*Université de sciences et de la technologie Houari Boumediene*

# Projet Bases de Données avancées
# Master 1 SII

**SQL3-Oracle**
**NoSQL (MongoDB)**


*Présenté Par :*

FERKOUS Sarah
LABIOD Dalil

*Professeur :* M. Boukhedouma

17 Mai 2023

# Table des matières

# Introduction

Les bases de données jouent un rôle crucial dans le stockage, la gestion et la manipulation des données pour de nombreuses applications informatiques. Deux approches populaires pour la gestion des données sont les bases de données relationnelles (SQL) et les bases de données NoSQL. Chacune de ces approches a ses propres caractéristiques, avantages et cas d'utilisation.

Les bases de données relationnelles, représentées par des systèmes tels que SQL3, ont dominé le paysage des bases de données depuis des décennies. Elles utilisent un modèle de données tabulaire avec des relations prédéfinies entre les tables. Les données sont organisées en lignes et colonnes, et les requêtes sont effectuées à l'aide du langage SQL. Les bases de données relationnelles offrent une structure stricte, une cohérence des données et des transactions ACID (Atomicité, Cohérence, Isolation et Durabilité). Elles sont idéales pour les applications nécessitant un schéma fixe, des relations complexes et une forte intégrité des données.

D'un autre côté, les bases de données NoSQL, comme MongoDB, ont émergé pour répondre aux besoins de scalabilité, de flexibilité et de performances des applications modernes. Les bases de données NoSQL utilisent des modèles de données différents, tels que le modèle orienté document, le modèle clé-valeur, le modèle de colonnes ou le modèle graphique. Ces bases de données permettent une évolutivité horizontale, une gestion flexible des schémas et une haute disponibilité. Elles sont bien adaptées aux applications avec des données semi-structurées ou non structurées, des besoins de mise à l'échelle rapide et une tolérance aux pannes.

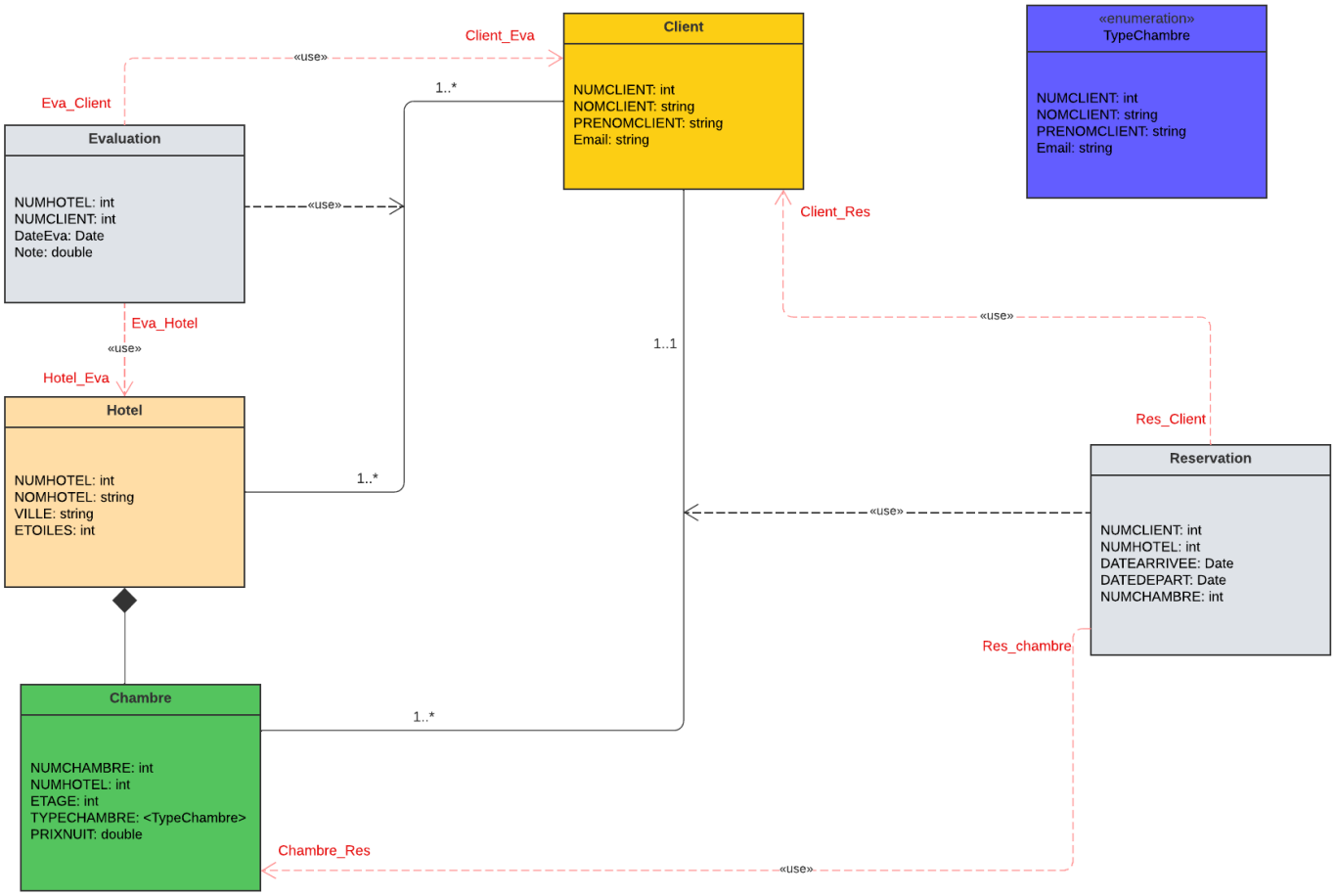
Dans ce projet, nous explorerons et comparerons l'utilisation des bases de données relationnelles (SQL3) et NoSQL (MongoDB). Nous étudierons en détail les concepts, les fonctionnalités et les performances de chaque approche. Nous manipulerons des requêtes, interagirons avec les bases de données et comprendrons comment chaque approche répond aux besoins spécifiques des applications. Cette exploration nous permettra d'acquérir une compréhension approfondie des bases de données relationnelles et NoSQL, et de prendre des décisions éclairées lors du choix de la meilleure approche pour nos projets.

# Première partie

# Relationnel-Objet

## 0.1 Modélisation orientée objet

### 0.1.1 Transformez ce schéma relationnel en un schéma Objet (diagramme de classes)



## 0.2 Création des TableSpaces et utilisateur

### 0.2.1 Créer deux TableSpaces SQL3TBS et SQL3TempTBS

```
CREATE TABLESPACE SQL3_TBS DATAFILE'C:\SQL3_TBS.dat'
SIZE 100M AUTOEXTEND ON;
CREATE TEMPORARY TABLESPACE SQL3_TempTBS TEMPFILE
'C:\SQL3_TempTBS.dat'SIZE 100M AUTOEXTEND on;
```

```
SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL> CREATE TABLESPACE Projet1_SQL3_TBS DATAFILE 'C:\Projet1_SQL3_TBS.dat' SIZE 100M AUTOEXTEND ON;

Tablespace created.

SQL> CREATE TEMPORARY TABLESPACE Projet1_SQL3_TempTBS TEMPFILE 'C:\Projet1_SQL3_TempTBS.dat' SIZE 100M AUTOEXTEND on;

Tablespace created.
```

### 0.2.2 Créer un utilisateur SQL3 en lui attribuant les deux tablespaces créés précédemment

```
CREATE USER projetSql IDENTIFIED BY sql3
DEFAULT TABLESPACE SQL3_TBS
TEMPORARY TABLESPACE SQL3_TempTBS
QUOTA UNLIMITED ON SQL3_TBS;
```

```
SQL> CREATE USER projetSql IDENTIFIED BY sql3
  2  DEFAULT TABLESPACE SQL3_TBS
  3  TEMPORARY TABLESPACE SQL3_TempTBS
  4  QUOTA UNLIMITED ON SQL3_TBS;

User created.
```

### 0.2.3 Donner tous les privilèges à cet utilisateur.

```
GRANT ALL PRIVILEGES TO projetSql;
```

```
SQL> GRANT ALL PRIVILEGES TO projetsql;

Grant succeeded.
```

## 0.3 Langage de définition de données

### 0.3.1 En se basant sur le diagramme de classes fait, définir tous les types nécessaires. Prendre en compte toutes les associations qui existent.

