

République Algérienne Démocratique et Populaire  
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique



Université des Sciences et de la Technologie Houari Boumédiène

Faculté d'Informatique  
Département Informatique

Master Systèmes Informatiques intelligents

Module : Bases de Données avancées

---

## Rapport de projet BDA

---

Réalisé par :

BOUCHOUL Bouchra, 191931081317  
MEKKAOUI Mohamed, 191931081338

Année universitaire : 2022 / 2023

# Table des matières

<b>1</b>	<b>Partie 1 : SQL3-Oracle</b>	<b>3</b>
1.1	Modélisation orientée objet . . . . .	3
1.1.1	Diagramme de classe : . . . . .	3
1.2	Création des TableSpaces et utilisateur . . . . .	4
1.2.1	Création des TableSpaces : . . . . .	4
1.2.2	Création de l'utilisateur SQL3 : . . . . .	5
1.2.3	Attribution des privilèges pour l'utilisateur SQL3 : : . . . . .	5
1.3	Langage de définition de données . . . . .	5
1.3.1	Définition des types nécessaires : . . . . .	5
1.3.2	Définition des méthodes : . . . . .	8
1.3.3	Définition des tables nécessaires : . . . . .	11
1.4	Langage de manipulation de données . . . . .	12
1.4.1	Remplissage des tables : . . . . .	12
1.5	Langage d'interrogation de données . . . . .	16
1.5.1	Lister les noms d'hôtels et leurs villes respectives : . . . . .	16
1.5.2	Lister les hôtels sur lesquels porte au moins une réservation. . . . .	16
1.5.3	Quels sont les clients qui ont toujours séjourné au premier étage? . . . . .	17
1.5.4	Quels sont les hôtels (nom, ville) qui offrent des suites? et donner le prix pour chaque suite . . . . .	17
1.5.5	Quel est le type de chambre le plus réservé habituellement, pour chaque hôtel d'Alger? . . . . .	17
1.5.6	Quels sont les hôtels (nom, ville) ayant obtenu une moyenne de notes $\geq 6$ , durant l'année 2022 . . . . .	18
1.5.7	Quel est l'hôtel ayant réalisé le meilleur chiffre d'affaire durant l'été 2022 (juin, juillet, aout . . . . .	19
<b>2</b>	<b>Partie 2 : NoSQL – Modèle orienté « documents »</b>	<b>21</b>
2.1	Modélisation orientée document . . . . .	21
2.1.1	La proposition d'une modélisation orientée document . . . . .	21
2.1.2	l'illustration des exemples de la BD fournie . . . . .	23
2.1.3	Conception 1 : . . . . .	23
2.1.4	Conception 2 : . . . . .	24
2.1.5	Justification de choix de conception . . . . .	25
2.2	Remplissage de la base de données . . . . .	26
2.2.1	Conception 1 : . . . . .	26
2.2.2	Conception 2 : . . . . .	27
2.3	Les requetes . . . . .	28

2.3.1	Affichage de tous les hôtels classés « 3 étoiles » . . . . .	28
2.3.2	Récupérer dans une nouvelle collection Hotels-NbResv, les noms des hôtels et le nombre total de réservations par hôtel ; la collection devra être ordonnée par ordre décroissant du nombre de réservations. . . . .	29
2.3.3	Dans une collection HotelsPas-cher, récupérer les hôtels dont le prix des chambres ne dépasse pas 6000 DA. . . . .	32
2.3.4	Afficher tous les noms d'hôtels ayant obtenu une note moyenne $\geq 5$ . . . .	32
2.3.5	Afficher toutes les réservations d'un client donné (on donnera l'e-mail du client). On affichera le nom de l'hôtel, le numéro de chambre et la date d'arrivée. . . . .	34
2.3.6	Afficher toutes les évaluations postées par un client donné (on donnera l'e-mail du client). On affichera le nom de l'hôtel, la date d'évaluation, la note. .	36
2.3.7	Augmenter de 2000DA, le prix unitaire de toutes les chambres des hôtels classés « 5 étoiles » . . . . .	38
2.3.8	la 2ème requête à l'aide du paradigme Map-Reduce . . . . .	38
2.4	Analyse . . . . .	41

# Chapitre 1

## Partie 1 : SQL3-Oracle

### 1.1 Modélisation orientée objet

#### 1.1.1 Diagramme de classe :

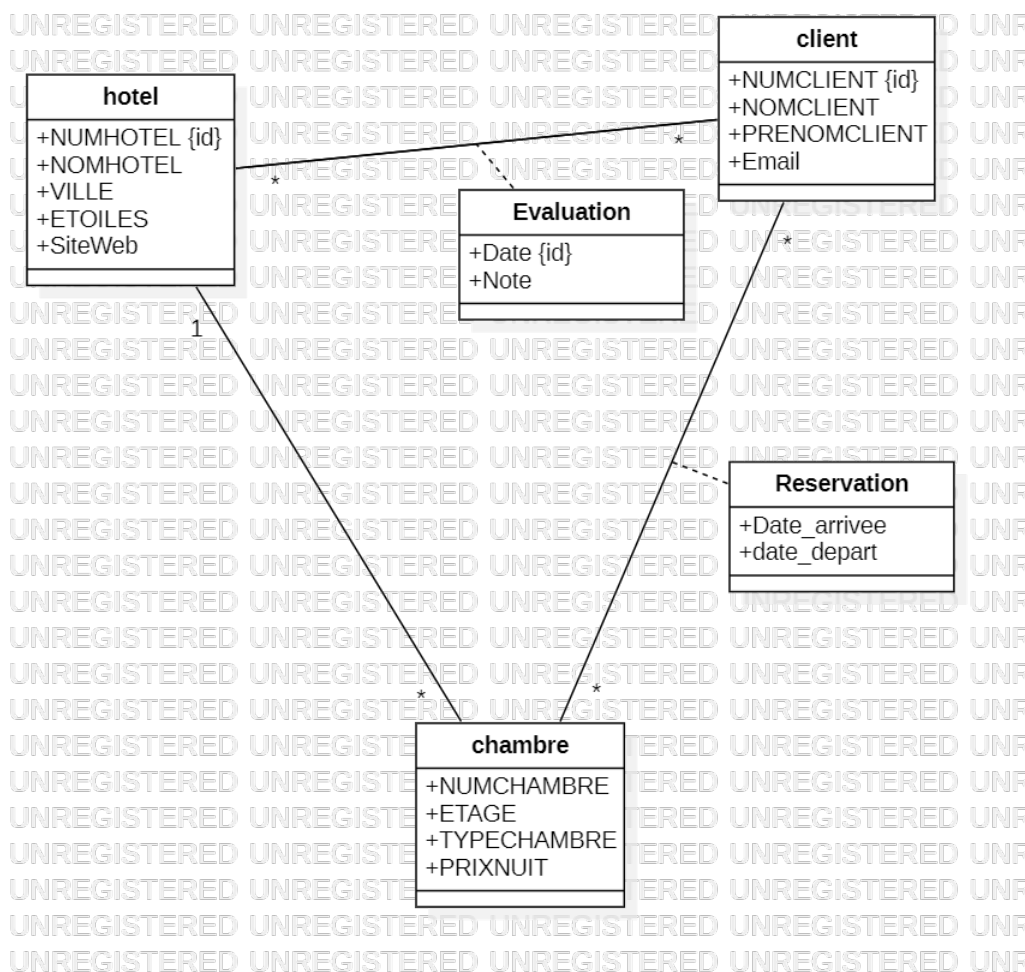
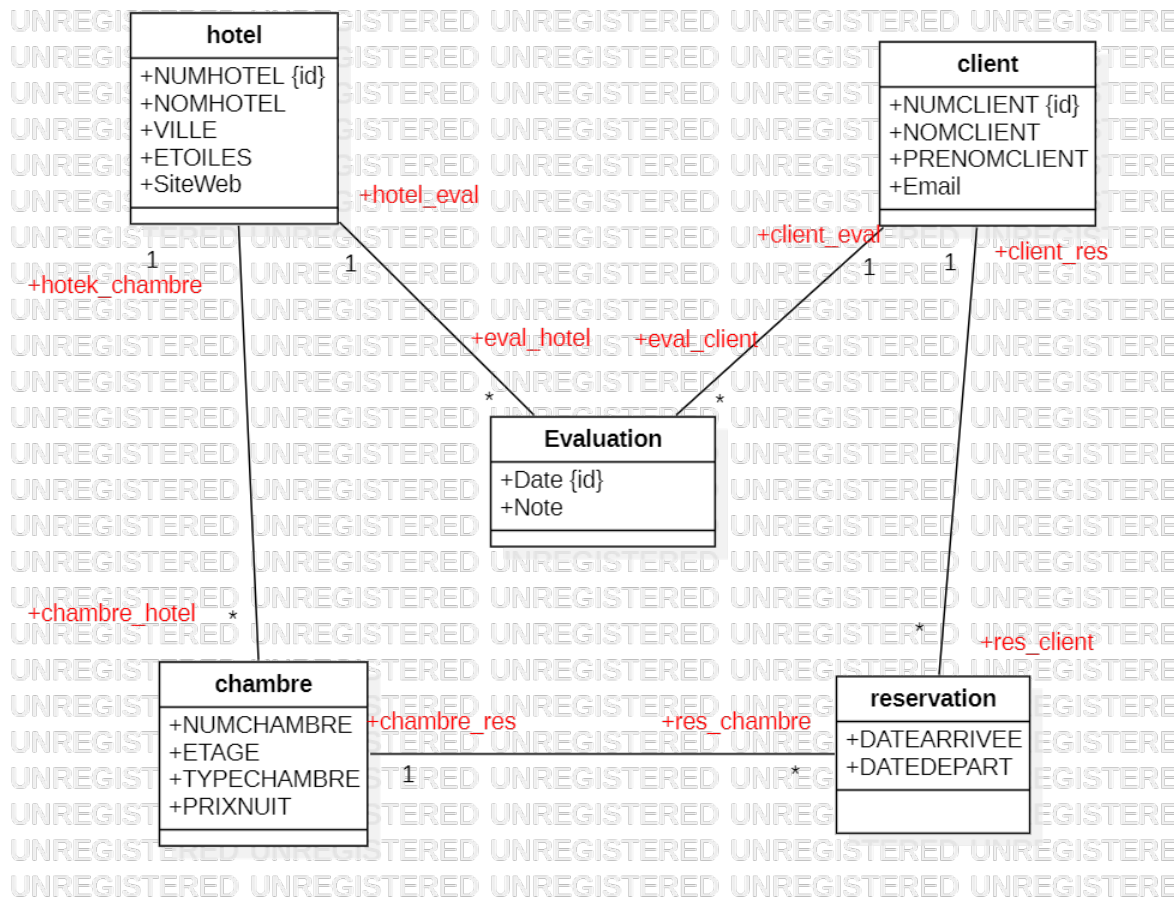


