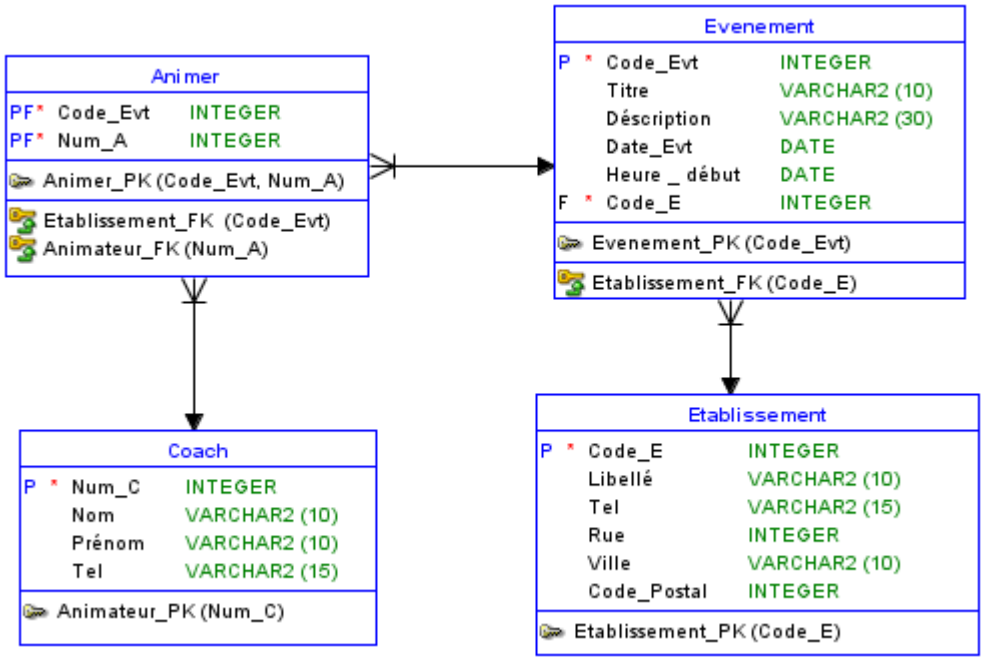

FIGURE *4.13:* *Schéma relationnel de la base "Événements sportifs"*

Ci dessous, les collections de la base avec un exemple des documents de chaque collection :

- La collection Établissement

Listing 4.2: *Collection Établissement de la modélisation équivalente au relationnel*

```
{
  "_id": 11,
  "libellé": "Stade 3 Frères zerga",
  "tel": "06 63 57 49 69",
  "adresse": {
    "rue": "boulevard Mohammed 5,",
    "ville": "Tlemcen",
    "code_postal": 13000
  }
}
```

Req : nous utilisons dans le document de la collection "Établissement" la valeur de la clé primaire "Code_E" pour l'identifiant "_id" qui représente la clé du document Json de mongodb au lieu du ObjectId qui est généré de manière automatique par mongodb. La même chose sera faite pour les autres tables sauf si on laisse Mongodb générer automatiquement la clé (c'est le cas pour la collection "Animer").

- La collection Événement

Listing 4.3: *Exemple de doc de collection Événement de la modélisation équivalente au relationnel*

```
{
  "Code_Evt": 111,
  "Titre": "Zumba Days",
  "Description": "Les Journées Zumba est un événement sportif
     qui permet de débarrassez votre corps physique et é
     motionnel des toxines, des tensions, ....",
  "Date_Evt": "21/12/2020",
  "Heure_Début": "14h00",
  "Code_E": 11
}
```

- La collection Coach

Listing 4.4: *Exemple de doc de la collection Coach de la modélisation équivalente au relationnel*

```
 {
    "Num_C": 1,
    "Nom": "Melloul",
    "Prénom": "Saliha",
    "Tel": "05 57 28 49 70"
  }
```

- La collection Animer

Listing 4.5: *Exemple de doc de la collection Animer de la modélisation équivalente au relationnel*

```
{ "_id" : ObjectId("5a99bded567089e00a38a0e4"),
  "Code_Evt": 111,
  "Code_E": 11,
}
```

Ce type de solution n'est généralement pas favorisé en BD orientée document. Il n'est pas conseillé de transformer le modèle relationnel tel quel, car il est préférable de minimiser les jointures qui sont assurées dans Mongodb par des requêtes avec le framework d'agrégation par l'opérateur $lookup

2. **Solution basée sur l'imbrication**

(a) Modélisation orientée Événement :

Listing 4.6: *Schéma de la collection Événement (modélisation basée sur l'imbrication)*

```
{
  "Code_Evt": 111,
  "titre": "Zumba Days",
  "description": "Les Journées Zumba est un évenement sportif
     qui permet de débarrassez votre corps physique et é
     motionnel des toxines, des tensions, ....",
```

```
  "Date_Evt": "21/12/2020",
  "Heure_Début": "14h00",
  "Etablissemnt": {
    "_id": 11,
    "libellé": "Stade 3 Frères zerga",
    "tel": "06 63 57 49 69",
    "adresse": {
      "rue": "boulevard Mohammed 5,",
      "ville": "Tlemcen",
      "code_postal": 13000
    },
    "Coachs": [
      {
        "Num_C": 1,
        "nom": "Melloul",
        "prénom": "Saliha",
        "tel": "05 57 28 49 70"
      },
      {
        "Num_C": 2,
        "nom": "Fighouli",
        "prénom": "Djamila",
        "tel": "06 61 57 50 90"
      }
    ]
  }
}
```

Le principal défaut de cette solution est sa redondance : toutes les données de l'établissement sont copiées dans chaque événement qui se déroule dans le même établissement. Idem pour le coach ;

(b) Orientée Établissement :

Listing 4.7: *Schéma de la collection Établissement (modélisation basée sur l'imbrication)*

```
{
  "_id": 11,
  "libellé": "Stade 3 Frères zerga",
  "tel": "06 63 57 49 69",
  "adresse": {
    "rue": "boulevard Mohammed 5,",
    "ville": "Tlemcen",
    "code_postal": 13000
  },
  "Évènements": [
    {
      "Code_Evt": 111,
      "titre": "Zumba Days",
      "description": "Les Journées Zumba est un évènement
          sportif qui permet de débarrassez votre corps
          physique et émotionnel des toxines, des tensions,
          ....",
      "Date_Evt": "21/12/2020",
      "Heure_Début": "14h00",
      "Coachs": [
```

```
        {
          "Num_C": 1,
          "nom": "Melloul",
          "prénom": "Saliha"
        },
        {
          "Num_C": 2,
          "nom": "Fighouli",
          "prénom": "Djamila",
          "tel": "06 61 57 50 90"
        }
      ]
    }
  ]
}
```

Pour assurer la relation 1-n l'élément Événements est un tableau d'objets.