**créer les types incomplets**

```
create type Hotel_Type;
/
create type Chambre_Type;
```

```
/
create type Client_Type;
/
create type Reservation_Type;
/
create type Evaluation_Type;
/
```

```
SQL> create type Hotel_Type;
  2  /

Type created.

SQL> create type Chambre_Type;
  2  /

Type created.

SQL> create type Client_Type;
  2  /

Type created.

SQL> create type Reservation_Type;
  2  /

Type created.

SQL> create type Evaluation_Type;
  2  /

Type created.
```

**création des types nécessaires aux associations**

```
create type t_set_ref_Hotel as table of ref  Hotel_Type;
/
create type t_set_ref_Chambre as table of ref Chambre_Type;
/
create type t_set_ref_Client as table of ref Client_Type;
/
create type t_set_ref_Reservation as table
of ref Reservation_Type;
/
create type t_set_ref_Evaluation as table
of ref Evaluation_Type;
/
```

```
SQL>
SQL> create type t_set_ref_Hotel as table of ref  Hotel_Type;
  2  /

Type created.

SQL>
SQL> create type t_set_ref_Chambre as table of ref Chambre_Type;
  2  /

Type created.

SQL>
SQL> create type t_set_ref_Client as table of ref Client_Type;
  2  /

Type created.

SQL>
SQL> create type t_set_ref_Reservation as table of ref Reservation_Type;
  2  /

Type created.

SQL>
SQL> create type t_set_ref_Evaluation as table of ref Evaluation_Type;
  2  /

Type created.
```

### Creation du type Hotel

```
CREATE OR REPLACE Type Hotel_Type AS OBJECT (
        NumHotel INTEGER,
        NomHotel VARCHAR(50),
        Ville VARCHAR(50),
        Etoiles INTEGER,
        SiteWeb VARCHAR(100),
        HotelChambre t_set_ref_Chambre,
        HotelClient t_set_ref_Client,
        HotelEvaluation t_set_ref_Evaluation
);
/
```

```
SQL>
SQL> CREATE OR REPLACE Type Hotel_Type AS OBJECT (
  2      NumHotel INTEGER,
  3      NomHotel VARCHAR(50),
  4      Ville VARCHAR(50),
  5      Etoiles INTEGER,
  6      SiteWeb VARCHAR(100),
  7  HotelChambre t_set_ref_Chambre,
  8  HotelClient t_set_ref_Client,
  9  HotelEvaluation t_set_ref_Evaluation
 10  );
 11  /

Type created.
```

### Creation du type Chambre

```
CREATE OR REPLACE Type Chambre_Type AS OBJECT (
        NumChambre INTEGER,
        NumHotel INTEGER,
        Etage INTEGER,
```

```
                TypeChambre VARCHAR(50),
                PrixNuit INTEGER,
                ChambreHotel ref Hotel_Type ,
                ChambreReservation t_set_ref_Reservation
        );
        /
```

```
SQL>
SQL> CREATE OR REPLACE Type Chambre_Type AS OBJECT (
  2      NumChambre INTEGER,
  3  NumHotel INTEGER,
  4      Etage INTEGER,
  5      TypeChambre VARCHAR(50),
  6      PrixNuit INTEGER,
  7  ChambreHotel ref Hotel_Type , -- pour referencier la clé etrangere numHotel dans chambre
  8  ChambreReservation t_set_ref_Reservation
  9  );
 10  /

Type created.
```

**Creation du type Client**

```
        CREATE OR REPLACE Type Client_Type AS OBJECT (
                NumClient INTEGER,
                NomClient varchar2(50),
                PrenomClient varchar2(50),
                Email VARCHAR(100),
                ClientReservation t_set_ref_Reservation ,
                ClientEvaluation t_set_ref_Evaluation ,
                ClientHotel t_set_ref_Hotel
        );
        /
```

```
SQL>
SQL> CREATE OR REPLACE Type Client_Type AS OBJECT (
  2      NumClient INTEGER,
  3      NomClient varchar2(50),
  4  PrenomClient varchar2(50),
  5      Email VARCHAR(100),
  6  ClientReservation t_set_ref_Reservation ,
  7  ClientEvaluation t_set_ref_Evaluation ,
  8  ClientHotel t_set_ref_Hotel
  9  );
 10  /

Type created.
```

**Creation du type Reservation**

```
        CREATE OR REPLACE Type Reservation_Type AS OBJECT (
                NumClient INTEGER,
                NUMCHAMBRE INTEGER,
                NumHotel INTEGER,
                DateArrivee DATE,
                DateDepart DATE,
```

```
                ReservationClient ref Client_Type,
                ReservationChambre ref Chambre_Type,
                ReservationHotel ref Hotel_Type
        );
        /
```

```
SQL>
SQL> CREATE OR REPLACE Type Reservation_Type AS OBJECT (
  2      NumClient INTEGER,
  3  NUMCHAMBRE INTEGER,
  4  NumHotel INTEGER,
  5  DateArrivee DATE,
  6      DateDepart DATE,
  7  ReservationClient ref Client_Type,
  8  ReservationChambre ref Chambre_Type,
  9  ReservationHotel ref Hotel_Type
 10  );
 11  /

Type created.
```

**Creation du type Evaluation**

```
                create or replace type Evaluation_Type as object (
                NumHotel INTEGER,
                NumClient INTEGER,
                DateEvaluation date,
                note INTEGER,
                EvaluationClient ref Client_Type,
                EvaluationHotel ref Hotel_Type
        );
        /
```

```
SQL>
SQL> create or replace type Evaluation_Type as object (
  2  NumHotel INTEGER,
  3  NumClient INTEGER,
  4  DateEvaluation date,
  5  note INTEGER,
  6  EvaluationClient ref Client_Type,
  7  EvaluationHotel ref Hotel_Type
  8   );
  9  /

Type created.
```

### 0.3.2 Définir les méthodes permettant de :

— Calculer pour chaque client, le nombre de réservations effectuées.

```
                        --la signature
                        alter type Client_Type
                        add member function resClient
                        return numeric cascade;
                        --le corps
                        create or replace type body Client_Type
                        as member function resClient
```

```
return numeric is
nbrRes number;
Begin
Select cardinality
(self.ClientReservation)
into nbrRes
From CLIENT ;
Return nbrRes ;
End ;
End;
/
/
```

```
SQL>
SQL> alter type Client_Type
  2  add member function resClient return numeric
  3  cascade;

Type altered.

SQL> create or replace type body Client_Type
  2  as member function resClient return numeric
  3  is
  4  nbrRes number;
  5  Begin
  6  Select cardinality(self.ClientReservation) into nbrRes
  7  From CLIENT ;
  8  Return nbrRes ;
  9  End ;
 10  End;
 11  /

Type body created.
```

— Calculer pour chaque hôtel, le nombre de chambres.

```
--la signature
alter type Hotel_Type
add member function nbCham
return numeric cascade;
--le corps
create or replace type body Hotel_Type
as member function nbCham
return numeric is
nbrchambre number;
begin
select cardinality(self.HotelChambre)
into nbrchambre
from HOTEL;
return nbrchambre;
end;
end;
/
```

```
SQL> alter type Hotel_Type
  2  add member function nbCham return numeric
  3  cascade;

Type altered.

SQL> create or replace type body Hotel_Type
  2  as member function nbCham return numeric
  3  is
  4      nbrchambre number;
  5  begin
  6      select cardinality(self.HotelChambre) into nbrchambre
  7      from HOTEL;
  8      return nbrchambre;
  9  end;
 10  end;
 11  /

Type body created.
```

— Calculer pour chaque chambre, son chiffre d'affaire.

```
--la signature
alter type Chambre_Type
add MEMBER FUNCTION ChifAf
RETURN numeric cascade;
--le corps
CREATE OR REPLACE TYPE BODY
Chambre_Type AS
MEMBER FUNCTION ChifAf RETURN numeric
AS
v_chiffre_affaire number;
BEGIN
SELECT SUM(PrixNuit * (r.DateDepart -
r.DateArrivee)) INTO v_chiffre_affaire
FROM Reservation r, Chambre c
WHERE r.NumChambre = c.NumChambre
AND c.NumChambre = self.NumChambre;
RETURN v_chiffre_affaire;
END;
END;
/
```

```
SQL>
SQL> alter type Chambre_Type
  2  add MEMBER FUNCTION ChiffreAffaire RETURN numeric
  3  cascade;

Type altered.

SQL> CREATE OR REPLACE TYPE BODY Chambre_Type AS
  2  MEMBER FUNCTION ChiffreAffaire RETURN numeric
  3  AS
  4  v_chiffre_affaire number;
  5  BEGIN
  6  SELECT SUM(PrixNuit * (r.DateDepart - r.DateArrivee)) INTO v_chiffre_affaire
  7  FROM Reservation r, Chambre c
  8  WHERE r.NumChambre = c.NumChambre AND c.NumChambre = self.NumChambre;
  9  RETURN v_chiffre_affaire;
 10  END;
 11  END;
 12  /

Type body created.
```