FIGURE 1.1 – Diagramme de classe

## Transformation des associations :



## 1.2 Création des TableSpaces et utilisateur

### 1.2.1 Création des TableSpaces :

Il est nécessaire de créer un espace de travail constitué d'une tablespace et d'une tablespace temporaire. Lors de la création, il est important de définir le nom, le chemin et la taille de chacune de ces tablespaces.

Requete :

```
CREATE TableSpace SQL3TBS DATAFILE 'C:\SQL3TBS.dat' SIZE 100M AUTOEXTEND ON ONLINE;
```

```
CREATE TEMPORARY TABLESPACE SQL3TempTBS TEMPFILE 'C:\SQL3TempTBS.dat' SIZE 100M  
AUTOEXTEND ON;
```

Resultat :

```
SQL> CREATE TableSpace SQL3TBS DATAFILE 'C:\SQL3TBS.dat' SIZE 100M AUTOEXTEND ON ONLINE;
Tablespace cr   .

SQL> CREATE TEMPORARY TABLESPACE SQL3TempTBS TEMPFILE 'C:\SQL3TempTBS.dat' SIZE 100M AUTOEXTEND ON;
Tablespace cr   .
```

## 1.2.2 Cr  ation de l'utilisateur SQL3 :

Requete :

```
Create user SQL3 identified by psw Default Tablespace SQL3TBS Temporary Tablespace
SQL3TempTBS ;
```

Resultat :

```
SQL> Create user SQL3 identified by psw Default Tablespace SQL3TBS Temporary Tablespace SQL3TempTBS ;
Utilisateur cr   .
```

## 1.2.3 Attribution des privil  ges pour l'utilisateur SQL3 :

Requete :

```
GRANT ALL PRIVILEGES TO SQL3;
```

Resultat :

```
SQL> GRANT ALL PRIVILEGES TO SQL3;
Autorisation de privil  ges (GRANT) accept  e.
```

## 1.3 Langage de d  finition de donn  es

### 1.3.1 D  finition des types n  cessaires :

Requete :

```
Create type thotel;
/
Create type tchambre;
/
Create type tclient;
/
Create type treservation;
/
Create type tevaluation;
```

/

**Resultat :**

```
SQL> Create type thotel;  
2 /  
Type cr   .  
SQL> Create type tchambre;  
2 /  
Type cr   .  
SQL> Create type tclient;  
2 /  
Type cr   .  
SQL> Create type treservation;  
2 /  
Type cr   .  
SQL> Create type tevaluation;  
2 /  
Type cr   .
```

**Cr  ation des tables de r  f  rences :**

**Requete :**

```
create or replace type t_set_ref_chambre as table of ref tchambre;  
/  
create or replace type t_set_ref_evaluation as table of ref tevaluation;  
/  
create or replace type t_set_ref_reservation as table of ref treservation;  
/
```

**Resultat :**

```
SQL> create or replace type t_set_ref_chambre as table of ref tchambre;  
2 /  
Type cr   .  
SQL> create or replace type t_set_ref_evaluation as table of ref tevaluation;  
2 /  
Type cr   .  
SQL> create or replace type t_set_ref_reservation as table of ref treservation;  
2 /  
Type cr   .
```

**Cr  ation des types finaux :**

**Requete :**

```
Create or replace type thotel as object (  
    NUMHOTEL INTEGER,
```

```

    NOMHOTEL VARCHAR2(50),
    VILLE VARCHAR2(50),
    ETOILES INTEGER,
    SiteWeb VARCHAR2(50),
    hotel_chambre t_set_ref_chambre,
    hotel_evaluation t_set_ref_evaluation
);
/

Create or replace type tchambre as object(
    NUMCHAMBRE INTEGER,
    NUMHOTEL INTEGER,
    ETAGE INTEGER,
    TYPECHAMBRE VARCHAR2(30),
    PRIXNUIT INTEGER,
    chambre_hotel ref thotel,
    chambre_res t_set_ref_reservation);
/

Create or replace type tclient as object(
    NUMCLIENT INTEGER,
    NOMCLIENT VARCHAR2(50),
    PRENOMCLIENT VARCHAR2(50),
    Email VARCHAR2(50),
    client_res t_set_ref_reservation,
    client_eval t_set_ref_evaluation);
/

Create or replace type treservation as object(
    NUMRES INTEGER,
    DATEARRIVEE DATE,
    DATEDEPART DATE,
    res_client ref tclient,
    res_chambre ref tchambre
);
/

Create or replace type tevaluation as object(
    NUMEVAL INTEGER,
    datee DATE,
    Note INTEGER,
    eval_client ref tclient,

```



```

    eval_hotel ref thotel
);
/

```

Resultat :

```

SQL> Create or replace type thotel as object (
2     NUMHOTEL INTEGER,
3     NOMHOTEL VARCHAR2(50),
4     VILLE VARCHAR2(50),
5     ETOILES INTEGER,
6     SiteWeb VARCHAR2(50),
7     hotel_chambre t_set_ref_chambre,
8     hotel_evaluation t_set_ref_evaluation
9 );
10 /

Type cr   .

SQL> Create or replace type tchambre as object(
2     NUMCHAMBRE INTEGER,
3     NUMHOTEL INTEGER,
4     ETAGE INTEGER,
5     TYPECHAMBRE VARCHAR2(30),
6     PRIXNUIT INTEGER,
7     chambre_hotel ref thotel,
8     chambre_res t_set_ref_reservation);
9 /

Type cr   .

SQL> Create or replace type tclient as object(
2     NUMCLIENT INTEGER,
3     NOMCLIENT VARCHAR2(50),
4     PRENOMCLIENT VARCHAR2(50),
5     Email VARCHAR2(50),
6     client_res t_set_ref_reservation,
7     client_eval t_set_ref_evaluation);
8 /

Type cr   .

```

```

SQL> Create or replace type treservation as object(
2     NUMRES INTEGER,
3     DATEARRIVEE DATE,
4     DATEDEPART DATE,
5     res_client ref tclient,
6     res_chambre ref tchambre
7 );
8 /

Type cr   .

SQL> Create or replace type tevaluation as object(
2     NUMEVAL INTEGER,
3     datee DATE,
4     Note INTEGER,
5     eval_client ref tclient,
6     eval_hotel ref thotel
7 );
8 /

Type cr   .

```

### 1.3.2 D  finition des m  thodes :

- Pour calculer pour chaque client, le nombre de r  servations effectu  es.

Requete :

```

alter type tclient add member function nbr_res_client return INTEGER CASCADE;

Create or replace type body tclient as member function nbr_res_client return INTEGER
    is
nb INTEGER;
BEGIN
select count(distinct t.column_value) INTO nb from table(self.client_res) t;
return nb;
END nbr_res_client;
END;
/

```

**Resultat :**

```

SQL> alter type tclient add member function nbr_res_client return INTEGER CASCADE;
Type modifi .

SQL>
SQL> Create or replace type body tclient as member function nbr_res_client return INTEGER is
2  nb INTEGER;
3  BEGIN
4  select count(distinct t.column_value) INTO nb from table(self.client_res) t;
5  return nb;
6  END nbr_res_client;
7  END;
8  /
Corps de type cr  .