3. **Solution basée sur les références**

   (a) Orientée Événement :

Listing 4.8: *Collection "Événement" (modélisation basée références)*

```
{
  "Code_Evt": 111,
  "Titre": "Zumba Days",
  "Description": "Les Journées Zumba est un évènement sportif
     qui permet de débarrassez votre corps physique et é
     motionnel des toxines, des tensions, ....",
  "Date_Evt": "21/12/2020",
  "Heure_Début": "14h00",
  "Etablissemnt": {
    "Code_E": 11,
    "Coachs": [1,2]
  }
}
```

   (b) Orientée Établissement :

Listing 4.9: *Collection "Établissement" (modélisation basée références)*

```
{
  "_id": 11,
  "libellé": "Stade 3 Frères zerga",
  "tel": "06 63 57 49 69",
  "adresse": {
    "rue": "boulevard Mohammed 5,",
    "ville": "Tlemcen",
    "code_postal": 13000
  },
  "Evenements": [
    {
      "Code_Evt": 111,
      "Coachs": [1,2]
    }
  ]
}
```

4. Il s'agit d'utiliser une **solution hybride**. En plus des trois collections (Établissement, Coach et Événement) , il faudra créer la quatrième collection "Annonce_Evt". Cette dernière contient les références plus les trois attributs qui apparaissent sur la page d'accueil de l'application à savoir : nom d'établissement, événement et coach. Les détails relatifs à ces trois attributs n'apparaissent qu'en cas de besoins.

   Un exemple de la collection "Annonce_Evt" est donnée dans le listing 4.10

Listing 4.10: *Collection "Annonce_Evt" (modélisation hybride)*

```
{
  "Titre_Evenemnt": "Zumba_Days",
  "Détails_Evt": 111,
  "Nom_établissement": "Stade 3 frères Zerga",
  "Détail_2tablmissement": 11,
  "Animateurs": [
    {
      "Coach": "Melloul Saliha",
      "Détail_Coach": 1
    },
    {
      "Coach": "Fighouli Djamila",
      "Détail_Coach": 2
    }
  ]
}
```

Série TD 5

Commandes Mongodb

Sommaire

5.1 Objectifs et Rappels

Cette série d'exercices a pour but de se familiariser avec les commandes usuelles et essentielles de Mongodb [7]. Un rappel de cette syntaxe est donné par catégorie dans les sections suivantes.

5.1.1 Opérations CRUD

a) **Opérations sur la base de donnée**

Le tableau 5.1 regroupe les commandes basiques de création, suppression et affichage d'une base de données Mongodb

Syntaxe	Sémantique de la commande
show dbs	Connaître les bases stockées dans le répertoire géré par le serveur
db	Afficher le nom de la base actuelle
use <nomBase>	Basculer à une autre base, si cette dernière n'est pas créée elle le sera à la première création de collection
db.dropDatabase()	Effacer la base courante, la base est effacée sur disque mais continue d'être référencée

Tableau 5.1: *Commandes Mongodb des opérations CRUD (bases de données)*

b) **Opérations sur les collections**

Le tableau 5.2 regroupe quelques commandes usuelles de création, modification et suppression des collections Mongodb

Syntaxe	Sémantique de la commande
`db.createCollection(<nomCollection>);`	Créer une collection
`show collections;`	Lister les collections
`db.<nomCollection>.rename(NouveauNom)`	Renommer la collection
`db.<nomCollection>.drop( )`	Supprimer la collection

Tableau 5.2: *Commandes Mongodb des opérations CRUD sur les collections*

c) **Opérations sur les documents d'une collection**

Le tableau 5.3 regroupe quelques commandes usuelles de création, modification et suppression des documents d'une collection Mongodb. Ces trois opérations peuvent concerner le document en entier ou un(des) attribut(s) du document. Dans le cas où l'attribut est un tableau, il faudra utiliser les opérateurs sur les tableaux de la fonction UPDATE.

Syntaxe	Sémantique de la commande
`db.<nomCollection>.insert({<attribut1>:<valeur1>,<attribut2>:<valeur2>, .. })`	Insérer un document dans la collection, le paramètre de la fonction insert doit être un document
`db.<nomCollection>.insertMany([{<attribut1>:<valeur1>,..},{<attribut2>:<valeur2>,..}])`	Insérer plusieurs documents dans la collection
`db.<nomCollection>.find()`	Afficher les données des documents
`db.<nomCollection>.update({<attribut1>:<valeur1>,..},{<attribut1>: <valeur1,..>})`	Le premier paramètre de la commande update permet de sélectionner le document selon les valeurs des couples attributs :valeurs, le deuxième permet de modifier en écrasant le document par les nouvelles données en gardant le même ID du document modifié.
`db.collection.updateMany()`	Ajouter et/ou modifier un champ à plusieurs documents
Opérateurs sur les attributs du Update	
`db.<nomCollection>.update({},{$set:{<attribut1>:<valeur1>,..}})`	Modifier et/ou rajouter un champ au document (opérateur $set)
`db.<nomCollection>.update({},{$unset:{<attribut1>:""}})`	Supprimer un attribut.
`db.<nomCollection>.update({},{$rename:{<attribut1>: <nouveauNom1>, ... }})`	Renommer un attribut.
Opérateurs sur l'attribut tableaux du Update	
`db.<nomCollection>.update({},{$set:{<attribut>:<valeur>}})`	<attribut> est un champ de type array il faudra utiliser la notation pointée pour représenter l'indice du tableau et modifier un élément EXP : attribut.0 est le premier élément du tableau.
`db.<nomCollection>.update({},{$addToSet:{<attribut1>:<valeur1>,..}})`	L'opérateur $addToset permet d'ajouter sans doublon
`db.<nomCollection>.update({},{$push:{<attribut1>:<valeur1>,..}})`	Rajouter les valeurs à la fin du tableau
`db.<nomCollection>.update({},{$pull:{<attribut1>:<value1|condition>,... } )`	Supprimer les valeurs d'un tableau
`db.<nomCollection>.remove( )`	Pour supprimer tous les documents de la collection. Dans le cas de suppression d'un ou quelques documents il faudra spécifier un filtre entre {}