— Calculer pour chaque hôtel, le nombre d'évaluations reçues à une date donnée (01-01-2022)

```
--la signature
alter type Hotel_Type
add member function
nbEval(p_dateEvaluation DATE)
return numeric cascade;
--le corps
CREATE OR REPLACE TYPE BODY Hotel_Type
AS MEMBER FUNCTION nbCham
RETURN NUMERIC IS
nbrchambre NUMBER;
BEGIN
SELECT COUNT(*) INTO nbrchambre
FROM Chambre
WHERE NumHotel = self.NumHotel;
RETURN nbrchambre;
END;

MEMBER FUNCTION
nbEval(p_dateEvaluation DATE)
RETURN NUMERIC IS
NbEvaluations NUMBER;
BEGIN
SELECT COUNT(*) INTO NbEvaluations
FROM Evaluation e
WHERE e.NumHotel = self.NumHotel
AND e.DateEvaluation=p_dateEvaluation;
RETURN NbEvaluations;
END;
END;
/
```

```
SQL>
SQL> alter type Hotel_Type
  2  add member function nbEval(p_dateEvaluation DATE) return numeric
  3  cascade;

Type altered.

SQL> CREATE OR REPLACE TYPE BODY Hotel_Type AS
  2    MEMBER FUNCTION nbCham RETURN NUMERIC
  3    IS
  4      nbrchambre NUMBER;
  5    BEGIN
  6      SELECT COUNT(*) INTO nbrchambre
  7      FROM Chambre
  8      WHERE NumHotel = self.NumHotel;
  9      RETURN nbrchambre;
 10    END;
 11
 12    MEMBER FUNCTION nbEval(p_dateEvaluation DATE) RETURN NUMERIC
 13    IS
 14      NbEvaluations NUMBER;
 15    BEGIN
 16      SELECT COUNT(*) INTO NbEvaluations
 17      FROM Evaluation e
 18      WHERE e.NumHotel = self.NumHotel AND e.DateEvaluation = p_dateEvaluation;
 19      RETURN NbEvaluations;
 20    END;
 21  END;
 22  /

Type body created.
```

### 0.3.3 Définir les tables nécessaires à la base de données

#### Creation de la table Hotel

```
CREATE TABLE Hotel OF Hotel_Type (
CONSTRAINT pk_Hotel primary key(NumHotel),
CONSTRAINT CHK_ETOILES CHECK (Etoiles >= 1 AND Etoiles<=5)
)
nested table HotelClient store as table_HotelClient,
nested table HotelEvaluation store as table_HotelEvaluation,
nested table HotelChambre store as table_HotelChambre;
```

```
SQL>
SQL> CREATE TABLE Hotel OF Hotel_Type (
  2     CONSTRAINT pk_Hotel primary key(NumHotel),
  3     CONSTRAINT CHK_ETOILES CHECK (Etoiles >= 1 AND Etoiles <= 5)
  4  )
  5  nested table HotelClient store as table_HotelClient,
  6  nested table HotelEvaluation store as table_HotelEvaluation,
  7  nested table HotelChambre store as table_HotelChambre;

Table created.
```

#### Creation de la table Chambre

```
CREATE TABLE Chambre OF Chambre_Type (
constraint pk_Chambre primary key(NumChambre, NumHotel),
constraint fk_Chambre1 foreign key(ChambreHotel)
references Hotel,
CONSTRAINT CHK_TypeCHAMBRE CHECK (TypeChambre IN ('simple',
'double', 'triple', 'suite', 'autre'))
)
nested table ChambreReservation store as
table_ChambreReservation;
```

```
SQL>
SQL> CREATE TABLE Chambre OF Chambre_Type (
  2      constraint pk_Chambre primary key(NumChambre, NumHotel),
  3   constraint fk_Chambre1 foreign key(ChambreHotel) references Hotel,
  4      CONSTRAINT CHK_TypeCHAMBRE CHECK (TypeChambre IN ('simple', 'double', 'triple', 'suite', 'autre'))
  5   )
  6   nested table ChambreReservation store as table_ChambreReservation;

Table created.
```

### Creation de la table Client

```
CREATE TABLE CLIENT OF Client_Type (
constraint pk_Client primary key(NumClient)
)
nested table ClientReservation store
as table_ClientReservation,
nested table ClientEvaluation  store
as table_ClientEvaluation,
nested table ClientHotel store as table_ClientHotel;
```

```
SQL>
SQL> CREATE TABLE CLIENT OF Client_Type (
  2      constraint pk_Client primary key(NumClient)
  3   )
  4   nested table ClientReservation store as table_ClientReservation,
  5   nested table ClientEvaluation  store as table_ClientEvaluation,
  6   nested table ClientHotel store as table_ClientHotel;

Table created.
```

### Creation de la table Reservation

```
CREATE TABLE RESERVATION OF Reservation_Type (
constraint pk_Reservation PRIMARY KEY
(NumClient, NumHotel, DateArrivee),
CONSTRAINT fk_Reservation FOREIGN KEY (NumHotel)
REFERENCES HOTEL,
CONSTRAINT fk_Reservation1 FOREIGN KEY (ReservationClient)
REFERENCES CLIENT ,
CONSTRAINT fk_Reservation3 FOREIGN KEY (ReservationChambre)
REFERENCES Chambre,
CONSTRAINT CHK_DATE CHECK (DateArrivee > DateDepart)
);
```

```
SQL>
SQL> CREATE TABLE RESERVATION OF Reservation_Type (
  2      constraint pk_Reservation PRIMARY KEY (NumClient, NumHotel, DateArrivee ),
  3   CONSTRAINT fk_Reservation FOREIGN KEY (NumHotel) REFERENCES HOTEL,
  4   CONSTRAINT fk_Reservation1 FOREIGN KEY (ReservationClient) REFERENCES CLIENT ,
  5      CONSTRAINT fk_Reservation3 FOREIGN KEY (ReservationChambre) REFERENCES Chambre,
  6      CONSTRAINT CHK_DATE CHECK (DateArrivee > DateDepart)
  7   );

Table created.
```

### Creation de la table Evaluation

```
CREATE TABLE EVALUATION OF Evaluation_Type (
CONSTRAINT pk_Evaluation PRIMARY KEY
(NumHotel, NumClient, DateEvaluation),
CONSTRAINT fk_Evaluation3 FOREIGN KEY ( EvaluationClient)
REFERENCES CLIENT,
CONSTRAINT fk_Evaluation4 FOREIGN KEY ( EvaluationHotel)
REFERENCES Hotel,
CONSTRAINT CHK_Note CHECK (Note >= 1 AND Note  <= 10)
);
```

```
SQL>
SQL> CREATE TABLE EVALUATION OF Evaluation_Type (
  2      CONSTRAINT pk_Evaluation PRIMARY KEY (NumHotel, NumClient, DateEvaluation),
  3  CONSTRAINT fk_Evaluation3 FOREIGN KEY ( EvaluationClient) REFERENCES CLIENT,
  4  CONSTRAINT fk_Evaluation4 FOREIGN KEY ( EvaluationHotel) REFERENCES Hotel,
  5  CONSTRAINT CHK_Note CHECK (Note >= 1 AND Note  <= 10)
  6  );

Table created.
```

## 0.4 Langage de manipulation de données

### Remplissage de la table Hotel

```
INSERT INTO Hotel values(Hotel_Type(1,'Renaissance'
,'Tlemcen', 5,null,t_set_ref_Chambre(),
t_set_ref_Client(),t_set_ref_Evaluation()));
```

```
SQL> INSERT INTO Hotel values (Hotel_Type (1, 'Renaissance ', 'Tlemcen', 5,null,t_set_ref_Chambre(),t_set_ref_Client(),t_set_ref_Evaluation()));
1 row created.
SQL> INSERT INTO Hotel  values (Hotel_Type (2, 'Seybouse ', 'Annaba', 3,null,t_set_ref_Chambre(),t_set_ref_Client(),t_set_ref_Evaluation()));
1 row created.
SQL> INSERT INTO Hotel values (Hotel_Type (3, 'Hôtel Novotel ', 'Constantine', 4,null,t_set_ref_Chambre(),t_set_ref_Client(),t_set_ref_Evaluation()));
1 row created.
SQL> INSERT INTO Hotel  values (Hotel_Type (4, 'Saint George d''Alger', 'Alger', 5,null,t_set_ref_Chambre(),t_set_ref_Client(),t_set_ref_Evaluation()));
1 row created.
SQL> INSERT INTO Hotel values (Hotel_Type (5, 'Ibis Alger Aéroport', 'Alger', 2,null,t_set_ref_Chambre(),t_set_ref_Client(),t_set_ref_Evaluation()));
1 row created.
SQL> INSERT INTO Hotel values (Hotel_Type (6, 'El Mountazah Annaba', 'Annaba', 3,null,t_set_ref_Chambre(),t_set_ref_Client(),t_set_ref_Evaluation()));
1 row created.
SQL> INSERT INTO Hotel  values (Hotel_Type (7, 'Hôtel Albert 1er', 'Alger', 3,null,t_set_ref_Chambre(),t_set_ref_Client(),t_set_ref_Evaluation()));
1 row created.
```