```

- Pour calculer pour chaque h tel, le nombre de chambres.
- pour calculer pour chaque h tel, le nombre d' valuations re ues   une date donn e

**Requete :**

```

alter type thotel add member function nombre_eval(d DATE) return INTEGER CASCADE;
ALTER TYPE THOTEL ADD MEMBER FUNCTION NBR_CHAMBRE RETURN INTEGER CASCADE;
Create or replace type body thotel as
member function nombre_eval(d DATE) return INTEGER is
nbeval INTEGER;
BEGIN
SELECT count(distinct t.column_value) INTO nbeval
FROM table(self.hotel_evaluation) t where deref(t.column_value).datee=d;
return nbeval;
END nombre_eval;
MEMBER FUNCTION NBR_CHAMBRE RETURN INTEGER IS
NB INTEGER;
BEGIN

```

```

SELECT
    COUNT(DISTINCT T.COLUMN_VALUE) INTO NB
FROM
    TABLE(SELf.HOTEL_CHAMBRE) T;
RETURN NB;
END NBR_CHAMBRE;
END;
/

```

Resultat :

```

SQL> alter type thotel add member function nombre_eval(d DATE) return INTEGER CASCADE;
Type altered.

SQL> ALTER TYPE THOTEL ADD MEMBER FUNCTION NBR_CHAMBRE RETURN INTEGER CASCADE;
Type altered.

SQL> Create or replace type body thotel as
2 member function nombre_eval(d DATE) return INTEGER is
3 nbeval INTEGER;
4 BEGIN
5 SELECT count(distinct t.column_value) INTO nbeval
6 FROM table(self.hotel_evaluation) t where deref(t.column_value).date=d;
7 return nbeval;
8 END nombre_eval;
9 MEMBER FUNCTION NBR_CHAMBRE RETURN INTEGER IS
10 NB INTEGER;
11 BEGIN
12 SELECT
13     COUNT(DISTINCT T.COLUMN_VALUE) INTO NB
14 FROM
15     TABLE(SELf.HOTEL_CHAMBRE) T;
16 RETURN NB;
17 END NBR_CHAMBRE;
18 END;
19 /

Type body created.

SQL> |

```

- pour calculer pour chaque chambre, son chiffre d'affaire.

```

alter type tchambre add member function calcul_chiffre return INTEGER CASCADE;

CREATE OR REPLACE TYPE BODY tchambre AS MEMBER FUNCTION calcul_chiffre RETURN INTEGER
IS
    ch_aff INTEGER;
BEGIN
    SELECT count(distinct t.column_value) INTO ch_aff FROM table(self.chambre_res) t;

    RETURN ch_aff*self.PRIXNUIT;

```

```

    END calcul_chiffre;
END;
/

```

Resultat :

```

SQL> alter type tchambre add member function calcul_chiffre return INTEGER CASCADE;
Type modifi  .

SQL>
SQL> CREATE OR REPLACE TYPE BODY tchambre AS MEMBER FUNCTION calcul_chiffre RETURN INTEGER IS
2   ch_aff INTEGER;
3   BEGIN
4       SELECT count(distinct t.column_value) INTO ch_aff FROM table(self.chambre_res) t;
5
6
7       RETURN ch_aff*self.PRIXNUIT;
8   END calcul_chiffre;
9   END;
10  /
corps de type cr    .

```

### 1.3.3 D  finition des tables n  cessaires :

Requete :

```

CREATE TABLE Hotel OF thotel(PRIMARY KEY(NUMHOTEL),CONSTRAINT etoiles_ck CHECK
    (etoiles BETWEEN 0 AND 5))
NESTED TABLE hotel_chambre store as table_hotel_chambre,
NESTED TABLE hotel_evaluation store as table_hotel_evaluation;

CREATE TABLE Chambre OF tchambre(PRIMARY KEY(NUMCHAMBRE,NUMHOTEL), FOREIGN
    KEY(chambre_hotel) REFERENCES Hotel,
check(TYPECHAMBRE in ('simple', 'double', 'triple','suite','autre')) )
NESTED TABLE chambre_res store as table_chambre_res;

Create table client of tclient(PRIMARY KEY(NUMCLIENT))
NESTED TABLE client_eval store as table_client_eval,
NESTED TABLE client_res store as table_client_res;

Create table evaluation of tevaluation(PRIMARY KEY(NUMEVAL),FOREIGN KEY(eval_client)
    REFERENCES client,
FOREIGN KEY(eval_hotel) REFERENCES Hotel,CONSTRAINT noteck CHECK (note BETWEEN 0 AND
    10));

Create table reservation of treservation(PRIMARY KEY(numres),FOREIGN KEY(res_client)
    REFERENCES client,
FOREIGN KEY(res_chambre) REFERENCES Chambre,CHECK(ATEARRIVEE<ATEDEPART));

```

## Resultat :

```
SQL> CREATE TABLE Hotel OF thotel(PRIMARY KEY(NUMHOTEL),CONSTRAINT etoiles_ck CHECK (etoiles BETWEEN 0 AND 5))
  2 NESTED TABLE hotel_chambre store as table_hotel_chambre,
  3 NESTED TABLE hotel_evaluation store as table_hotel_evaluation;
Table cr  e.

SQL>
SQL> CREATE TABLE Chambre OF tchambre(PRIMARY KEY(NUMCHAMBRE,NUMHOTEL), FOREIGN KEY(chambre_hotel) REFERENCES Hotel,
  2 check(TYPECHAMBRE in ('simple', 'double', 'triple','suite','autre')) )
  3 NESTED TABLE chambre_res store as table_chambre_res;
Table cr  e.

SQL>
SQL> Create table client of tclient(PRIMARY KEY(NUMCLIENT))
  2 NESTED TABLE client_eval store as table_client_eval,
  3 NESTED TABLE client_res store as table_client_res;
Table cr  e.

SQL>
SQL> Create table evaluation of tevaluation(PRIMARY KEY(NUMEVAL),FOREIGN KEY(eval_client) REFERENCES client,
  2 FOREIGN KEY(eval_hotel) REFERENCES Hotel,CONSTRAINT noteck CHECK (note BETWEEN 0 AND 10));
Table cr  e.

SQL>
SQL> Create table reservation of treservation(PRIMARY KEY(numres),FOREIGN KEY(res_client) REFERENCES client,
  2 FOREIGN KEY(res_chambre) REFERENCES Chambre,CHECK (DATEARRIVEE<DATEDEPART));
Table cr  e.
```

## 1.4 Langage de manipulation de donn  es

### 1.4.1 Remplissage des tables :

Nous ins  rons d'abord les donn  es dans les tables sans leurs associations, puis nous cr  ons les r  f  rences entre les tables.

#### Requete :

```
-- la table hotel
INSERT into hotel values(
    thotel(1 ,
        'Renaissance' ,
        'Tlemcen',
        5,
        NULL,
        t_set_ref_chambre(),
        t_set_ref_evaluation() )
);

-- la table client
INSERT into client values(
    tclient(1,
        ' BOUROUBI',
        'Taous',
        NULL,
        t_set_ref_reservation(),
```

```

        t_set_ref_evaluation()));
-- la table chambre
INSERT into CHAMBRE values(
    tchambre(1,
    2,
    1,
    'simple',
    4500,
    NULL,
    t_set_ref_reservation()));
--la table reservation
INSERT into reservation values(
    treservation(1,
    '11/05/2022' ,
    '15/05/2022',
    NULL,
    NULL));
--la table evaluation
INSERT into evaluation values(
    tevaluation(1,
    '11/05/2022',
    5,
    NULL,
    NULL));

```

**Resultat :**

```

SQL> INSERT into hotel values(
2   thotel(1,
3   'Renaissance' ,
4   'Tlemcen',
5   5,
6   NULL,
7   t_set_ref_chambre(),
8   t_set_ref_evaluation() )
9   );

1 ligne cr  e.

SQL> INSERT into client values(
2   tclient(1,
3   'BOUROUBI',
4   'Taous',
5   NULL,
6   t_set_ref_reservation(),
7   t_set_ref_evaluation()));

1 ligne cr  e.

SQL> INSERT into CHAMBRE values(
2   tchambre(1,
3   2,
4   1,
5   'simple',
6   4500,
7   NULL,
8   t_set_ref_reservation()));

1 ligne cr  e.

SQL> INSERT into reservation values(
2   treservation(1,
3   '11/05/2022' ,
4   '15/05/2022',
5   NULL,
6   NULL));

1 ligne cr  e.

```

```

SQL> INSERT into evaluation values(
2   tevaluation(1,
3   '11/05/2022',
4   5,
5   NULL,
6   NULL));

1 ligne cr  e.