Tableau 5.3: *Commandes Mongodb des opérations CRUD sur les documents d'une collection*

5.1.2 Interrogation des données

L'interrogation des données se fait à l'aide de la fonction find() de la commande :

```
db.collectionName.find({...},{...})
```

La commande contient deux arguments :

- Le premier argument est la condition de sélection/filtrage où seuls les documents satisfaisant cette condition seront retenus ;
- Le second argument décrit la projection sur les documents retenus

C'est l'équivalent en sql :

```
SELECT <attributs du deuxième augment> FROM collectionName
WHERE <conditions du premier paramètre>
```

Un ensemble d'opérateurs (voir tableau5.4) peut être utilisé dans la projection ou sélection pour écrire les conditions de la requête.

Opérateur	Sémantique de l'opérateur
logiques	
$and, $or	Et logique, ou logique
$not, $nor	NOT et NOR logique
comparaisons	
$eq, $ne	==, !=
$gt, $gte, $lt,$lte	>, >=, <, <=
test sur éléments	
$exists	Existence d'un champ
$type	Teste le type d'un champ
évaluation	
$mod	Calcule et teste le résultat d'un modulo
$regex	Recherche d'expression régulière
$text	Analyse d'un texte
Tableaux	
$all	Test sur le contenu
$elemMatch	Limite le contenu d'un tableau à partir des résultats de la requête pour ne contenir que le premier élément correspondant à la condition $elemMatch.
$size	Taille du tableau

Tableau 5.4: *Liste des opérateurs (requête de sélection/projection) de la commande find*

Quelques Exemples de requêtes :

- L'opérateur $and :

```
db.livres.find({$and :[{langue : "Français"},{annee:{$gte:
    2010}},{annee:{$lt:2012}}]},{titre:1,_id:0})
```

La requête retourne tous les livres (titres) écrits en français et sortis entre : 2010 (inclus) et 2012 (exclus).

- L'opérateur $regex

```
db.livres.find({titre : {$regex : "sql" , $options : "i"} })
```

Retourne les livres qui contiennent sql, ou Sql, ou SQL,... L'option "i" est insensible à la casse.

5.1.3 Framework d'agrégation

Depuis la version 2.2, MongoDB propose un framework d'agrégation qui permet de :

- Réaliser des opérations complexes d'analyse ;
- Récupérer et manipuler les données en pipeline dans Mongodb sans utiliser des traitements complexes en batch avec Map/Reduce [10]

```
db.<collectionName>.aggregate([{<Étape1>}, {<Étape2>}, {<Étape3>}..])
```

L'intérêt de ce pipeline d'opérations est que le résultat de sortie d'une étape est l'entrée de la suivante jusqu'à la sortie finale de l'agrégation.
Dans chaque étape, un opérateur [11] est utilisé. Nous présentons dans le tableau 5.5, ceux qui sont les plus utilisés.

Opérateur	sémantique
$Group	Permet de regrouper les enregistrements selon l'_id (qui peut être redéfini), et d'appliquer des fonctions de groupe ($sum,$min,$avg,etc.) aux autres attributs projetés.
$match	Permet de sélectionner/filtrer les documents à passer au pipeline, selon une ou des conditions.
$sort	Permet de trier les documents et de les transmettre de façon ordonnée au pipeline
$project	permet de mettre en forme les documents avec les champs existants ou de nouveaux champs, avant de les passer au pipeline
$unwind	Permet de réorganiser les champs d'un tableau, avec un document pour chaque élément du tableau.
$out	Permet de créer une collection avec le résultat d'un pipeline
$lookup	Permet de faire une "jointure" entre des objets de deux collections

Tableau 5.5: *Quelques opérateurs du framework d'agrégation*

- Exemple :

Soit les schémas suivant

```
{"_id" : 1, "item" : "a", "prix" : 10, "quantite" : 2, "date" : ISODate
    ("2014-03-01T08:00:00Z") }
{"_id" : 2, "item" : "z", "prix" : 10, "quantite" : 2, "date" : ISODate
    ("2014-03-01T08:00:00Z") }
{"_id" : 3, "item" : "b", "prix" : 10, "quantite" : 2, "date" : ISODate
    ("2015-03-01T08:00:00Z") }
```

La requête suivante permet de donner le prix total des ventes par mois de chaque année.

```
db.test.aggregate([{$group:{_id:{ mois:{$month: "$date" },an: {$year: "
    $date" }},PrixTotal:{$sum:{$multiply: ["$prix","$quantite" ]}}}}])
```

5.2 Exercices

5.2.1 Exercice 1

On considère la collection "Hotels" d'une base "GuideTour" donnant des informations sur des hôtels de l'Algérie. La base est créée sous Mongodb, un exemple d'un document BSon de la collection est donné dans le listing 5.1

Listing 5.1: *Schéma du document de la collection Hotels*

```
{
    "_id" : ObjectId("5ab005b4511178a4ef5dd97b"),
    "nom" : "Renaissance Tlemcen",
    "adresse" : {
        "rue" : "Boulevard De Lala Sitti",
        "commune" : "Tlemcen",
        "ville" : "Tlemcen"
    },
   "groupe_hotelier" : "Marriott International",
    "nombre de chambres" : 204,
    "prix_moyen" : 22500,
    "Site internet" : "http://www.marriott.com/hotels/travel/algbr-
       renaissance-tlemcen-hotel",
    "etoiles" : 5,
    "equipements" : [
        "Parking",
        "restaurant",
        "Navette vers l'aéroport",
        "salle de sport",
        "piscine",
        "SPA multiservice"
    ],
    "description" : "Situé à 6 minutes à pied du parc de loisirs Lalla
       Setti, cet hôtel contemporain est à 7 km de la  Grande Mosquée
       de Tlemcen...",
    "annee_inauguration" : 2011,
    "avis" : [
        {  "date_publi" : ISODate(2017-12-11 00:00:00.000),
           "commentaire" : "Très bon service et accueil",
           "note" : 7  },
        {  "date_publi" : ISODate(2018- 03-06 00:00:00.000),
           "commentaire" : "Nous avons vraiment apprécié notre séjour
              ...",
           "note" : 5  },
    ]
}
```