### Remplissage de la table Client

```
INSERT INTO CLIENT VALUES (Client_Type (
1, 'BOUROUBI', 'Taous', null,t_set_ref_Reservation(),
t_set_ref_Evaluation(),t_set_ref_Hotel()));
```

```
SQL> INSERT INTO CLIENT VALUES (Client_Type (2, 'BOUZIDI', 'AMEL', null,t_set_ref_Reservation(),t_set_ref_Evaluation(),t_set_ref_Hotel()));

1 row created.

SQL> INSERT INTO CLIENT VALUES (Client_Type (3, 'LACHEMI', 'Bouzid',  null,t_set_ref_Reservation(),t_set_ref_Evaluation(),t_set_ref_Hotel()));

1 row created.

SQL> INSERT INTO CLIENT  VALUES (Client_Type (4, 'BOUCHEMLA', 'Elias',  null,t_set_ref_Reservation(),t_set_ref_Evaluation(),t_set_ref_Hotel()));

1 row created.

SQL> INSERT INTO CLIENT  VALUES (Client_Type (5, 'HADJ', 'Zouhir',  null,t_set_ref_Reservation(),t_set_ref_Evaluation(),t_set_ref_Hotel()));

1 row created.

SQL> INSERT INTO CLIENT  VALUES (Client_Type (6, 'OUSSEDIK', 'Hakim',  null,t_set_ref_Reservation(),t_set_ref_Evaluation(),t_set_ref_Hotel()));

1 row created.
```

### Remplissage de la table Chambre

```
INSERT INTO chambre  VALUES (Chambre_Type(
1, 4, 0, 'autre', 13000,(select ref(a) from HOTEL a
where NumHotel=4),t_set_ref_Reservation()));
```

### Remplissage de la table Reservation

```
INSERT INTO reservation VALUES (Reservation_Type(
1, 1, 5, '2022-05-15', '2022-05-11',
(select ref(a) from Client a where NumClient=1 and rownum=1),
(select ref(a) from Chambre a where NumChambre=1 and rownum=1),
(select ref(a) from Hotel a where NumHotel=5 and rownum=1)));
```

```
SQL> alter session set nls_date_format='RRRR-MM-DD HH24:MI:SS';

Session altered.

SQL> INSERT INTO reservation VALUES (Reservation_Type(1, 1, 5, '2022-05-15', '2022-05-11', (select ref(a) from Client a where NumClient=1 and rownum = 1),(select ref(a)
 from Chambre a where NumChambre=1 and rownum = 1),(select ref(a) from Hotel a where NumHotel=5 and rownum = 1)));

1 row created.

SQL> INSERT INTO reservation VALUES (Reservation_Type(2, 2, 5, '2022-04-18', '2022-04-11', (select ref(a) from Client a where NumClient=2 and rownum = 1),(select ref(a)
 from Chambre a where NumChambre=2 and rownum = 1),(select ref(a) from Hotel a where NumHotel=5 and rownum = 1)));

1 row created.

SQL> INSERT INTO reservation VALUES (Reservation_Type(6, 2, 5, '2022-04-06', '2022-04-05',(select ref(a) from Client a where NumClient=6 and rownum = 1),(select ref(a)
from Chambre a where NumChambre=2 and rownum = 1),(select ref(a) from Hotel a where NumHotel=5 and rownum = 1)));

1 row created.

SQL> INSERT INTO reservation VALUES (Reservation_Type(6, 30, 7, '2022-05-01', '2022-04-27',(select ref(a) from Client a where NumClient=6 and rownum = 1),(select ref(a)
 from Chambre a where NumChambre=30 and rownum = 1),(select ref(a) from Hotel a where NumHotel=7 and rownum = 1)));

1 row created.

SQL> INSERT INTO reservation VALUES (Reservation_Type(6, 100, 10, '2022-06-14', '2022-06-11',(select ref(a) from Client a where NumClient=6 and rownum = 1),(select ref(
a) from Chambre a where NumChambre=100 and rownum = 1),(select ref(a) from Hotel a where NumHotel=10 and rownum = 1)));

1 row created.

SQL> INSERT INTO reservation VALUES (Reservation_Type(13, 2, 6, '2022-05-10', '2022-05-02',(select ref(a) from Client a where NumClient=13 and rownum = 1),(select ref(a
) from Chambre a where NumChambre=2 and rownum = 1),(select ref(a) from Hotel a where NumHotel=6 and rownum = 1)));

1 row created.
```

### Remplissage de la table Evaluation

```
INSERT INTO Evaluation VALUES (Evaluation_Type(
5, 1, '2022-05-15', 4,
(select ref(a) from Client a where NumClient=1),
(select ref(b) from Hotel b where NumHotel=5)));
```

```
SQL> INSERT INTO Evaluation VALUES (Evaluation_Type(11, 29, '2022-04-08', 2,(select ref(a) from Client a where NumClient=29),(select ref(b) from Hotel b where NumHotel=
11)));

1 row created.

SQL> INSERT INTO Evaluation VALUES (Evaluation_Type(8, 37, '2022-05-10', 4,(select ref(a) from Client a where NumClient=37),(select ref(b) from Hotel b where NumHotel=8
)));

1 row created.

SQL> INSERT INTO Evaluation VALUES (Evaluation_Type(12, 29, '2022-05-09', 3,(select ref(a) from Client a where NumClient=29),(select ref(b) from Hotel b where NumHotel=
12)));

1 row created.

SQL> INSERT INTO Evaluation VALUES (Evaluation_Type(12, 35, '2022-05-06', 5,(select ref(a) from Client a where NumClient=35),(select ref(b) from Hotel b where NumHotel=
12)));

1 row created.

SQL> INSERT INTO Evaluation VALUES (Evaluation_Type(5, 35, '2022-05-23', 4,(select ref(a) from Client a where NumClient=35),(select ref(b) from Hotel b where NumHotel=5
)));

1 row created.

SQL> INSERT INTO Evaluation VALUES (Evaluation_Type(6, 35, '2022-06-02', 2,(select ref(a) from Client a where NumClient=35),(select ref(b) from Hotel b where NumHotel=6
)));

1 row created.

SQL> INSERT INTO Evaluation VALUES (Evaluation_Type(12, 37, '2022-05-18', 3,(select ref(a) from Client a where NumClient=37),(select ref(b) from Hotel b where NumHotel=
12)));

1 row created.
```

### Mis a jour de la table HotelClient

```
insert into table (select l.HotelClient from Hotel l
where NUMHOTEL=5) (select ref(c)
from CLIENT c where NUMCLIENT=1);
```

```
SQL> insert into table (select l.HotelClient from Hotel l where NUMHOTEL=11)
  2  (select ref(c) from CLIENT c where NUMCLIENT=29);

1 row created.

SQL> insert into table (select l.HotelClient from Hotel l where NUMHOTEL=8)
  2  (select ref(c) from CLIENT c where NUMCLIENT=37);

1 row created.

SQL> insert into table (select l.HotelClient from Hotel l where NUMHOTEL=12)
  2  (select ref(c) from CLIENT c where NUMCLIENT=29);

1 row created.

SQL> insert into table (select l.HotelClient from Hotel l where NUMHOTEL=12)
  2  (select ref(c) from CLIENT c where NUMCLIENT=35);

1 row created.

SQL> insert into table (select l.HotelClient from Hotel l where NUMHOTEL=5)
  2  (select ref(c) from CLIENT c where NUMCLIENT=35);

1 row created.

SQL> insert into table (select l.HotelClient from Hotel l where NUMHOTEL=6)
  2  (select ref(c) from CLIENT c where NUMCLIENT=35);

1 row created.

SQL> insert into table (select l.HotelClient from Hotel l where NUMHOTEL=12)
  2  (select ref(c) from CLIENT c where NUMCLIENT=37);

1 row created.
```

### Mis a jour de la table ClientReservation

```
insert into table (select l.ClientReservation from CLIENT l
where NUMCLIENT=1)
(select ref(c) from RESERVATION c where NUMCLIENT=1);
```

```
SQL> insert into table (select l.ClientReservation from CLIENT l where NUMCLIENT=2)
  2  (select ref(c) from RESERVATION c where NUMCLIENT=2);

1 row created.

SQL> insert into table (select l.ClientReservation from CLIENT l where NUMCLIENT=3)
  2  (select ref(c) from RESERVATION c where NUMCLIENT=3);

1 row created.

SQL> insert into table (select l.ClientReservation from CLIENT l where NUMCLIENT=4)
  2  (select ref(c) from RESERVATION c where NUMCLIENT=4);

0 rows created.

SQL> insert into table (select l.ClientReservation from CLIENT l where NUMCLIENT=5)
  2  (select ref(c) from RESERVATION c where NUMCLIENT=5);

0 rows created.

SQL> insert into table (select l.ClientReservation from CLIENT l where NUMCLIENT=6)
  2  (select ref(c) from RESERVATION c where NUMCLIENT=6);

3 rows created.

SQL> insert into table (select l.ClientReservation from CLIENT l where NUMCLIENT=7)
  2  (select ref(c) from RESERVATION c where NUMCLIENT=7);

1 row created.
```