```

remplissage des associations :

Requete :

```

-- association hotel chambre
UPDATE chambre SET chambre_hotel = (SELECT REF(H) FROM HOTEL H WHERE H.NUMHOTEL = 2)
WHERE NUMHOTEL = 2 AND numchambre = 1;
INSERT INTO TABLE(SELECT H.hotel_chambre FROM HOTEL H WHERE H.NUMHOTEL = 2) VALUES (
(SELECT REF(c) FROM chambre c WHERE c.NUMHOTEL = 2 AND c.numchambre = 1)
);

-- association reservation chambre client
UPDATE reservation SET res_chambre = (SELECT REF(c) FROM chambre c WHERE c.numchambre
=1 and c.NUMHOTEL=5)
,res_client=(SELECT REF(c) FROM client c WHERE c.NUMCLIENT =1 )

```

```

INSERT INTO TABLE(SELECT c.chambre_res FROM chambre c WHERE c.numchambre =1 and
c.NUMHOTEL=5) VALUES (
(SELECT REF(r) FROM reservation r WHERE r.numres = 1 )
);
INSERT INTO TABLE(SELECT c.client_res FROM client c WHERE c.numclient =1 ) VALUES (
(SELECT REF(r) FROM reservation r WHERE r.numres = 1 )
);
-- association evaluation hotel client
UPDATE evaluation SET eval_hotel = (SELECT REF(h) FROM hotel h WHERE h.NUMHOTEL =1 )
,eval_client=(SELECT REF(c) FROM client c WHERE c.NUMCLIENT =1 )
WHERE NUMEVAL=1;
INSERT INTO TABLE(SELECT h.hotel_evaluation FROM hotel h WHERE h.numhotel =1 ) VALUES
(
(SELECT REF(e) FROM evaluation e WHERE e.numeval = 1 )
);
INSERT INTO TABLE(SELECT c.client_eval FROM client c WHERE c.numclient =1 ) VALUES (
(SELECT REF(e) FROM evaluation e WHERE e.numeval = 1 )
);

```

## Resultat :

```

SQL> UPDATE chambre SET chambre_hotel = (SELECT REF(H) FROM HOTEL H WHERE H.NUMHOTEL =2)
2 WHERE NUMHOTEL =2AND numchambre =1;
1 ligne mise à jour.

SQL> INSERT INTO TABLE(SELECT H.hotel_chambre FROM HOTEL H WHERE H.NUMHOTEL =2) VALUES
2 ( (SELECT REF(c) FROM chambre c WHERE c.NUMHOTEL =2AND c.numchambre =1));
1 ligne créée.

SQL> UPDATE evaluation SET eval_hotel = (SELECT REF(h) FROM hotel h WHERE h.NUMHOTEL =1) ,
2 eval_client=(SELECT REF(c) FROM client c WHERE c.NUMCLIENT =1)WHERE NUMEVAL=1;
1 ligne mise à jour.

SQL> INSERT INTO TABLE(SELECT h.hotel_evaluation FROM hotel h WHERE h.numhotel =1) VALUES
2 ( (SELECT REF(e) FROM evaluation e WHERE e.numeval =1));
1 ligne créée.

SQL> INSERT INTO TABLE(SELECT c.client_eval FROM client c WHERE c.numclient =1)
2 VALUES ( (SELECT REF(e) FROM evaluation e WHERE e.numeval =1));
1 ligne créée.

SQL> UPDATE reservation SET res_chambre = (SELECT REF(c) FROM chambre c WHERE c.numchambre =1and c.NUMHOTEL=5) ,
2 res_client=(SELECT REF(c) FROM client c WHERE c.NUMCLIENT =1)WHERE numres=1;
1 ligne mise à jour.

SQL> INSERT INTO TABLE(SELECT c.chambre_res FROM chambre c WHERE c.numchambre =1and c.NUMHOTEL=5)
2 VALUES ((SELECT REF(r) FROM reservation r WHERE r.numres =1));
1 ligne créée.

SQL> INSERT INTO TABLE(SELECT c.client_res FROM client c WHERE c.numclient =1) VALUES
2 ((SELECT REF(r) FROM reservation r WHERE r.numres =1));
1 ligne créée.

```



## 1.5 Langage d'interrogation de données

### 1.5.1 Lister les noms d'hôtels et leurs villes respectives :

Requete :

```
SELECT H.NOMHOTEL AS NOM , H.VILLE AS VILLE FROM Hotel H;
```

Resultat :

```
SQL> SET LINESIZE 1000;
SQL> SET PAGESIZE 100;
SQL> SELECT H.NOMHOTEL AS NOM , H.VILLE AS VILLE FROM Hotel H;

NOM-----VILLE-----
Renaissance          Tlemcen
Seybouse             Annaba
Hôtel Novotel        Constantine
Saint George d'Alger Alger
Ibis Alger Aéroport Alger
El Mountazah Annaba Annaba
Hôtel Albert 1er     Alger
Chems                Oran
Colombe              Oran
Mercure              Alger
Le Méridien          Oran
Hôtel Sofitel        Alger

12 ligne(s) sélectionné(s).
```

### 1.5.2 Lister les hôtels sur lesquels porte au moins une réservation.

La fonction d'agrégation COUNT() permet de compter le nombre d'enregistrement dans une table.

Requete :

```
select  H.NOMHOTEL, count(distinct r.column_value) as nb_reservation from hotel h,
table(h.hotel_chambre) c, table(deref(c.column_value).chambre_res) r group by
H.NOMHOTEL
having count(distinct r.column_value)>=1;
```

Resultat :

```
SQL> select  H.NOMHOTEL, count(distinct r.column_value) as nb_reservation from hotel h,
2 table(h.hotel_chambre) c, table(deref(c.column_value).chambre_res) r group by H.NOMHOTEL
3 having count(distinct r.column_value)>=1;

NOMHOTEL-----NB_RESERVATION-----
Chems                1
Colombe              2
El Mountazah Annaba  4
Hôtel Albert 1er     1
Hôtel Sofitel        3
Ibis Alger Aéroport  5
Le Méridien          7
Mercure              1
Saint George d'Alger  1
Seybouse             1

10 ligne(s) sélectionné(s).
```

### 1.5.3 Quels sont les clients qui ont toujours séjourné au premier étage ?

Requete :

```
select c.NUMCLIENT,c.NOMCLIENT,C.PRENOMCLIENT,
       deref(deref(r.column_value).res_chambre).numchambre AS CHAMBRE FROM CLIENT C ,
TABLE(c.client_res) r where deref(deref(r.column_value).res_chambre).ETAGE=1;
```

Resultat :

```
SQL> SET LINESIZE 1500;
SQL> select c.NUMCLIENT,c.NOMCLIENT,C.PRENOMCLIENT, deref(deref(r.column_value).res_chambre).numchambre AS CHAMBRE FROM CLIENT C ,
2 TABLE(c.client_res) r where deref(deref(r.column_value).res_chambre).ETAGE=1;
```

NUMCLIENT	NOMCLIENT	PRENOMCLIENT	CHAMBRE
2	BOUZIDI	AMEL	2
6	OUSSEDIK	Hakim	2
6	OUSSEDIK	Hakim	100
13	ABBOU	Mohamed	2
14	ABDELAZIZ	Ahmed	2
23	AGGOUN	Khadidja	1
28	BABACI	Mourad	3
20	BENOUADAH	Mohammed	2
22	ADDAD	Fadila	25
26	AROUEL	Leila	25
35	BEHADI	Youcef	101
35	BEHADI	Youcef	2
35	BEHADI	Youcef	2

13 ligne(s) sélectionnée(s).

### 1.5.4 Quels sont les hôtels (nom, ville) qui offrent des suites ? et donner le prix pour chaque suite

Requete :

```
SELECT H.NOMHOTEL AS NOM , H.VILLE AS VILLE, Deref(r.column_value).PRIXNUIT AS PRIX
FROM Hotel H,table(h.hotel_chambre) r where Deref(r.column_value).TYPECHAMBRE= 'suite'
;
```

Resultat :

```
SQL> SELECT H.NOMHOTEL AS NOM , H.VILLE AS VILLE, Deref(r.column_value).PRIXNUIT AS PRIX
2 FROM Hotel H,table(h.hotel_chambre) r where Deref(r.column_value).TYPECHAMBRE='suite' ;
```

NOM	VILLE	PRIX
Seybouse	Annaba	16000
Seybouse	Annaba	16500
Hôtel Sofitel	Alger	19500
Hôtel Sofitel	Alger	19500
Hôtel Sofitel	Alger	19500

### 1.5.5 Quel est le type de chambre le plus réservé habituellement, pour chaque hôtel d'Alger ?

la fonction d'agrégation MAX() permet de retourner la valeur maximale d'une colonne dans un set d'enregistrement.

Requete :

```

select c.TYPECHAMBRE as type,count(r.column_value )as nb from chambre
      c,table(c.chambre_res) r
where deref(c.chambre_hotel).ville='Alger'
group by c.TYPECHAMBRE
HAVING COUNT(r.COLUMN_VALUE) = (
      SELECT MAX(nb) FROM (
            SELECT count(r2.column_value )as nb
            FROM CHAMBRE c2,
            table(c2.chambre_res) r2
where deref(c2.chambre_hotel).ville='Alger'
group by c2.TYPECHAMBRE
      )
);

```

Resultat :

```

SQL> select c.TYPECHAMBRE as type,count(r.column_value )as nb from chambre c,table(c.chambre_res) r
2  where deref(c.chambre_hotel).ville='Alger'
3  group by c.TYPECHAMBRE
4  HAVING COUNT(r.COLUMN_VALUE) = (
5      SELECT MAX(nb) FROM (
6          SELECT count(r2.column_value )as nb
7          FROM CHAMBRE c2,
8          table(c2.chambre_res) r2
9  where deref(c2.chambre_hotel).ville='Alger'
10 group by c2.TYPECHAMBRE
11 )
12 );

```

TYPE	NB
double	5
simple	5

### 1.5.6 Quels sont les hôtels (nom, ville) ayant obtenu une moyenne de notes $\geq 6$ , durant l'année 2022

La fonction d'agrégation AVG() permet de calculer une valeur moyenne sur un ensemble d'enregistrement de type numérique et non nul.