Formuler les requêtes suivantes :

1. Rajouter la collection restaurant
2. Insérer le restaurant nommé 'Restaurant Marrakech' qui est spécialisé en cuisine traditionnelle, ce dernier a été inauguré en 2013, il propose pour l'instant deux menus : Hrira qui coute 150 da et Tagine Zitoune qui coute 350 da.
3. Récupérer la liste complète des hôtels.
4. Modifier le nombre de chambre de l'hôtel renaissance Tlemcen à 206
5. Ajouter l'équipement salle de jeux à l'hôtel renaissance
6. Liste des hôtels triés par nom et par année d'inauguration.
7. Le nom des hôtels 3 étoiles.
8. Liste des hôtels de Tlemcen inaugurés entre 2000 et 2016.

9. Liste des hôtels de Tlemcen récents (inaugurés après 2016) ou classés 4 étoiles.
10. Liste des hôtels qui contiennent une piscine.
11. Liste des hôtels qui ont une piscine et un parking.
12. Les hôtels qui ont de très bon commentaire : commentaire qui contient le mot 'bon' ou 'excellent'.
13. Augmenter le prix moyen de chambre pour les hôtels de 4 ou 5 étoiles.
14. Donner pour chaque hôtel le nombre de ses équipements.
15. Calculer la note moyenne des avis de chaque hôtel.
16. Donner la liste des hôtels pour chaque équipement.
17. Le nombre d'hôtels par équipement.

5.2.2 Exercice 2

On considère les deux collections : "Etablissement" (listing 5.2) et "Guide_Gastronomie"(listing 5.3) tirées à partir de la base du site www.guide-alger.com/ou-manger qui répertorie les bonnes adresses d'établissements gastronomiques de la wilaya d'Alger.

Listing 5.2: *Schéma de la collection "Etablissemnt"*

```
/* 1 */
{
    "_id" : ObjectId("5c5ab54ca533301748251c9a"),
      "_Id_Etablissement " : " Tacos16",
    "Nom" : "Gratinados Tacos",
    "Description" : "Gratinados Tacos sert des tacos faits maisons sous
       toutes les coutures. Medium, large, Xl pour les petits et grands
        mangeurs.  De délicieux tacos garnis de frites et de sauce
       fromagère.. Large choix de viandes et de sauces. Steak mariné,
       poulet ",
    "Adresse" : "05, bd Colonel Bougara. El Biar",
    "jours d'ouverture" : [    "samedi",  "dimanche",
           "lundi","mardi","mercredi","jeudi"     ],
    "horaires" : [   11,  01],   /*heure ouverture, fermeture*/
    "Mobile" : "05 42 49 40 92"
}
/* 2 */
{
    "_id" : ObjectId("5c5aba4ba7e8813fe6fc695d"),
    "Nom" : "D-Tox",
    "Description" : "D-Tox a ouvert ses portes il y a deux jours.
       Concept: healthy food pour succomber au pécher de gourmandise
       sans prendre un gramme et sans mettre sa santé en péril . Crédo:
        D-Tox, le péché sain. Bienvenu au slow food avec des légumes
       cuits à la vapeur ou poêlés, des salades variées, des pâtes
       faites maisons à base de blé complet, des sauces naturelles, des
        desserts sans sucre ajouté. On peut composer soi-même sa salade
        en choisissant entre 5 bases, 50 ingrédients et 8 sauces. Les
```

```
        sandwichs sont préparés à base  de pain complet, avoine, seigle)
        . ",
    "Adresse" : "10 Bd Sidi Yahia",
    "jours d'ouverture" : ["samedi",   "dimanche",       "lundi", "mardi
        ",   "jeudi",  "Vendredi"],
    "horaires" : [ 11,   23 ],
    "Mobile" : "07 71 81 83 85",
    "capacité" : 240

}
/* 3 */
{
    "_id" : ObjectId("5c5abe63a7e8813fe6fc6acb"),
    "Nom" : "Maestro",
    "Description" : "Pêché de gourmandise chez Maestro. Un nouveau
        glacier dont les glaces artisanales, 100% naturelles, à base de
        fruits et de produits frais. Tout est préparé au labo, à l'étage
        , selon une recette française jalousement gardée aux saveurs
        Pistache, chocolat noir, yaourt, tiramisu, figues, orange,
        banane.. ",
    "Adresse" : "Sidi Yahia (Rond point du haut. Côté hôtel Lalla
        Doudja)",
    "jours d'ouverture" :  ["samedi",   "dimanche", "lundi", "mardi",  "
        mercredi",  "jeudi", "Vendredi"],
 "horaires" : [ 12,   02  ],
  "Mobile" : "05 61 06 83 75"
}
```