### Mis a jour de la table HotelChambre

```
insert into table (select l.HotelChambre from hotel l
where NUMHOTEL=2)
(select ref(c) from chambre c where NUMCHAMBRE=1);
```

```
SQL> insert into table (select l.HotelChambre from hotel l where NUMHOTEL=9)
  2  (select ref(c) from chambre c where NUMCHAMBRE=1);

6 rows created.

SQL> insert into table (select l.HotelChambre from hotel l where NUMHOTEL=11)
  2  (select ref(c) from chambre c where NUMCHAMBRE=1);

6 rows created.

SQL> insert into table (select l.HotelChambre from hotel l where NUMHOTEL=2)
  2  (select ref(c) from chambre c where NUMCHAMBRE=2);

5 rows created.

SQL> insert into table (select l.HotelChambre from hotel l where NUMHOTEL=5)
  2  (select ref(c) from chambre c where NUMCHAMBRE=2);

5 rows created.

SQL> insert into table (select l.HotelChambre from hotel l where NUMHOTEL=6)
  2  (select ref(c) from chambre c where NUMCHAMBRE=2);

5 rows created.

SQL> insert into table (select l.HotelChambre from hotel l where NUMHOTEL=9)
  2  (select ref(c) from chambre c where NUMCHAMBRE=2);

5 rows created.

SQL> insert into table (select l.HotelChambre from hotel l where NUMHOTEL=11)
  2  (select ref(c) from chambre c where NUMCHAMBRE=2);

5 rows created.

SQL> insert into table (select l.HotelChambre from hotel l where NUMHOTEL=12)
  2  (select ref(c) from chambre c where NUMCHAMBRE=303);

2 rows created.
```

### Mis a jour de la table HotelEvaluation

```
insert into table (select l.HotelEvaluation from hotel l
where NUMHOTEL=1)
(select ref(c) from Evaluation c where NUMHOTEL=1);
```

```
SQL> insert into table (select l.HotelEvaluation from hotel l where NUMHOTEL=6)
  2  (select ref(c) from Evaluation c where NUMHOTEL=6);

4 rows created.

SQL> insert into table (select l.HotelEvaluation from hotel l where NUMHOTEL=7)
  2  (select ref(c) from Evaluation c where NUMHOTEL=7);

1 row created.

SQL> insert into table (select l.HotelEvaluation from hotel l where NUMHOTEL=8)
  2  (select ref(c) from Evaluation c where NUMHOTEL=8);

3 rows created.

SQL> insert into table (select l.HotelEvaluation from hotel l where NUMHOTEL=9)
  2  (select ref(c) from Evaluation c where NUMHOTEL=9);

3 rows created.

SQL> insert into table (select l.HotelEvaluation from hotel l where NUMHOTEL=10)
  2  (select ref(c) from Evaluation c where NUMHOTEL=10);

1 row created.

SQL> insert into table (select l.HotelEvaluation from hotel l where NUMHOTEL=11)
  2  (select ref(c) from Evaluation c where NUMHOTEL=11);

9 rows created.

SQL> insert into table (select l.HotelEvaluation from hotel l where NUMHOTEL=12)
  2  (select ref(c) from Evaluation c where NUMHOTEL=12);

3 rows created.
```

## 0.5 Langage d'interrogation de données

### 0.5.1 Lister les noms d'hôtels et leurs villes respectives.

```
SELECT CONCAT (NomHotel, Ville) FROM Hotel;
```

```
SQL> SELECT CONCAT (NomHotel, Ville) FROM Hotel;

CONCAT(NOMHOTEL,VILLE)
--------------------------------------------------
Renaissance Tlemcen
Seybouse Annaba
Hôtel Novotel Constantine
Saint George d'AlgerAlger
Ibis Alger AéroportAlger
El Mountazah AnnabaAnnaba
Hôtel Albert 1erAlger
Chems Oran
Colombe Oran
Mercure Alger
Le Méridien Oran

CONCAT(NOMHOTEL,VILLE)
--------------------------------------------------
Hôtel Sofitel Alger

12 rows selected.
```

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

```
select H.NOMHOTEL
from hotel H, table(H.HotelClient) c
group by H.NOMHOTEL;
```

```
SQL> select H.NOMHOTEL
  2  from hotel H, table(H.HotelClient) c
  3  group by H.NOMHOTEL;

NOMHOTEL
--------------------------------------------------
Le Méridien
Seybouse
Colombe
Saint George d'Alger
Chems
Ibis Alger Aéroport
Hôtel Albert 1er
Mercure
Renaissance
Hôtel Sofitel
El Mountazah Annaba

11 rows selected.
```

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

```
select CONCAT(c.NOMCLIENT, c.PRENOMCLIENT)
from client c, table(c.ClientReservation) r
where
deref(deref(value(r)).ReservationChambre).etage=1;

SELECT C.NumClient, C.NomClient, C.PrenomClient
FROM Client C
WHERE NOT EXISTS (
SELECT *
FROM Reservation R
INNER JOIN Chambre CH
ON R.ReservationChambre = REF(CH)
WHERE R.ReservationClient = REF(C) AND CH.Etage!=1
);
```

```
SQL> select CONCAT(c.NOMCLIENT, c.PRENOMCLIENT)
  2  from client c, table(c.ClientReservation) r
  3  where deref(deref(value(r)).ReservationChambre).etage=1;

CONCAT(C.NOMCLIENT,C.PRENOMCLIENT)
--------------------------------------------------------------------------------
BOUROUBITaous
BOUZIDIAMEL
OUSSEDIKHakim
OUSSEDIKHakim
ABBOUMohamed
ABDELAZIZAhmed
ABDELMOUMENNassima

7 rows selected.
```

**where deref(deref(value(r)).ReservationChambre).etage=1** : Cette clause spécifie une condition de filtrage pour la requête. La fonction "deref" est utilisée pour extraire la valeur réelle d'une référence à un objet. Dans ce cas, nous extrayons la valeur de la référence "r" et accédons ensuite à la colonne "ReservationChambre" de cette valeur. Ensuite, nous accédons à la colonne "etage" de la chambre associée à la réservation. La condition indique que seules les réservations ayant une chambre à l'étage 1 seront sélectionnées.

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

```
select h.NOMHOTEL, h.VILLE, deref(value(ch)).PrixNuit
from hotel h, table(h.HotelChambre) ch
where deref(value(ch)).TYPECHAMBRE='suite';
```

```
SQL> select h.NOMHOTEL, h.VILLE, deref(value(ch)).PrixNuit
  2  from hotel h, table(h.HotelChambre) ch
  3  where deref(value(ch)).TYPECHAMBRE='suite';

NOMHOTEL
--------------------------------------------------
VILLE                                          DEREF(VALUE(CH)).PRIXNUIT
---------------------------------------------------------------------------
Hôtel Sofitel
Alger                                                                19500
```

**from hotel h, table(h.HotelChambre) ch** : Cette partie spécifie les tables utilisées dans la requête. "hotel" est la table principale, et "h" est l'alias utilisé pour faire référence à cette table. "table(h.HotelChambre)" fait référence à une table de relation "HotelChambre" associée à la table "hotel". L'alias "ch" est utilisé pour faire référence à cette table de relation.

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

**FROM Reservation r JOIN Chambre c ON r.ReservationChambre = REF(c) JOIN Hotel h ON r.ReservationHotel = REF(h)** : Cette partie spécifie les tables utilisées dans la requête et les conditions de jointure. Nous joignons la table "Reservation" avec la table "Chambre" en utilisant la colonne "ReservationChambre" et avec la table "Hotel" en utilisant la colonne "ReservationHotel".

```
SELECT h.NomHotel, c.TypeChambre, COUNT(*)
AS Nombre_de_reservations
FROM Reservation r
JOIN Chambre c ON r.ReservationChambre = REF(c)
JOIN Hotel h ON r.ReservationHotel = REF(h)
WHERE h.Ville = 'Alger'
GROUP BY h.NomHotel, c.TypeChambre
ORDER BY h.NomHotel, Nombre_de_reservations DESC;
```

```
SQL> SELECT h.NomHotel, c.TypeChambre, COUNT(*) AS Nombre_de_reservations
  2  FROM Reservation r
  3  JOIN Chambre c ON r.ReservationChambre = REF(c)
  4  JOIN Hotel h ON r.ReservationHotel = REF(h)
  5  WHERE h.Ville = 'Alger'
  6  GROUP BY h.NomHotel, c.TypeChambre
  7  ORDER BY h.NomHotel, Nombre_de_reservations DESC;

NOMHOTEL
--------------------------------------------------
TYPECHAMBRE                                        NOMBRE_DE_RESERVATIONS
-------------------------------------------------- ----------------------
Hôtel Albert 1er
simple                                                                  1

Hôtel Sofitel
simple                                                                  2

Hôtel Sofitel
double                                                                  1


NOMHOTEL
--------------------------------------------------
TYPECHAMBRE                                        NOMBRE_DE_RESERVATIONS
-------------------------------------------------- ----------------------
Ibis Alger Aéroport
simple                                                                  5

Mercure
simple                                                                  1

Saint George d'Alger
simple                                                                  1


6 rows selected.
```