Requete :

```

SELECT H.NOMHOTEL AS NOM , H.VILLE AS VILLE ,AVG(DEREF(eval.column_value).note) AS
      NOTEMOY
FROM Hotel H,table(h.hotel_evaluation) eval where
      DEREF(eval.column_value).datee<'01/01/2023'
      AND DEREF(eval.column_value).datee>'31/12/2021'
group by H.NOMHOTEL,H.VILLE
having AVG(DEREF(eval.column_value).note) >=6;

```

Resultat :

```
SQL> SELECT H.NOMHOTEL AS NOM , H.VILLE AS VILLE ,AVG(DEREF(eval.column_value).note) AS NOTEMOY
2 FROM Hotel H,table(h.hotel_evaluation) eval where DEREF(eval.column_value).datee<'01/01/2023'
3 AND DEREF(eval.column_value).datee>'31/12/2021'
4 group by H.NOMHOTEL,H.VILLE
5 having AVG(DEREF(eval.column_value).note) >=6;
```

NOM	VILLE	NOTEMOY
Hôtel Novotel	Constantine	10
Colombe	Oran	9
Seybouse	Annaba	7
Le Méridien	Oran	7

```
SQL>
```

### 1.5.7 Quel est l'hôtel ayant réalisé le meilleur chiffre d'affaire durant l'été 2022 (juin, juillet, aout)

Cette requête récupère le nom de l'hôtel (NOMHOTEL) et la somme d'une valeur calculée (calculchiffre()) pour chaque hôtel, en fonction de certaines conditions.

Les conditions spécifient que la date d'arrivée (DATEARRIVEE) d'une réservation doit être le 1er juin 2022 ou après, et la date de départ (DATEDEPART) doit être le 31 août 2022 ou avant.

Le résultat est regroupé par le nom de l'hôtel (NOMHOTEL), et la clause HAVING garantit que seuls les hôtels ayant la somme maximale de calculchiffre() dans la plage de dates spécifiée sont inclus.

Le résultat final de la requête est le nom de l'hôtel et la somme totale de calculchiffre() pour les hôtels qui satisfont aux critères.

**Requete :**

```
select h.NOMHOTEL AS NOM_HOTEL,SUM(deref(c.column_value).calcul_chiffre()) from hotel
  h , table(h.hotel_chambre) c ,
  table(deref(c.column_value).chambre_res) r
Where deref(r.column_value).DATEARRIVEE>='01/06/2022' AND
  deref(r.column_value).DATEDEPART<='31/08/2022'
group by h.NOMHOTEL
HAVING SUM(deref(c.column_value).calcul_chiffre()) = (
  SELECT MAX(CA) FROM (
    SELECT SUM(deref(c2.column_value).calcul_chiffre()) AS CA from hotel h ,
      table(h.hotel_chambre) c2 ,
      table(deref(c2.column_value).chambre_res) r2
    Where deref(r2.column_value).DATEARRIVEE>='01/06/2022' AND
      deref(r2.column_value).DATEDEPART<='31/08/2022'
    group by h.NOMHOTEL
  )
);
```

**Resultat :**

```

SQL> select h.NOMHOTEL AS NOM_HOTEL,SUM(deref(c.column_value).calcul_chiffre()) from hotel h , table(h.hotel_chambre) c ,
2   table(deref(c.column_value).chambre_res) r
3   where deref(r.column_value).DATEARRIVEE>='01/06/2022' AND deref(r.column_value).DATEDEPART<='31/08/2022'
4   group by h.NOMHOTEL
5   HAVING SUM(deref(c.column_value).calcul_chiffre()) = (
6     SELECT MAX(CA) FROM (
7       SELECT SUM(deref(c2.column_value).calcul_chiffre()) AS CA from hotel h , table(h.hotel_chambre) c2 ,
8       table(deref(c2.column_value).chambre_res) r2
9       where deref(r2.column_value).DATEARRIVEE>='01/06/2022' AND deref(r2.column_value).DATEDEPART<='31/08/2022'
10      group by h.NOMHOTEL
11     )
12  );

```

NOM_HOTEL	SUM(DEREF(C.COLUMN_VALUE).CALCUL_CHIFFRE())
Ibis Alger Aéroport	16000

# Chapitre 2

## Partie 2 : NoSQL – Modèle orienté « documents »

### 2.1 Modélisation orientée document

#### 2.1.1 La proposition d'une modélisation orientée document

Dans ce projet, nous proposerons deux conceptions différentes pour les comparer et déterminer la meilleure. La première inclut la jointure, tandis que la deuxième se concentre sur l'imbrication.

La première comprend les 3 collections suivantes :

- hôtel contenant les informations relatives à l'hôtel telles que son nom, sa ville, ses chambres ainsi que ses évaluations. Cette collection contient également des collections imbriquées pour les chambres, qui fournissent des informations sur chaque chambre, ainsi qu'une autre collection imbriquée pour les évaluations de l'hôtel..

- client contenant les informations relatives à le client telles que son nom, son prénom et son email

- réservation pourrait être représentée sous forme d'un document contenant les informations relatives à la réservation telles que la date de départ et d'arrivée, le num de chambre réservée, le num de client qui a réservé, etc

```

{
  "numhotel": "int",
  "nomhotel": "string",
  "ville": "string",
  "etoiles": "int",
  "chambre": [
    { "numchambre": "int",
      "etage": "int",
      ....
    },
    ....
  ],
  "evaluation": [ { "numeval": "int",
                    "date": "Date",
                    "note": "int",
                    "numclient": "int" },
                  .... ]
}
{
  "numclient": 'int',
  "nomclient": "string",
  "prenomclient": "string",
  "email": "string"
}
{
  "numres": "string",
  "numclient": "int",
  "numchambre": "int",
  "numhotel": "int",
  "Datearr": "Date",
  "Datedep": "Date"
}

```

FIGURE 2.1 – exemple pour la modelisation1

Par rapport à la deuxième proposition, nous suggérons de réduire le nombre de collections en créant une seule collection pour les hôtels et les réservations, tout en conservant la collection client pour les informations relatives aux clients. Chaque document hotel contiendra les informations essentielles telles que le nom, la ville, les chambres, les évaluations et les réservations. Cette collection inclura également trois collections imbriquées : une pour les chambres qui fournira des informations sur chaque chambre, ainsi que deux autres pour les évaluations et les réservations d'hôtel.

```

{
  "numhotel": "int",
  "nomhotel": "string",
  "ville": "string",
  "etoiles": "int",
  "chambre": [
    { "numchambre": "int",
      "etage": "int",
      ....
    },
    ....
  ],
  "reservation": [
    { "numres": "string",
      "numclient": "int",
      "numchambre": "int ",
      "Datearr": "Date",
      "Datedep": "Date"
    },
    ....
  ],
  "evaluation": [ { "numeval": "int",
    "date": "Date",
    "note": "int",
    "numclient": "int" },
    .... ]
}
{
  "numclient": 'int',
  "nomclient": "string",
  "prenomclient": "string",
  "email": "string"
}

```

FIGURE 2.2 – exemple pour la modelisation2

### 2.1.2 l'illustration des exemples de la BD fournie

#### 2.1.3 Conception 1 :

hotel :



```

  ▶ _id: ObjectId('644c00057068083632a7829f')
    numhotel: 1
    nomhotel: "Renaissance"
    ville: "Tlemcen"
    etoiles: 5
  ▼ chambre: Array
    ▼ 0: Object
      numchambre: 10
      etage: 1
      type: "simple"
      prixnuit: 7100
    ▶ 1: Object
    ▶ 2: Object
    ▶ 3: Object
    ▶ 4: Object
    ▶ 5: Object
    ▶ 6: Object
    ▶ 7: Object
  ▼ evaluation: Array
    ▼ 0: Object
      numeval: 1
      date: "11/05/2022"
      note: 5
      numclient: 40

```

FIGURE 2.3 – exemple pour le document hotel

client :

```

_id: ObjectId('644d941b7068083632a7831b')
numclient: 1
nomclient: "BOUROUBI"
prenomclient: "Taous"
email: "BOUROUBI@gmail.com"

```

FIGURE 2.4 – exemple pour le document client

reservation :

```

_id: ObjectId('644f7e80b02cee08773bfd6f')
numres: 1
numclient: 40
numchambre: 23
numhotel: 1
Datearr: "05/09/2022"
Datedep: "05/13/2022"

```

FIGURE 2.5 – exemple pour le document reservation

## 2.1.4 Conception 2 :

hotel :

```

_id: ObjectId('644f7f04b02cee08773bfd93')
numhotel: 1
nomhotel: "Renaissance"
ville: "Tlemcen"
etoiles: 5
▼ chambre: Array
  ▼ 0: Object
    numchambre: 10
    etage: 1
    type: "simple"
    prixnuit: 7100
  ▶ 1: Object
  ▶ 2: Object
  ▶ 3: Object
  ▶ 4: Object
  ▶ 5: Object
  ▶ 6: Object
  ▶ 7: Object
▼ reservation: Array
  ▼ 0: Object
    numres: 1
    numclient: 40
    numchambre: 23
    Datearr: "05/09/2022"
    Datedep: "05/13/2022"
▼ evaluation: Array
  ▼ 0: Object
    numeval: 1
    date: "11/05/2022"
    note: 5
    numclient: 40

```

FIGURE 2.6 – exemple pour le document hotel

client :

```

_id: ObjectId('644d941b7068083632a7831b')
numclient: 1
nomclient: "BOUROUBI"
prenomclient: "Taous"
email: "BOUROUBI@gmail.com"

```

FIGURE 2.7 – exemple pour le document client

### 2.1.5 Justification de choix de conception

La première proposition utilise des collections distinctes pour chaque entité (hôtel, client, réservation) et utilise des collections imbriquées pour stocker les informations détaillées telles que les chambres et les évaluations. Cette approche peut être bénéfique si les données doivent être analysées de manière isolée pour chaque entité. Cela facilite également l'ajout ou la suppression de données pour chaque entité sans affecter les autres.