Listing 5.3: *Schéma de la collection "GuideGastronomie"*

```
/* 1 */
{   "_id" : ObjectId("5c5ab54ca533301748251c9a"),
     "Nom_Etablissement" : "Gratinados Tacos",
     "Description" : "Gratinados Tacos sert des tacos faits maisons sous
         toutes les coutures...",
     "Type" : "fast food",
    "spécialité" : ["Sandwich Tacos"],
    "Menu" : [
        {    "Reference" : "Tacos M",
             "Prix" : 450   },
        {   "Reference" : "Tacos L",
             "Prix" : 600    }
    ]
}

/* 2 */
{    "_id" : ObjectId("5c5abe63a7e8813fe6fc6acb"),
    "Nom_Etablissement" : "Maestro",
    "Description" : "Pêché de gourmandise chez Maestro. Un nouveau
        glacier dont les glaces artisanales, 100% naturelles, à base de
        fruits .... ",
    "Type" : "Crèmerie",
    "spécialité" : ["Crème artisanale" , "Sorbets de fruits"],
    "Menu" : [
        {  "Reference" : "Cornet 1 boule",
             "Prix" : 150 },
        {    "Reference" : "Cornet 2 boules",
             "Prix" : 300  },
```

```
        {  "Reference" : "Coupe",
           "Prix" : 600  }
    ]
}
/* 3 */

{     "_id" : ObjectId("5c5aba4ba7e8813fe6fc695d"),
    "Nom_Etablissement" : "D-TOX",
     "Description" : "D-Tox a ouvert ses portes il y a deux jours. Le
        concept: healthy food permet de succomber au pécher de
        gourmandise sans prendre un gramme.... ",
    "Type" : "Restaurant",
    "spécialité" : ["Plats Healthy "],
    "Menu" : [
        {  "Reference" : "Croquettes de légume",
           "Prix" : 350   },
        {   "Reference" : "Soupe du jour",
           "Prix" : 350   },
        {  "Reference" : "Calamars façon D-Tox",
           "Prix" : 400    },
        { "Reference" : "Sandwich multivitaminé",
           "Prix" : 650    },
        {    "Reference" : "Carrot cake",
           "Prix" : 400   }
    ]
}
```

1. Rajouter la spécialité 'Sandwich' à l'établissement 'D-TOX' .
2. Donner les établissements qui restent ouverts après-minuit.
3. Donner le nom des restaurants qui ont au moins un plat inférieur à 450 da.
4. Afficher le nombre de restaurants situés à Sidi Yahia qui ont une capacité supérieure à 100 places
5. Donner, pour chaque type d'établissement, la liste (nom d'établissement). Le résultat de l'exécution de la requête est :

```
{    "_id" : "Crèmerie",
     "Liste_d'établissements" : [ "Maestro" ]
}
{    "_id" : "restaurant",
     " Liste_d'établissements" : [ "D-Tox" ]
}
...
```

6. Afficher tous les documents de la collection "Guide_Guastronomie" avec toutes les informations des établissements associés.
7. Donner la sémantique des requêtes suivantes :

   a
   ```
   db.Etablissement.find({"jours d'ouverture":{"$size":7}})
   ```

b
```
db.Etablissement.update({}, {$addToSet: {"jours d'ouverture":"
    Vendredi"}})
```

c
```
db.Etablissement.find({}).forEach(function(doc)
                {var Nom_Etablissement = doc.Nom;
                 db.Guide_Gastronomie.update({Nom_Etablissement
                     },{$set:{"info":doc}});})
```

5.3 Exercices supplémentaires

5.3.1 Exercice supplémentaire 1

On considère les deux collections : "Cinéma" et "Agenda_Ciméma" de la base "Programme-Ciné" donnant des informations sur les cinémas, les films et leur diffusion en salles de cinéma. Un exemple de documents contenus dans les deux collections est donné respectivement dans le listing 5.4 et 5.5.

Listing 5.4: *Schéma de la collection Ciméma*

```
/* 1 */
{
    "_id" : ObjectId("5af2ed8c7afa5a2e7cb6997e"),
    "nom" : "Salle Ibn Khaldoun",
    "adresse" : {
        "rue" : "12, Rue Docteur Saâdane",
        "ville" : "Alger"
    },
    "téléphone" : "021 40 30 22",
    "capacité salle" : 240
}
/* 2 */
{
"_id" : ObjectId("5af2f1497afa5a2e7cb69980"),
    "nom" : "Le Mouggar",
    "adresse" : {
        "rue" : "Rue Asselah Hocine",
        "ville" : "Alger"
    },
"téléphone" : "021 40 87 06",
 "capacité salle" : 160
}
```

Listing 5.5: *Schéma de la collection AgendaCiméma*

```
/* 1 */
{    "_id" : ObjectId("5af8a730f952dd77ff60f743"),
    "salle_cinéma" : "Salle Ibn Khaldoun",
    "film" : "TAXI 5",
    "année_sortie" : 2018,
    "mois_sortie" : "avril",
```

```
    "durée" : "1h 40mn",
    "genre" : [ "Action",  "Policier" ],
    "tarif" : 600,
    "synopsys" : "Sylvain Marot, super flic parisien et pilote d'
       exception, est muté ....",
    "programme" : [
        {   "date" : "05 avril 2018",
            "séances" : [  "10h00",  "17h00"  ]  },
        {  "date" : "07 avril 2018",
            "séances" : [ "20h00",   "21h45" ]  },
        {   "date" : "17 mai 2018",
            "séances" : [   "10h00",   "20h00"  ]  }
    ]
}
/* 2 */
{   "_id" : ObjectId("3ef978a4c755e06f72118e19"),
    "salle_cinéma" : "Salle El Mouggar",
    "film" : "TAXI 5",
    "année_sortie" : 2018,
    "mois_sortie" : "avril",
    "durée" : "1h 40mn",
    "genre" : [  "Action",  "Policier"  ],
    "tarif" : 700,
    "synopsys" : "Sylvain Marot, super flic parisien et pilote d'
       exception, est muté ...",
    "programme" : [
        {   "date" : " 08 mai 2018",
            "séances" : [ "13h45",  "20h00" ]   },
        {  "date" : "17 mai 2018",
            "séances" : [  "20h00",  "21h45" ]  }
    ]
}
/* 3 */
{
    "_id" : ObjectId("5af8a792f952dd77ff60f78d"),
    "salle_cinéma" : "Salle Ibn Khaldoun",
    "film" : "Tomb Raider",
    "année_sortie_salle" : 2018,
    "mois_sortie_salle" : "mars",
    "genre" : [  "Action" ],
    "synopsys" : "Lara Croft, 21 ans, n'a ni projet, ni ambition..."
    "programme" : [
        {   "date" : "29 mai 2018",
            "séances" : [  "17h00" ] }
    ]
}
```

1. Rajouter le genre "Comédie" au film "TAXI 5" .
2. Afficher le nombre de salles de cinéma situées à Alger qui ont une capacité de salle supérieure à 200 places
3. Modifier le nom de l'attribut : "année_sortie_salle" à "année_sortie".
4. Afficher le nom des salles où passe un film d'action policier le 05 avril 2018.
5. Donne pour chaque cinéma la liste de tous les films diffusés. Le résultat de l'exécution de la

requête est :

```
/* 1 */
{    "_id" : "Salle El Mouggar",
     "Liste_films" : [ "TAXI 5" ]
}
/* 2 */
{    "_id" : "Salle Ibn Khaldoun",
     "Liste_films" : [ "TAXI 5",  "Tomb Raider" ]
}
```

6. On voudrait rajouter au document Agenda_Cinéma, les informations de chaque cinéma dans le champ 'information_ciné'. Donnez la requête adéquate.