### 0.5.6 Quels sont les hôtels (nom, ville) ayant obtenu une moyenne de notes >=6, durant l'année 2022

```
SELECT h.NOMHOTEL, h.VILLE
FROM hotel h, TABLE(h.HotelEvaluation) he
where EXTRACT(YEAR FROM
deref(VALUE(he)).DateEvaluation) = 2022
GROUP BY h.NOMHOTEL, h.VILLE
HAVING AVG(deref(VALUE(he)).note) >= 6;

SELECT h.NOMHOTEL, h.VILLE
FROM hotel h, TABLE(h.HotelEvaluation) he
where EXTRACT(YEAR FROM
deref(VALUE(he)).DateEvaluation) = 2022
GROUP BY h.NOMHOTEL, h.VILLE
HAVING AVG(deref(VALUE(he)).note) >= 5;
```

```
SQL> SELECT h.NOMHOTEL, h.VILLE
  2  FROM hotel h, TABLE(h.HotelEvaluation) he
  3  where EXTRACT(YEAR FROM deref(VALUE(he)).DateEvaluation) = 2022
  4  GROUP BY h.NOMHOTEL, h.VILLE
  5  HAVING AVG(deref(VALUE(he)).note) >= 6;

no rows selected
```

```
SQL> SELECT h.NOMHOTEL, h.VILLE
  2  FROM hotel h, TABLE(h.HotelEvaluation) he
  3  where EXTRACT(YEAR FROM deref(VALUE(he)).DateEvaluation) = 2022
  4  GROUP BY h.NOMHOTEL, h.VILLE
  5  HAVING AVG(deref(VALUE(he)).note) >= 5;

NOMHOTEL
--------------------------------------------------
VILLE
--------------------------------------------------
Hôtel Albert 1er
Alger

Saint George d'Alger
Alger

Seybouse
Annaba
```

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

```
SELECT h.NomHotel, SUM(c.PrixNuit * (r.DateDepart -
r. DateArrivee)) AS Chiffre_daffaire
FROM Reservation r
JOIN Chambre c ON r.ReservationChambre = REF(c)
JOIN Hotel h ON r.ReservationHotel = REF(h)
WHERE EXTRACT(MONTH FROM r.DateArrivee) IN (6,7,8)
GROUP BY h.NomHotel
ORDER BY Chiffre_daffaire DESC;
```

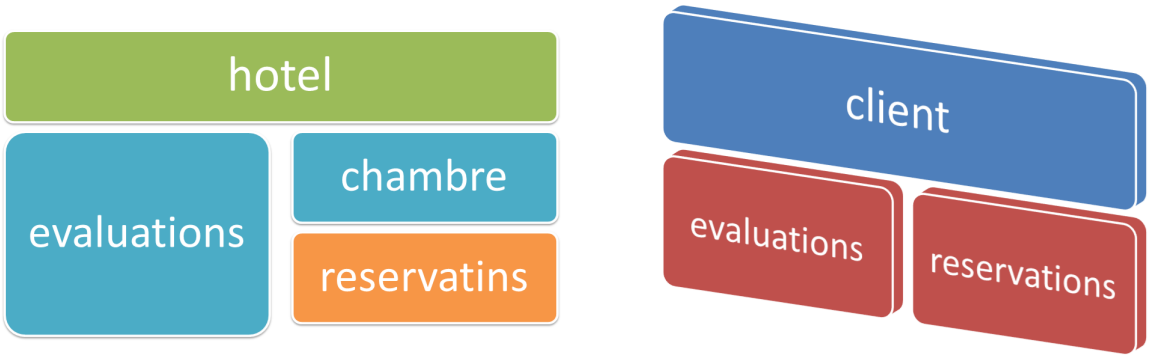
```
SQL> SELECT h.NomHotel, SUM(c.PrixNuit * (r.DateDepart -r. DateArrivee)) AS Chiffre_daffaire
  2  FROM Reservation r
  3  JOIN Chambre c ON r.ReservationChambre = REF(c)
  4  JOIN Hotel h ON r.ReservationHotel = REF(h)
  5  WHERE EXTRACT(MONTH FROM r.DateArrivee) IN (6, 7, 8)
  6  GROUP BY h.NomHotel
  7  ORDER BY Chiffre_daffaire DESC;

NOMHOTEL                                           CHIFFRE_DAFFAIRE
-------------------------------------------------- ----------------
Mercure                                                      -8700
Ibis Alger Aéroport                                         -13500
El Mountazah Annaba                                         -28800
```

# Deuxième partie

# NoSQL – Modèle orienté « documents »

## 0.6 Modélisation orientée document

### 0.6.1 Proposer une modélisation orientée document de la base de données




### 0.6.2 Illustrez la modélisation sur un exemple de la BD fournie


```
"hotel":{
  "numHotel": 1,
  "nomHotel": "Renaissance",
  "ville": "Tlemcen",
  "etoiles": 5,
  "siteWeb": "https://www.hotelabc.com",
  "evaluations": [
    {
      "numClient": 1,
      "date": "2023-05-10",
      "note": 4.5
    },
    {
      "numClient": 2,
      "date": "2023-05-09",
      "note": 3.0
    }
  ],
  "chambre":[
      "NUMCHAMBRE": 12,
      "ETAGE": 2,
      "TYPECHAMBRE": "simple",
      "PRIXNUIT": 7700,
      "reservations": [
        {
          "numClient": 1,
          "DATEARRIVEE": "2023-05-10",
          "DATEDEPART": "2023-05-10",
          "NUMCHAMBRE": 12
        }
    ]
  ]
}
```

```
"client":{
  "numClient": 37,
  "nomClient": "BELABES",
  "prenomClient": "Abdelkader",
  "email": "pierre.dupont@mail.com",
  "evaluations": [
    {
      "numHotel": 1,
      "date": "2023-05-10",
      "note": 4.5
    }
  ],
  "reservations": [
    {
      "numHotel": 1,
      "DATEARRIVEE": "18-05-2022",
      "DATEDEPART": "11-05-2022",
      "NUMCHAMBRE": 202
    }
  ]
}
```


### 0.6.3 Justifiez le choix de conception

1. Les documents imbriqués reflètent les relations : Dans cette modélisation, les documents sont imbriqués les uns dans les autres pour refléter les relations entre les entités, par exemple les évaluations associées à un hôtel ou à un client. Cette modélisation imbriquée est intuitive et facilite la récupération des données relatives à ces relations.
2. Les performances sont optimisées : Avec cette modélisation, les requêtes peuvent être optimisées en utilisant des index sur les champs de recherche. Par exemple, si l'on souhaite récupérer toutes les évaluations associées à un hôtel donné, on peut créer un index sur le champ "numHotel" pour accélérer la recherche.

3. Les données sont dénormalisées : Avec cette modélisation, les données sont dénormalisées pour éviter les jointures coûteuses en temps de traitement. Par exemple, les informations sur les évaluations sont dupliquées dans les documents correspondant à chaque hôtel et à chaque client, évitant ainsi les jointures entre les tables "HOTEL", "CLIENT" et "EVALUATION".