La deuxième proposition utilise une seule collection pour les hôtels et les réservations. Cela simplifie le processus de récupération de données pour une réservation spécifique car toutes les

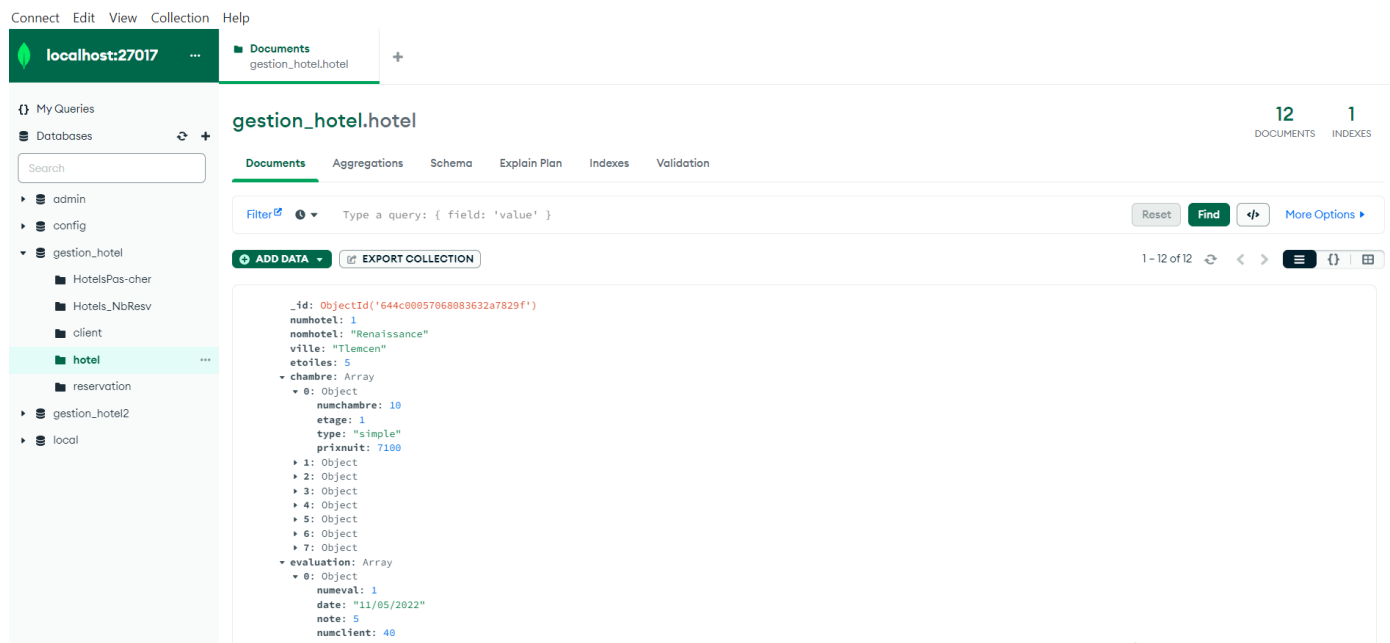
informations sont stockées dans un seul document. En outre, la méthode permet une recherche rapide pour un hotel donnée. Enfin, cela évite les redondances de données telles que la ville ou le nom de l'hôtel qui peuvent apparaître dans plusieurs documents avec la première proposition.

## 2.2 Remplissage de la base de données

Pour le processus de remplissage, nous avons pris chaque collection de données et les avons enregistrées dans des fichiers JSON distincts. Ensuite, nous avons importé ces fichiers .

### 2.2.1 Conception 1 :

Collection hotel :



The screenshot shows the MongoDB Compass interface. On the left, the 'Databases' sidebar lists the database structure, with 'gestion\_hotel' expanded and 'hotel' selected. The main panel displays the 'gestion\_hotel.hotel' collection. The 'Documents' tab is active, showing a single document with the following structure:

```
{
  "_id": ObjectId("644c00057060083632a7029f"),
  "numhotel": 1,
  "nomhotel": "Renaissance",
  "ville": "Tlemcen",
  "etoiles": 5,
  "chambre": [
    {
      "numchambre": 10,
      "etage": 1,
      "type": "simple",
      "prixnuit": 7100
    },
    {
      "numchambre": 11,
      "etage": 1,
      "type": "simple",
      "prixnuit": 7100
    },
    {
      "numchambre": 12,
      "etage": 1,
      "type": "simple",
      "prixnuit": 7100
    },
    {
      "numchambre": 13,
      "etage": 1,
      "type": "simple",
      "prixnuit": 7100
    },
    {
      "numchambre": 14,
      "etage": 1,
      "type": "simple",
      "prixnuit": 7100
    },
    {
      "numchambre": 15,
      "etage": 1,
      "type": "simple",
      "prixnuit": 7100
    },
    {
      "numchambre": 16,
      "etage": 1,
      "type": "simple",
      "prixnuit": 7100
    },
    {
      "numchambre": 17,
      "etage": 1,
      "type": "simple",
      "prixnuit": 7100
    }
  ],
  "evaluation": [
    {
      "numeval": 1,
      "date": "11/05/2022",
      "note": 5,
      "numclient": 40
    }
  ]
}
```

Collection reservation :

gestion\_hotel.reservation

33 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find More Options

ADD DATA EXPORT COLLECTION

1 - 20 of 33

```

_id: ObjectId('644f7e8b02cee08773bfd6f')
numres: 1
numclient: 40
numchambre: 23
numhotel: 1
Datearr: "05/09/2022"
Datedep: "05/13/2022"

_id: ObjectId('644f7e8b02cee08773bfd70')
numres: 2
numclient: 40
numchambre: 8
numhotel: 2
Datearr: "05/01/2022"
Datedep: "05/05/2022"

_id: ObjectId('644f7e8b02cee08773bfd71')
numres: 3
numclient: 14
numchambre: 21
numhotel: 4

```

## Collection client :

gestion\_hotel.client

69 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find More Options

ADD DATA EXPORT COLLECTION

1 - 20 of 69

```

_id: ObjectId('644d941b70608083632a7831b')
numclient: 1
nomclient: "BOUROUBI"
prenomclient: "Taous"
email: "BOUROUBI@gmail.com"

_id: ObjectId('644d941b70608083632a7831c')
numclient: 2
nomclient: "BOUZIDI"
prenomclient: "AMEL"
email: "BOUZIDI@gmail.com"

_id: ObjectId('644d941b70608083632a7831d')
numclient: 3
nomclient: "LACHEMI"
prenomclient: "Bouzid"
email: "LACHEMI@gmail.com"

_id: ObjectId('644d941b70608083632a7831e')
numclient: 4

```

## 2.2.2 Conception 2 :

### Collection hotel :

**gestion\_hotel2.hotel** 12 DOCUMENTS 1 INDEXES

Filter: Type a query: { field: 'value' } [Reset] [Find] [More Options]

ADD DATA EXPORT COLLECTION

1 - 12 of 12

```

{
  "_id": ObjectId("6447f84b2cee88773bf993"),
  "numhotel": 1,
  "nomhotel": "Renaissance",
  "ville": "Tlemcen",
  "etoiles": 5,
  "chambre": Array,
  "reservation": Array,
  "evaluation": Array
}

{
  "_id": ObjectId("6447f84b2cee88773bf994"),
  "numhotel": 2,
  "nomhotel": "Seybouse",
  "ville": "Annaba",
  "etoiles": 3,
  "chambre": Array,
  "reservation": Array,
  "evaluation": Array
}

{
  "_id": ObjectId("6447f84b2cee88773bf995"),
  "numhotel": 3,
  "nomhotel": "Hôtel Novotel",
  "ville": "Constantine",
  "etoiles": 4,
  "chambre": Array
}

```

## Collection client :

**gestion\_hotel2.client** 69 DOCUMENTS 1 INDEXES

Filter: Type a query: { field: 'value' } [Reset] [Find] [More Options]

ADD DATA EXPORT COLLECTION

1 - 20 of 69

```

{
  "_id": ObjectId("644c087f7068883632a782b9"),
  "numclient": 1,
  "nomclient": "BOUROUBI",
  "prenomclient": "Taous",
  "email": "BOUROUBI@gmail.com"
}

{
  "_id": ObjectId("644c087f7068883632a782ba"),
  "numclient": 2,
  "nomclient": "BOUZIDI",
  "prenomclient": "AMEL",
  "email": "BOUZIDI@gmail.com"
}

{
  "_id": ObjectId("644c087f7068883632a782bb"),
  "numclient": 3,
  "nomclient": "LACHEMI",
  "prenomclient": "Bouzi",
  "email": "LACHEMI@gmail.com"
}

{
  "_id": ObjectId("644c087f7068883632a782bc"),
  "numclient": 4,
  "nomclient": "BOUCHEMLA",
  "prenomclient": "Elias",
  "email": "BOUCHEMLA@gmail.com"
}

```

## 2.3 Les requetes

### 2.3.1 Affichage de tous les hôtels classés « 3 étoiles »

Requete :

```
//requete 1
db.hotel.find({ etoiles: 3 }, { _id: 0, nomhotel: 1 });
```

Resultat :

```
> db.hotel.find({ etoiles: 3 }, { _id: 0, nomhotel: 1 });
< {
  nomhotel: 'Seybouse'
}
{
  nomhotel: 'ElMountazah Annaba'
}
{
  nomhotel: 'Hôtel Albert 1er'
}
{
  nomhotel: 'Colombe'
}
>
```

2.3.2 Récupérer dans une nouvelle collection Hotels-NbResv, les noms des hôtels et le nombre total de réservations par hôtel; la collection devra être ordonnée par ordre décroissant du nombre de réservations.