7. Expliquer les requêtes suivantes et donner le résultat obtenu sur la base :

a
```
db.Agenda_Cinema.update({"film":"TAXI 5"}, {$addToSet: {"
    programme.0.séances":"17h00"}})
```

b
```
db.Agenda_Cinema.find({"salle_cinéma":"Salle Ibn Khaldoun"}).
    forEach(function(doc)
{db.Agenda_Cinema.update({_id: doc._id},{$set:{"tarif": doc.
    tarif + 100 }});})
```

c
```
db. Agenda_Cinema.aggregate([{ $group : { _id : "$film",
    Liste_cinema:{$push:"$salle_cinéma"}}}, {$out:"Films"}]);
```

5.3.2 Exercice supplémentaire 2

Soit la collection "sallesDeSport" de la base "guide sportif" donnant des informations sur les salles de sport de Tlemcen et leurs programmes

Listing 5.6: *Schéma de la collection sallesDeSport*

```
{
 "_id" : ObjectId("5a99a676f7b45ad721c49fb7"),
 "salle_sport" : "30075445",
 "nom" : "Club Nour ",
 "spécialité" : "fitness",
 "année_inauguration" : 2013,
 "adresse" : {
     "Rue" : "Essalam",
     "CodePostal" : "13100",
     "quartier" : "imama"
 },

 "programme" : [
     {
         "séance" : "aerobic",
         "prix" : "600 da"
     },
     {
```

```
            "séance" : "Zumba",
            "prix" : "450 da"
        },
        {
            "séance" : "aquagym",
            "prix" : "500 da"
        }

    ],
Equipements :[ piscine, vélo, tapis ],
    "Avis" : [
        {
            "date" : "2018-03-03",
            "note" : 2
        },
        {
            "date" : "2017-01-24",
            "note" : 10
        }
    ]
}
```

Donnez les requêtes qui permettent de

1. Retourner la liste des salles de sport triée par nom décroissant.
2. Rajoutez au programme de la salle 'Club Nour' la séance 'Step' qui coûte 350 da.
3. Donner le nom de la salle qui a une piscine.
4. Donner La liste des salles inaugurées entre 2009 et 2017.
5. Donner pour chaque salle le prix moyen d'une séance.
6. Donner pour chaque quartier le nombre et la liste de toutes les salles.
7. Donner l'équivalent de cette requête en utilisant la fonction forEach()

```
db.salles.update({"année_inauguration":{$in:[2017,2018]}},{$set:{"
    commentaire":"restaurant récent" }})
```

5.4 Solutions d'exercices

5.4.1 Solution de l'exercice 1

1. Rajouter la collection restaurant

```
db.createCollection("restaurant") ;
```

2. Insérer le restaurant nommé 'Restaurant Marrakech' qui est spécialisé en cuisine traditionnelle, ce dernier a été inauguré en 2013, il propose pour l'instant deux menus : Hrira qui coute 150 da et Tagine Zitoune qui coute 350 da.

```
db.restaurant.insert({"nom" : "Restaurant Marrakech",
    "spécialité" : "traditionnel",
    "année_inauguration" : 2013,
    "menu" : [
        {
            "plat" : "Hrira",
            "prix" : "150 da"
        },
        {
            "plat" : "Tagine Zitoun",
            "prix" : "350 da"
        }
    ]})
```

3. Récupérer la liste complète des hôtels.

```
db.hotels.find()
```

4. Modifier le nombre de chambre de l'hôtel renaissance Tlemcen à 206

```
db.hotels.update({nom:"Renaissance Tlemcen"},{$set:{"nombre de
    chambres":206}})
```

5. Ajouter l'équipement salle de jeux à l'hôtel renaissance

```
db.hotels.update({nom:"Renaissance Tlemcen"},{$push:{equipements:"
    salle de jeux"}})
```

NB : si on veut rajouter à la fin du tableau sans doublon, il faudra utiliser l'opérateur $addToSet

6. Liste des hôtels triés par nom et par année d'inauguration.

```
db.Hotels.find({}).sort({"nom":1, annee_inauguration:1})
```

7. Le nom des hôtels 3 étoiles.

```
db.Hotels.find({etoiles:3},{"nom":1,_id:0})
```

8. Liste des hôtels de Tlemcen inaugurés entre 2000 et 2016.

- Solution avec $and implicite

```
db.Hotels.find({"adresse.ville":"Tlemcen", annee_inauguration
    :{$gte:2000,$lte:2016}})
```

Attention : Si les deux tests numériques sur les mêmes attributs ne sont pas regroupés, alors MongoDB ne retiendra que le résultat du dernier opérateur logique. Dans notre exemple le retour se fera sur la dernière condition c.à.d on aura tous les hôtels < 2016 , même ceux inaugurés avant 2000.