## 0.7 Remplir la base de données

### 0.7.1 Exemple d'insertion dans le document Hotel

```
document=({
        "numHotel": 2,
        "nomHotel": "Seybouse",
        "ville": "Annaba",
        "etoiles": 3,
        "siteWeb": "",
        "evaluations": [
        {
                "numClient": 2,
                "date": "2023-08-10",
                "note": 3
        },
        {
                "numClient": 1,
                "date": "2023-08-10",
                "note": 6.5
        },
        {
                "numClient": 21,
                "date": "2023-09-09",
                "note": 6.0
        }
        ],
        "chambre":[
        {
                "NUMCHAMBRE": 10,
                "ETAGE": 1,
                "TYPECHAMBRE": "simple",
                "PRIXNUIT": 4500,
                "reservations": [
                {
                        "numClient": 1,
                        "DATEARRIVEE": "2023-08-15",
                        "DATEDEPART": "2023-08-10",
                        "NUMCHAMBRE": 10
```

```
                        },
                        {
                                "numClient": 2,
                                "DATEARRIVEE": "2023-01-20",
                                "DATEDEPART": "2023-01-10",
                                "NUMCHAMBRE": 10
                        }
                        ]
                },
                {
                        "NUMCHAMBRE": 2,
                        "ETAGE": 1,
                        "TYPECHAMBRE": "simple",
                        "PRIXNUIT": 4800,
                        "reservations": [
                        {
                                "numClient": 1,
                                "DATEARRIVEE": "2023-05-10",
                                "DATEDEPART": "2023-05-1",
                                "NUMCHAMBRE": 2
                        },
                        {
                                "numClient": 2,
                                "DATEARRIVEE": "2023-05-10",
                                "DATEDEPART": "2023-04-3",
                                "NUMCHAMBRE": 2
                        }
                        ]
                },
                {
                        "NUMCHAMBRE": 3,
                        "ETAGE": 1,
                        "TYPECHAMBRE": "double",
                        "PRIXNUIT": 7100,
                        "reservations": [
                        {
                                "numClient": 1,
                                "DATEARRIVEE": "2023-05-10",
                                "DATEDEPART": "2023-05-6",
                                "NUMCHAMBRE": 3
                        },
                        {
                                "numClient": 2,
                                "DATEARRIVEE": "2023-12-11",
```

```
                        "DATEDEPART": "2023-12-10",
                        "NUMCHAMBRE": 3
                }
                ]
        },
        {
                "NUMCHAMBRE": 4,
                "ETAGE": 1,
                "TYPECHAMBRE": "simple",
                "PRIXNUIT": 5400,
                "reservations": [
                {
                        "numClient": 1,
                        "DATEARRIVEE": "2023-05-10",
                        "DATEDEPART": "2023-05-10",
                        "NUMCHAMBRE": 4
                },
                {
                        "numClient": 2,
                        "DATEARRIVEE": "2023-03-30",
                        "DATEDEPART": "2023-03-10",
                        "NUMCHAMBRE": 4
                }
                ]
        },
        {
                "NUMCHAMBRE": 5,
                "ETAGE": 2,
                "TYPECHAMBRE": "double",
                "PRIXNUIT": 8600,
                "reservations": [
                {
                        "numClient": 1,
                        "DATEARRIVEE": "2023-02-21",
                        "DATEDEPART": "2023-02-10",
                        "NUMCHAMBRE": 5
                },
                {
                        "numClient": 2,
                        "DATEARRIVEE": "2023-09-10",
                        "DATEDEPART": "2023-09-8",
                        "NUMCHAMBRE": 5
                }
                ]
```

```
                        },
                        {
                                "NUMCHAMBRE": 6,
                                "ETAGE": 2,
                                "TYPECHAMBRE": "simple",
                                "PRIXNUIT": 5800,
                                "reservations": [
                                {
                                        "numClient": 1,
                                        "DATEARRIVEE": "2023-03-10",
                                        "DATEDEPART": "2023-05-5",
                                        "NUMCHAMBRE": 6
                                },
                                {
                                        "numClient": 2,
                                        "DATEARRIVEE": "2023-06-17",
                                        "DATEDEPART": "2023-06-10",
                                        "NUMCHAMBRE": 6
                                }
                                ]
                        },
                        {
                                "NUMCHAMBRE": 7,
                                "ETAGE": 2,
                                "TYPECHAMBRE": "simple",
                                "PRIXNUIT": 5800,
                                "reservations": [
                                {
                                        "numClient": 40,
                                        "DATEARRIVEE": "2022-05-13",
                                        "DATEDEPART": "2022-05-09",
                                        "NUMCHAMBRE": 7
                                },
                                {
                                        "numClient": 24,
                                        "DATEARRIVEE": "2023-05-20",
                                        "DATEDEPART": "2023-05-10",
                                        "NUMCHAMBRE": 7
                                }
                                ]
                        },
                        {
                                "NUMCHAMBRE": 8,
                                "ETAGE": 2,
```

```
                        "TYPECHAMBRE": "double",
                        "PRIXNUIT": 8600,
                        "reservations": [
                        {
                                "numClient": 40,
                                "DATEARRIVEE": "2022-05-05",
                                "DATEDEPART": "2022-05-01",
                                "NUMCHAMBRE": 8
                        },
                        {
                                "numClient": 24,
                                "DATEARRIVEE": "2023-05-20",
                                "DATEDEPART": "2023-05-10",
                                "NUMCHAMBRE": 8
                        }
                        ]
                },
                {
                        "NUMCHAMBRE": 10,
                        "ETAGE": 3,
                        "TYPECHAMBRE": "suite",
                        "PRIXNUIT": 5800,
                        "reservations": [
                        {
                                "numClient": 40,
                                "DATEARRIVEE": "2022-05-13",
                                "DATEDEPART": "2022-05-09",
                                "NUMCHAMBRE": 10
                        },
                        {
                                "numClient": 24,
                                "DATEARRIVEE": "2023-05-20",
                                "DATEDEPART": "2023-05-10",
                                "NUMCHAMBRE": 10
                        }
                        ]
                }
                ]
        })

    db.hotel.insert(document)
```

### 0.7.2 Exemple d'insertion dans le document Client

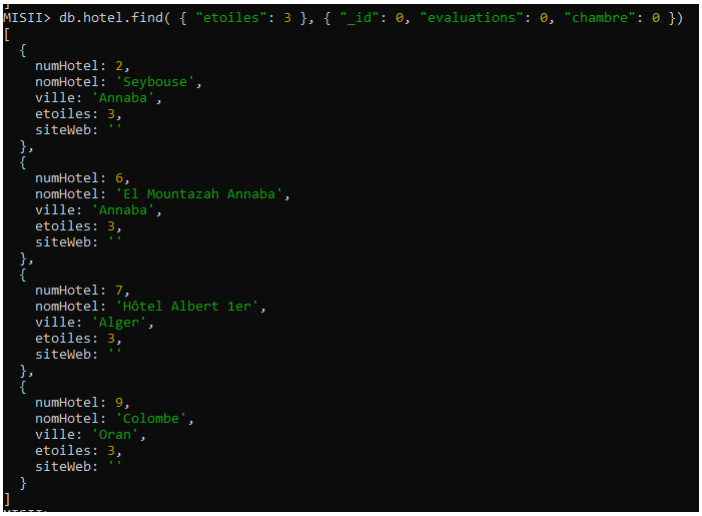
```
            document=({
                    "numClient": 1,
                    "nomClient": "BOUROUBI",
                    "prenomClient": "Taous",
                    "email": "taous.dupont@mail.com",
                    "evaluations": [
                    {
                            "numHotel": 1,
                            "date": "2023-05-10",
                            "note": 4.5
                    }
                    ],
                    "reservations": [
                    {
                            "numHotel": 1,
                            "DATEARRIVEE": "18-05-2022",
                            "DATEDEPART": "11-05-2022",
                            "NUMCHAMBRE": 202
                    }
                    ]
            })
    db.client.insert(client)
```

## 0.8 Répondre aux requêtes

### 0.8.1 Creation de la BD et les collections


### 0.8.2 Afficher tous les hôtels classés « 3 étoiles »

```
            db.hotel.find(
            {"etoiles": 3},
            {"_id":0, "evaluations": 0, "chambre": 0})
```




### 0.8.3 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. Afficher le contenu de la collection.

```
db.hotel.aggregate([
{
        $project: {
                _id: 0,
                nomHotel: 1,
                reservations: "$chambre.reservations"
        }
},
{
        $unwind: "$reservations"
},
{
        $group: {
                _id: "$nomHotel",
                nbReservations: { $sum: 1 }
        }
},
{
        $sort: { nbReservations: -1 }
},
{
        $out: "HotelsNbResv"
}
])

db.HotelsNbResv.find()
```

```
MISII> db.hotel.aggregate([ { $project: { _id: 0, nomHotel: 1, reservations: "$chambre.reservations" } }, { $unwind: "$reservations" }, { $group: { _i
MISII> db.HotelsNbResv.find()
[
  { _id: 'Hôtel Sofitel ', nbReservations: 34 },
  { _id: 'Le Méridien', nbReservations: 19 },
  { _id: 'Renaissance', nbReservations: 14 },
  { _id: 'Hôtel Albert 1er', nbReservations: 13 },
  { _id: 'Chems', nbReservations: 10 },
  { _id: "Saint George d'Alger", nbReservations: 9 },
  { _id: 'Seybouse', nbReservations: 9 },
  { _id: 'El Mountazah Annaba', nbReservations: 6 },
  { _id: 'Mercure', nbReservations: 5 },
  { _id: 'Hôtel Novotel ', nbReservations: 5 },
  { _id: 'Colombe', nbReservations: 4 },
  { _id: 'Ibis Alger Aéroport', nbReservations: 3 }
]
```

unwind : Cette étape est utilisée pour dérouler ou "déplier" un tableau dans une collection en plusieurs documents distincts. Dans ce cas, le champ "reservations" qui a été créé précédemment est déroulé, de sorte que chaque réservation devient un document distinct avec le même contenu que les autres champs.