**Collection1 :**

Requete :

```
//requete 2
db.reservation.aggregate([
  {
    $group: {
      _id: "$numhotel",
      count: { $sum: 1 },
    },
  },
  {
    $lookup: {
      from: "hotel",
      localField: "_id",
      foreignField: "numhotel",
      as: "hotelInfo",
    },
  },
  {
    $project: {
      _id: 0,
      NOMHOTEL: { $arrayElemAt: ["$hotelInfo.nomhotel", 0] },
      NBRRESERVATION: "$count",
    },
  },
  { $sort: { NBRRESERVATION: -1 } },
  { $out: "Hotels_NbResv" },
]);
//.explain()
```

Resultat :

The screenshot shows the MongoDB Compass interface. On the left, the 'gestion\_hotel' database is selected, and the 'Hotels\_NbResv' collection is highlighted. The main panel displays the 'gestion\_hotel.Hotels\_NbResv' collection with 1 document. The document contains the following data:

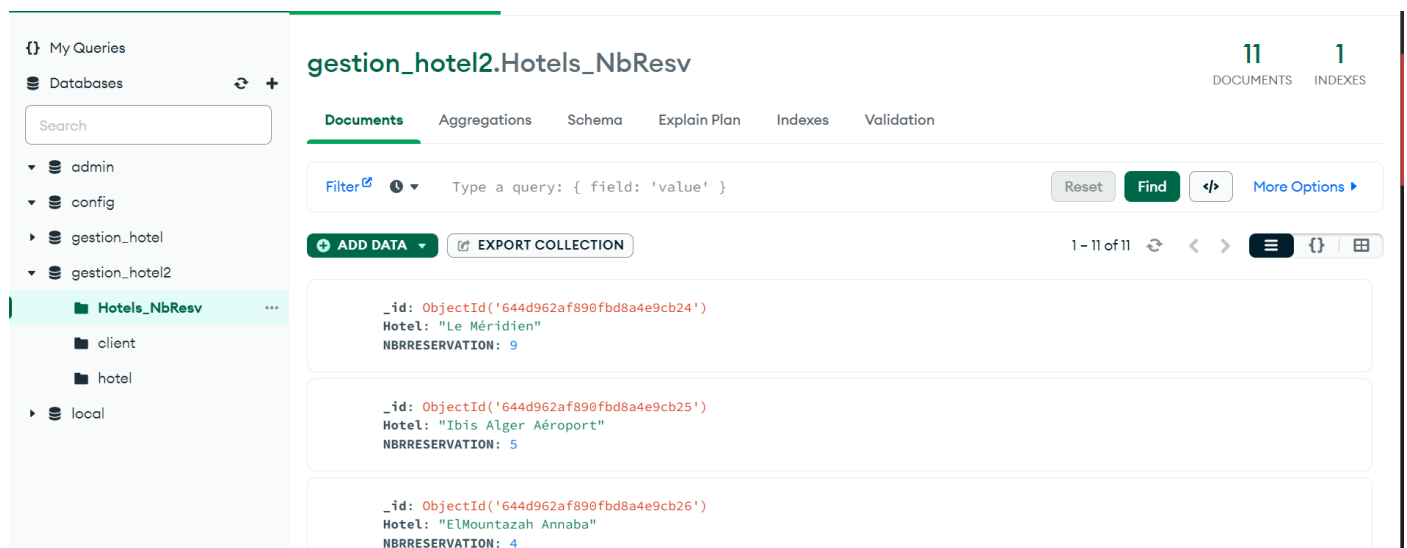
_id	NOMHOTEL	NBRRESERVATION
ObjectId('644d9513f890fbd8a4e9ca1c')	"Le Méridien"	9

## Collection2 :

Requete :

```
//requete 2
db.hotel.aggregate([
  { $unwind: "$reseravtion" },
  {
    $group: {
      _id: "$nomhotel",
      count: { $sum: 1 },
    },
  },
  {
    $project: {
      _id: 0,
      Hotel: "$_id",
      NBRRESERVATION: "$count",
    },
  },
  { $sort: { NBRRESERVATION: -1 } },
  { $out: "Hotels_NbResv" },
]);
```

Resultat :



The screenshot shows the MongoDB Compass interface. On the left, the 'Databases' sidebar lists 'gestion\_hotel2' as the selected database, with 'Hotels\_NbResv' highlighted as the current collection. The main panel displays the 'gestion\_hotel2.Hotels\_NbResv' collection with 11 documents and 1 index. The 'Documents' tab is active, showing a list of documents. Each document contains an ObjectId, a hotel name, and a reservation count (NBRRESERVATION).

_id	Hotel	NBRRESERVATION
ObjectId('644d962af899fbd8a4e9cb24')	Le Méridien	9
ObjectId('644d962af899fbd8a4e9cb25')	Ibis Alger Aéroport	5
ObjectId('644d962af899fbd8a4e9cb26')	ElMountazah Annaba	4



### 2.3.3 Dans une collection HotelsPas-cher, récupérer les hôtels dont le prix des chambres ne dépasse pas 6000 DA.

Requete :

```
//requete 3
db.hotel.aggregate([
  {
    $match: {
      "chambre.prixnuit": { $lte: 6000 },
    },
  },
  {
    $out: "HotelsPas-cher",
  },
]);
```

Resultat :

The screenshot shows the MongoDB Compass interface. On the left, the 'Databases' sidebar lists 'gestion\_hotel' and its collections, including 'HotelsPas-cher'. The main panel shows the 'gestion\_hotel.HotelsPas-cher' collection with 8 documents and 1 index. The 'Documents' tab is active, displaying two documents:

```
{
  "_id": ObjectId('644c00057068083632a782a0'),
  "numhotel": 2,
  "nomhotel": "Seybouse",
  "ville": "Annaba",
  "etoiles": 3,
  "chambre": Array,
  "evaluation": Array
}
```

```
{
  "_id": ObjectId('644c00057068083632a782a1'),
  "numhotel": 3,
  "nomhotel": "Hôtel Novotel",
  "ville": "Constantine"
}
```

### 2.3.4 Afficher tous les noms d'hôtels ayant obtenu une note moyenne $\geq 5$

Requete :

```
//requete 4
db.hotel.aggregate([
  { $unwind: "$evaluation" },
  {
    $group: {
      _id: {
        num: "$numhotel",
        nom: "$nomhotel",
      },
      count: { $sum: 1 },
      total: { $sum: "$evaluation.note" },
    },
  },
  {
    $project: {
      noteglobale: { $divide: ["$total", "$count"] },
    },
  },
  {
    $match: {
      noteglobale: { $gte: 5 },
    },
  },
  { $sort: { noteglobale: -1 } }
]);
```

Resultat :

```

< {
  _id: {
    num: 11,
    nom: 'Le Méridien'
  },
  noteglobale: 7
}
{
  _id: {
    num: 5,
    nom: 'Ibis Alger Aéroport'
  },
  noteglobale: 6
}
{
  _id: {
    num: 2,
    nom: 'Seybouse'
  },
  noteglobale: 6
}

```

2.3.5 Afficher toutes les réservations d'un client donné (on donnera l'e-mail du client). On affichera le nom de l'hôtel, le numéro de chambre et la date d'arrivée.

**Collection1 :**

Requete :

```

db.reservation.aggregate([
  { $match: {
    "numclient": db.client
      .find({ email: "BELHAMIDI@gmail.com" }, { _id: 0, numclient: 1 })
      .next().numclient,
  }, },
  {
    $project: {
      nom_hotel: db.hotel.find({ numhotel: "numhotel" }, { _id: 0, nomhotel: 1 }).next().nomhotel,
      NUMCHAMBRE: "$numchambre",
      CLIENT: "$numclient",
      DATE: "$datearrivee",
    },
  },
]);

```

Resultat :

```

< {
  _id: ObjectId("644d94237068083632a78361"),
  nom_hotel: [
    'Renaissance'
  ],
  NUMCHAMBRE: 23,
  CLIENT: 40,
  DATE: '05/13/2022'
}
{
  _id: ObjectId("644d94237068083632a78362"),
  nom_hotel: [
    'Seybouse'
  ],
  NUMCHAMBRE: 8,
  CLIENT: 40,
  DATE: '05/05/2022'
}
{
  _id: ObjectId("644d94237068083632a7837a"),

```

Collection2 :

Requete :

```

db.hotel.aggregate([
  { $unwind: "$reseravtion" },
  {
    $match: {
      "reseravtion.numclient": db.client
        .find({ email: "BELHAMIDI@gmail.com" }, { _id: 0, numclient: 1 })
        .next().numclient,
    },
  },
  {
    $project: {
      Nomhotel: "$nomhotel",
      Numchambre: "$reseravtion.numchambre",
      Date: "$reseravtion.datearrivee",
    },
  },
]);

```

Resultat :

```

< {
  _id: ObjectId("644c010f7068083632a78300"),
  Nomhotel: 'Renaissance',
  Numchambre: 23,
  Date: '05/13/2022'
}
{
  _id: ObjectId("644c010f7068083632a78301"),
  Nomhotel: 'Seybouse',
  Numchambre: 8,
  Date: '05/05/2022'
}
{
  _id: ObjectId("644c010f7068083632a7830a"),
  Nomhotel: 'Le Méridien',
  Numchambre: 9,
  Date: '04/14/2022'
}

```

**2.3.6** Afficher toutes les évaluations postées par un client donné (on donnera l'e-mail du client). On affichera le nom de l'hôtel, la date d'évaluation, la note.