- Solution avec $and explicite

```
db.Hotels.find({$and:[{"adresse.ville":"Tlemcen"},      {
    annee_inauguration:{$gte:2009}},
{annee_inauguration:{$lte:2016}}]})
```

9. Liste des hôtels de Tlemcen récents (inaugurés après 2016) ou classés 4 étoiles.

```
db.Hotels.find({$and:[
                        {"adresse.ville":"Tlemcen"},
                         {$or:[
                                {annee_inauguration:{$gt:2016}},
                                {etoiles:4}
                               ]
                         }
                        ]
                })
```

Deuxième solution :

```
db.Hotels.find({"adresse.ville":"Tlemcen",$or:[{annee_inauguration
    :{$gt:2016}},{etoiles:3} ]})
```

10. Liste des hôtels qui contiennent une piscine.

```
db.Hotels.find({equipements:"piscine"})
```

11. Liste des hôtels qui ont une piscine et un parking.

```
db.Hotels.find({equipements:{$all:["piscine","Parking"]}})
```

12. Les hôtels qui ont de très bon commentaire : commentaire qui contient le mot 'bon' ou 'excellent'.

```
db.Hotels.find({$or:[{"avis.commentaire":{$regex:"bon",$options:"i
    "}},{"avis.commentaire":{$regex:"apprécié",$options:"i"}}]})
```

Nous pouvons aussi utiliser le caractère qui exprime le choix (ou logique)

```
db.hotels.find({"avis.commentaire":{$regex:/bon|exelent/, $options
    :"i"}})
```

Ou encore utiliser directement les pattern sans le $regex

```
db.hotels.find({"avis.commentaire":/bon|exelent/i})
```

13. Augmenter le prix moyen de chambre pour les hôtels de 4 ou 5 étoiles.

```
db.hotels.update({"etoiles":{"$in":[4,5]}},{"$mul":{"prix_moyen
    ":1.20} (augmenter de 20%)});
```

En utilisant le forEach

```
db.hotels.find({"etoiles":{$in:[4,5]}}).forEach(function(doc)
{var updatedprix = doc.prix_moyen + (doc.prix_moyen * 10/100);
    db.Hotels.update({_id: doc._id},{$set:{"prix_moyen":
        updatedprix }});})
```

14. Donner, pour chaque hôtel, le nombre de ses équipements.

```
db.hotels.aggregate([{$project:{ Hotel:"$nom", "nbequipement":{
    $size:"$equipements"}, _id:0}}])
```

On n'a pas besoin du groupe car on a un seul hôtel par collection, néanmoins si on veut utiliser le $groupe il faudra éclater le tableau équipement

```
db.hotels.aggregate([{$unwind: "$equipements" },{$group:{_id:"$nom
    ","nbequipement":{$sum:1}}}])
```

15. Calculer La note moyenne des avis de chaque hôtel.id.

```
db.hotels.aggregate([{$project:{_id:"$nom","notemoy":{$moy:"$avis.
    note"}}}])
```

Avec $group

```
db.hotels.aggregate([{ $unwind: "$avis" },{$group:{_id: "$nom","
    notemoy":{$avg:"$avis.note"}}}])
```

16. la liste des hôtels pour chaque équipement

```
db.hotels.aggregate([{ $unwind: "$equipements" },{$group:{_id: "
    $equipements", hotels: {$push: "$nom"}}} ])
```

17. Le nombre d'hôtels par équipement

```
db.hotels.aggregate([{ $unwind: "$equipements" },{$group:{_id: "
    $equipements", nombre:{$sum:1} }} ])
```

5.4.2 Solution de l'exercice 2

1. Rajouter la spécialité 'Sandwich' à l'établissement 'D-TOX'.

```
db.Guide_Gastronomie.update({"Nom_Etablissement":"D–TOX"},{$push:{"
    spécialité":"Sandwich"}})
```

2. Donner les établissements qui restent ouvert après-minuit.

```
db.Etablissement.find({"horaires.1":{$gt:00, $lt:11}})
```

3. Donner le nom des restaurants qui ont au moins un plat inférieur à 450 da.

```
db.Guide_Gastronomie.find({"Type":"Restaurant", "Menu.Prix":{$lte
    :450}}, {nom :1,_id :-1})
```

4. Afficher le nombre de restaurants situés à Sidi Yahia qui ont une capacité supérieure à 100 places.

```
db.Etablissement.find({"Adresse":{$regex:"Sidi Yahia", $options:"i
    "}}).count()
```

5. Donner, pour chaque type d'établissement, la liste (nom d'établissement).

```
db.Guide_Gastronomie.aggregate([{ $group : { _id : "$Type",
    Liste_Etablissement :{$push:"$Nom_Etablissement"}}}]);
```

6. Afficher tous les documents de la collection "Guide_Guastronomie" avec toutes les informations des établissements associés

```
db.Guide_Gastronomie.aggregate([ { $lookup:
{ from: "Etablissement",
   localField: "Nom_Etablissement",
   foreignField: "Nom",
   as: "information_Etablissement"
} } ])
```

7. La sémantique des requêtes :

   a) La requête permet de retourner les salles qui ouvrent tous les jours de la semaine. On suppose qu'il n'y a pas de redondance. Car syntaxiquement la requête retourne les salles dont la taille du tableau "jours d'ouverture" est égale à 7.

   b) La requête va écraser le tableau "jours d'ouverture " avec le Vendredi. Si on veut ajouter à la fin du tableau le Vendredi, au cas où il n'existe pas parmi les "jours d'ouverture", il faudra utiliser $addToset.

   c) La requête permet d'ajouter à chaque établissement le champ "info" qui va contenir tous les champs du document correspondant à celui de la collection "Guide_Guastronomie". C'est l'équivalent d'une jointure avec l'opérateur $lookup faite sur le champ "nom_Etablissement" entre la collection "Guide_guastronomie" et "Etablissement".

5.5 Solution des exercices supplémentaires

5.5.1 Solution de l'exercice supplémentaire 1

1. Rajouter le genre "Comédie" au film "TAXI 5".

```
db.Agenda_Cinema.update({"film":"TAXI 5"},{$push:{"genre":"Comédie
    "}},{multi:1})
```

2. Afficher le nombre de salles de cinéma situées à Alger qui ont une capacité de salle supérieure à 200 places.

```
db.Cinema.find({"adresse.ville": "Alger","capacité salle": {$gte
    :200}}).count()
```

3. Modifier le nom de l'attribut : "année_sortie_salle" à "année_sortie".

```
db.Agenda_cinema.updateMany({},{$rename:{"année_sortie_salle": "ann
    ée_sortie"}})
```

4. Afficher Le nom des salles où passe un film d'action policier le 05 avril 2018.

```
db.Agenda_Cinema.find({"genre":{$all:["Action", "Policier"]}, "
    programme.date":"05 avril 2018"},{salle_cinéma:1, _id:0})
```

5. Donner pour chaque cinéma la liste de tous les films diffusés.

```
db.Agenda_Cinema.aggregate([{ $group : { _id : "$salle_cinéma",
    Liste_films:{$push:"$film"}}}]);
```

6. On voudrait rajouter, en affichage, au document Agenda_Cinéma, les informations de chaque cinéma dans le champ 'information_ciné'. Donner la requête adéquate.