Requete :

```
//requete 6
db.hotel.aggregate([
  { $unwind: "$evaluation" },
  {
    $match: {
      "evaluation.numclient": db.client
        .find({ email: "BELHAMIDI@gmail.com" }, { _id: 0, numclient: 1 })
        .next().numclient,
    },
  },
  {
    $project: {
      Client: "$evaluation.numclient",
      nom_hotel: "$nomhotel",
      NOTE: "$evaluation.note",
      DATE: "$evaluation.date",
    },
  },
]);
```

Resultat :

```
< {
  _id: ObjectId("644c00057068083632a7829f"),
  Client: 40,
  nom_hotel: 'Renaissance',
  NOTE: 5,
  DATE: '11/05/2022'
}
{
  _id: ObjectId("644c00057068083632a782a0"),
  Client: 40,
  nom_hotel: 'Seybouse',
  NOTE: 6,
  DATE: '28/04/2022'
}
gestion_hotel>
```

### 2.3.7 Augmenter de 2000DA, le prix unitaire de toutes les chambres des hôtels classés « 5 étoiles »

Requete :

```
//requete 7
db.hotel.updateMany({ etoiles: 5 }, {
  $inc: {
    "chambre.$[].prixnuit": 2000,
  },
});
```

Resultat :

```
> db.hotel.updateMany({ etoiles: 5 }, {
  $inc: {
    "chambre.$[].prixnuit": 2000,
  },
});
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 4,
  upsertedCount: 0
}
```

### 2.3.8 la 2ème requête à l'aide du paradigme Map-Reduce

**Collection 1 :** Requete :

```

var mapFunction = function() {
    emit(this.nomhotel, this.reservation.length);
};

var reduceFunction = function(key, values) {
    return values[0];
};

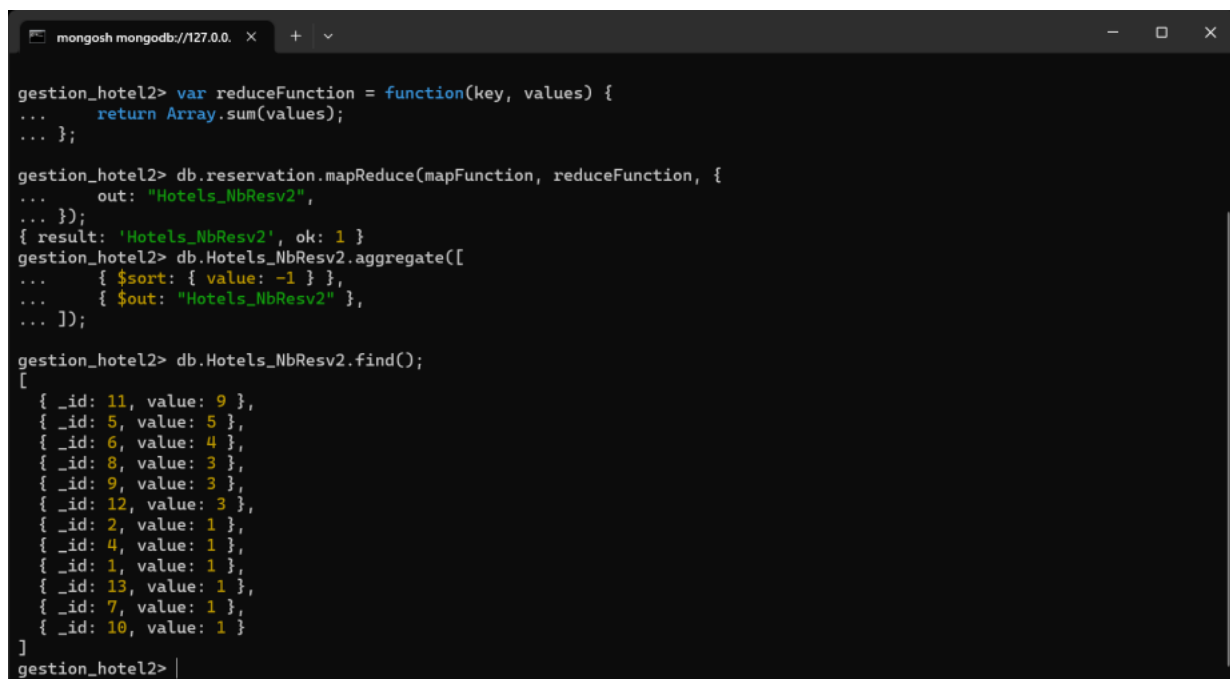
db.hotel.mapReduce(mapFunction, reduceFunction, {
    out: "Hotels_NbResv2",
});

db.Hotels_NbResv2.aggregate([
    { $sort: { value: -1 } },
    { $out: "Hotels_NbResv2" },
]);

db.Hotels_NbResv2.find();

```

Resultat :



```

mongosh mongodb://127.0.0.1:27017
gestion_hotel2> var reduceFunction = function(key, values) {
...   return Array.sum(values);
... };

gestion_hotel2> db.reservation.mapReduce(mapFunction, reduceFunction, {
...   out: "Hotels_NbResv2",
... });
{ result: 'Hotels_NbResv2', ok: 1 }
gestion_hotel2> db.Hotels_NbResv2.aggregate([
...   { $sort: { value: -1 } },
...   { $out: "Hotels_NbResv2" },
... ]);

gestion_hotel2> db.Hotels_NbResv2.find();
[
  { _id: 11, value: 9 },
  { _id: 5, value: 5 },
  { _id: 6, value: 4 },
  { _id: 8, value: 3 },
  { _id: 9, value: 3 },
  { _id: 12, value: 3 },
  { _id: 2, value: 1 },
  { _id: 4, value: 1 },
  { _id: 1, value: 1 },
  { _id: 13, value: 1 },
  { _id: 7, value: 1 },
  { _id: 10, value: 1 }
]
gestion_hotel2>

```

Collection 2 : Requete :



```

    ✓ var mapFunction = function() {
      |   emit(this.numhotel, 1);
      | };

    ✓ var reduceFunction = function(key, values) {
      |   return Array.sum(values);
      | };

    ✓ db.reservation.mapReduce(mapFunction, reduceFunction, {
      |   out: "Hotels_NbResv2",
      | });

    ✓ db.Hotels_NbResv2.aggregate([
      |   { $sort: { value: -1 } },
      |   { $out: "Hotels_NbResv2" },
      | ]);

```

Resultat :

```

Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

gestion_hotel3> db.hotel.mapReduce(mapFunction, reduceFunction, {
...   out: "Hotels_NbResv2",
... });
DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation instead.
See https://docs.mongodb.com/manual/core/map-reduce for details.
{ result: 'Hotels_NbResv2', ok: 1 }
gestion_hotel3> db.Hotels_NbResv2.aggregate([
...   { $sort: { value: -1 } },
...   { $out: "Hotels_NbResv2" },
... ]);

gestion_hotel3> db.Hotels_NbResv2.find();
[
  { _id: 'Le M ridien', value: 9 },
  { _id: 'Ibis Alger A roport', value: 5 },
  { _id: 'ElMountazah Annaba', value: 4 },
  { _id: 'H tel Sofitel', value: 4 },
  { _id: 'Chems', value: 3 },
  { _id: 'Colombe', value: 3 },
  { _id: 'H tel Albert 1er', value: 1 },
  { _id: 'Saint George d'Alger', value: 1 },
  { _id: 'H tel Novotel', value: 1 },
  { _id: 'Seybouse', value: 1 },
  { _id: 'Mercure', value: 1 },
  { _id: 'Renaissance', value: 1 }
]
gestion_hotel3>

```

## 2.4 Analyse

En comparant les deux conceptions de requêtes, on peut constater que la deuxième conception (avec l'imbrication) est la meilleure dans certains cas :

**-Simplicité des requêtes :** La deuxième conception utilise des requêtes plus simples par rapport à la première conception, qui nécessite plusieurs jointures dans la requête des clients. Cela rend la deuxième conception plus facile à comprendre et à maintenir.

**-Performance de lecture :** La deuxième conception offre une lecture des données plus rapide, car elle évite les jointures complexes. Cela peut être bénéfique lorsque la lecture des données est l'objectif principal.

Cependant, si les requêtes impliquent des mises à jour des réservations, la première conception peut être plus adaptée. Elle peut offrir une meilleure gestion des mises à jour grâce aux jointures entre les collections.

En conclusion, le choix de la conception dépend de nos besoins spécifiques et de la qualité des requêtes. Si la lecture des données est la principale préoccupation, la deuxième conception peut être préférable. Mais si les requêtes nécessitent des mises à jour des réservations, la première conception avec les jointures peut être plus adaptée.