   Première solution :

```
db.Agenda_Cinema.aggregate([ { $lookup:
{ from: "Cinema",
   localField: "salle_cinéma",
   foreignField: "nom",
   as: "information_ciné"
} } ])
```

   Deuxième solution :

```
db.Cinema.find({}).forEach(function(doc)
{var salle_cinéma = doc.nom;
    db.Agenda_Cinema.update({salle_cinéma},{$set:{"info":doc}});})
```

7. Expliquer les requêtes :

   a La requête permet de rajouter une séance de 17h à la première date de diffusion (programme.0). Cette séance est ajoutée seulement si elle n'existe pas, car l'opérateur "$addToSSet" n'admet pas les doublons. La séance ne sera rajoutée qu'au premier film Taxi5 de la base car il n'y a pas l'opérateur "$multi"

   - Le résultat retourné est :

   Au niveau du document rien à ajouter car la séance existe déjà.

   b La requête permet de rajouter 100 da au tarif des films diffusés à la salle "ibn khaldoun".

   - Le résultat retourné est :

   Pour le premier film de la salle "Ibn Khaldoun", le tarif a été modifié ("tarif" : 700).

   Pour le deuxième film, l'attribut "tarif" n'existait pas, alors un attribut a été rajouté avec une valeur vide ! ("tarif" : NAV). Pour éviter ce genre de résultat, il faudra tester l'existence du champ avant d'appliquer la modification.

   c La requête permet de créer et de retourner la collection film, cette dernière affiche pour chaque film la liste des cinémas où ils sont diffusés.

   - Le résultat retourné est la collection "Film" :

```
/* 1 */
{
    "_id" : "Tomb Raider",
    "Liste_cinema" : [ "Salle Ibn Khaldoun" ]
}
/* 2 */
{
    "_id" : "TAXI 5",
    "Liste_cinema" : [ "Salle Ibn Khaldoun", "Salle El
       Mouggar" ]
}
```

5.5.2 Solution de l'exercice supplémentaire 2

1. Retourner la liste des salles de sport triée par nom décroissant.

```
db.salles.find({}).sort({"nom":1})
```

2. Rajouter au programme de la salle 'Club Nour' la séance 'Step' qui coûte 350 da.

```
db.salles.update({"nom":" Club Nour "},{$addToSet:{"programme":{"sé
    ance" : "step", "prix" : "350 da"}}}) (on peut aussi utiliser
    push)
```

3. Donner le nom de la salle qui a une piscine.

```
db.salles.find({"Equipements": "piscine"})
```

4. Donner la liste des salles inaugurées entre 2009 et 2017.

```
db.salles.find({ annee_inauguration:{$gte:2000,$lte:2017}},{nom:1,
    _id:0})
```

5. Donner, pour chaque salle, le prix moyen d'une séance.

```
db.salles.aggregate([{ $project:{_id:"$nom",
                                 "prixmoy":{$avg:"$programme.prix"} }}
                                     ])
```

6. Donner, pour chaque quartier, le nombre et la liste de toutes les salles.

```
db.salles.aggregate([{ $group : { _id : "$adresse.quartier", total:
    {$sum:1}, Liste_salles:{$push:"$nom"}}}]);
```

7. Donner l'équivalent de la requête en utilisant la fonction forEach()

```
db.salles.find({"année_inauguration":{$in:[2017,2018]}}).forEach(
    function(doc)
{db.salles.update({_id: doc._id},{$set:{"commentaire": "salle ré
    cente" }});})
```

Bibliographie & Webographie

[1] George Gardarin. *Bases de données*. Eyrolles, 2003.

[2] Christian Soutou. *SQL pour Oracle*. Editions Eyrolles, 2013.

[3] Jean-Luc Hainaut. *Bases de données : concepts, utilisation et développement*. Editions DUNOD, 2012.

[4] Claude Chrisment, Guillaume Cabanac, Karen Pinel-Sauvagnat, Olivier Teste, and Michel Tuffery. *Bases de données orientées-objet : Concepts, mise en oeuvre et exercices résolus*. Lavoisier, 2011.

[5] Rudi Bruchez. *Les bases de données NoSQL et le BigData : Comprendre et mettre en oeuvre*. Editions Eyrolles, 2015.

[6] Derrick Harris. 10gen embraces what it created, becomes mongodb inc, 2013.

[7] Mongo manual. https ://docs.mongodb.com/v4.0/release-notes/4.0/, année d'accées : 2020.

[8] Mongo templates. https ://docs.mongodb.com/v4.0/core/data-modeling-introduction/, année d'accées : 2020.

[9] Philippe Lacomme, Raksmey Phan, and Sabeur Aridhi. *Bases de données NoSQL et Big Data : concevoir des bases de données pour le Big Data : cours et travaux pratiques niveau B*. Ellipses Marketing, 2014.

[10] Jeffrey Dean and Sanjay Ghemawat. Mapreduce : simplified data processing on large clusters. *Communications of the ACM*, 51(1) :107–113, 2008.

[11] Mongodb aggregation framework. https ://docs.mongodb.com/manual/reference/operator/aggregation/, année d'accèes : 2020.

Ce document a été préparé à l'aide de l'éditeur Texmaker 5.0.4: un logiciel libre sous la licence open source GNU GPL, qui se présente sous la forme d'un environnement de développement intégré pour le langage LaTeX sous Windows.
Il est basé sur les distributions TeXLive 2019

FACULTÉ DES SCIENCES

Faculty of Sciences

كلية العلوم

Faculté des Sciences – Tidjani HADDAM

Tél.: 043 21 63 70 / Tél &Fax: 043 21 63 68 / 043 21 63 71

Site Web: www.fs.univ-tlemcen.dz

Email : vdrpg.facscience@gmail.com