### 0.8.4 Dans une collection HotelsPas-cher, récupérer les hôtels dont le prix des chambres ne dépasse pas 6000 DA. Afficher le contenu de la collection

```
db.hotel.aggregate([
{
        $unwind: "$chambre"
},
{
        $match: {
                "chambre.PRIXNUIT": {$lte: 6000}
        }
},
{
        $group: {
                _id: "$nomHotel",
                ville: { $first: "$ville" },
                siteWeb: { $first: "$siteWeb" },
                etoiles: { $first: "$etoiles" },
                evaluations: { $first: "$evaluations" },
                chambre: { $push: "$chambre" }
        }
},
{
        $out: "HotelsPasCher"
}
])

db.HotelsPasCher.find()
```

```
MISII> db.hotel.aggregate([ { $unwind: "$chambre" }, { $match: { "chambre.PRIXNUIT": { $lte: 6000 } } }, { $group: { _id: "$nomHotel", ville: { $first: "$ville" }, site
Web: { $first: "$siteWeb" }, etoiles: { $first: "$etoiles" }, evaluations: { $first: "$evaluations" }, chambre: { $push: "$chambre" } } }, { $out: "HotelsPasCher" }])

MISII> db.HotelsPasCher.find({},{"evaluations":0, "chambre":0})
[
  {
    _id: 'El Mountazah Annaba',
    ville: 'Annaba',
    siteWeb: '',
    etoiles: 3
  },
  { _id: 'Hôtel Albert 1er', ville: 'Alger', siteWeb: '', etoiles: 3 },
  { _id: 'Chems', ville: 'Oran', siteWeb: '', etoiles: 2 },
  { _id: 'Colombe', ville: 'Oran', siteWeb: '', etoiles: 3 },
  {
    _id: "Saint George d'Alger",
    ville: 'Alger',
    siteWeb: '',
    etoiles: 5
  },
  { _id: 'Mercure', ville: 'Alger', siteWeb: '', etoiles: 4 },
  { _id: 'Seybouse', ville: 'Annaba', siteWeb: '', etoiles: 3 },
  {
    _id: 'Ibis Alger Aéroport',
    ville: 'Alger',
    siteWeb: '',
    etoiles: 2
  },
  {
    _id: 'Hôtel Novotel ',
    ville: 'Constantine',
    siteWeb: '',
    etoiles: 4
  }
]
```

### 0.8.5 Afficher tous les noms d'hôtels ayant obtenu une note moyenne >=5

```
db.hotel.aggregate([
{
        $unwind: "$evaluations"
},
{
        $group: {
                _id: "$nomHotel",
                noteMoyenne: { $avg: "$evaluations.note" }
        }
},
{
        $match: {
                noteMoyenne: { $gte: 5 }
        }
},
{
        $project: {
                _id: 0,
                nomHotel: "$_id",
                noteMoyenne: 1
        }
}
])
```

```
MISII> db.hotel.aggregate([ { $unwind: "$evaluations" }, { $group: { _id: "$nomHotel", noteMoyenne: { $avg: "$evaluations.note" } } }, { $match: { noteMoyenne: { $gte:
5 } } }, { $project: { _id: 0, nomHotel: "$_id", noteMoyenne: 1 } }])
[
  { noteMoyenne: 5.166666666666667, nomHotel: 'Seybouse' },
  { noteMoyenne: 5.166666666666667, nomHotel: "Saint George d'Alger" }
]
```

### 0.8.6 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.

```
db.client.aggregate([
{
        $match: { email: "Mourad@mail.com" }
},
{
        $lookup: {
                from: "hotel",
                localField: "reservations.numHotel",
                foreignField: "numHotel",
                as: "hotel_reservations"
        }
},
{
        $project: {
                _id: 0,
                "reservations.DATEARRIVEE": 1,
                "hotel_reservations.nomHotel": 1,
                "reservations.NUMCHAMBRE": 1
        }
}
])
```

```
MISII> db.client.aggregate([ { $match: { email: "Mourad@gmail.com" } }, { $lookup: { from: "hotel", localField: "reservations.numHotel", foreignField:
hotel_reservations" } }, { $project: { _id: 0, "reservations.DATEARRIVEE": 1, "hotel_reservations.nomHotel": 1, "reservations.NUMCHAMBRE": 1 } }])
[
  {
    reservations: [ { DATEARRIVEE: '18-05-2022', NUMCHAMBRE: 202 } ],
    hotel_reservations: [ { nomHotel: 'Renaissance' }, { nomHotel: 'Renaissance' } ]
  }
]
```

lookup : Cette étape est utilisée pour effectuer une jointure entre la collection actuelle ("client") et une autre collection ("hotel") en utilisant une condition de correspondance entre les champs spécifiés. Dans ce cas, la jointure est réalisée entre le champ "reservations.numHotel" de la collection "client" et le champ "numHotel" de la collection "hotel". Les résultats de la jointure sont stockés dans le champ "hotelreservations".

### 0.8.7 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.

```
db.client.aggregate([
{
        $match: { email: "Mourad@mail.com" }
```

```
        },
        {
                $lookup: {
                        from: "hotel",
                        localField: "evaluations.numHotel",
                        foreignField: "numHotel",
                        as: "hotel_evaluations"
                }
        },
        {
                $project: {
                        _id: 0,
                        "evaluations.date": 1,
                        "hotel_evaluations.nomHotel": 1,
                        "evaluations.note": 1
                }
        }
        ])
```

```
MISII> db.client.aggregate([ { $match: { email: "Mourad@gmail.com" } }, { $lookup: { from: "hotel", localField: "evaluations.numHotel", foreignField:
otel_evaluations" } }, { $project: { _id: 0, "evaluations.date": 1, "hotel_evaluations.nomHotel": 1, "evaluations.note": 1 } }])
[
  {
    evaluations: [ { date: '2023-05-10', note: 1 } ],
    hotel_evaluations: [ { nomHotel: 'Ibis Alger Aéroport' } ]
  }
]
```

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

```
db.hotel.updateMany(
{ etoiles: 5 },
{ $inc: { "chambre.$[].PRIXNUIT": 2000 } }
)
```

```
MISII> db.hotel.updateMany(
...   { etoiles: 5 },
...   { $inc: { "chambre.$[].PRIXNUIT": 2000 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 6,
  modifiedCount: 6,
  upsertedCount: 0
}
MISII>
```

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

```
var mapFunction = function() {
        emit(this.nomHotel, 1);
```

```
};

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

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

db.result.find().pretty();
```


— mapFunction : Cette fonction de mappage est utilisée pour émettre une clé et une valeur pour chaque document dans la collection "hotel". Dans ce cas, la clé émise est la valeur du champ "nomHotel" du document, et la valeur émise est simplement le nombre 1.
— reduceFunction : Cette fonction de réduction est utilisée pour réduire les valeurs associées à une clé donnée. Dans ce cas, la fonction somme toutes les valeurs pour une clé donnée.
— db.hotel.mapReduce() : Cette méthode est utilisée pour appeler la fonction mapReduce sur la collection "hotel". Les arguments passés sont la fonction de mappage, la fonction de réduction et un objet de configuration. L'option out spécifie le nom de la collection de sortie où les résultats seront stockés.
— db.result.find().pretty() : Cette commande est utilisée pour récupérer et afficher les résultats de l'opération mapReduce à partir de la collection "result".

## 0.9 Analyse

1. Afficher tous les hôtels classés « 3 étoiles » : cette requête est en lien avec la collection "hotels" et ne nécessite pas de modification de la conception proposée.
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. Afficher le contenu de la collection. : cette requête implique la création d'une nouvelle collection, "Hotels-NbResv", qui stockera des informations agrégées sur les réservations. Cette collection n'a pas été prévue dans la conception proposée, mais il est possible de l'ajouter. Il faut également prévoir un script de mise à jour de cette collection chaque fois qu'une réservation est effectuée ou annulée.
3. Dans une collection HotelsPas-cher, récupérer les hôtels dont le prix des chambres ne dépasse pas 6000 DA. Afficher le contenu de la collection. : cette requête implique la création d'une nouvelle collection, "HotelsPas-cher", qui stockera les hôtels dont le prix des chambres ne dépasse pas un certain seuil. Cette collection n'a pas été prévue dans la conception proposée, mais il est possible de l'ajouter. Il faut également prévoir un script de mise à jour de cette collection chaque fois qu'un prix de chambre est modifié.
4. Afficher tous les noms d'hôtels ayant obtenu une note moyenne >=5 : cette requête est en lien avec la collection "hotels" et ne nécessite pas de modification de la conception proposée.
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. : cette requête est en lien avec la collection "clients" et nécessite une jointure avec la collection "hotels". La conception proposée prévoit déjà une référence à l'hôtel dans chaque réservation, ce qui facilite la réalisation de cette requête.
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. : cette requête est en lien avec la collection "clients" et nécessite une jointure avec la collection "hotels". La conception proposée prévoit déjà une référence à l'hôtel dans chaque évaluation, ce qui facilite la réalisation de cette requête.
7. Augmenter de 2000DA, le prix unitaire de toutes les chambres des hôtels classés « 5 étoiles » : cette requête implique la modification de la collection "hotels" en augmentant le prix de toutes les chambres des hôtels classés "5 étoiles". La conception proposée prévoit déjà une référence au nombre d'étoiles de chaque hôtel, ce qui facilite la réalisation de cette requête.

# Conclusion

Au final, nous avons acquis une compréhension approfondie de l'utilisation des bases de données relationnelles (SQL3) et NoSQL (MongoDB). Nous avons pu expérimenter différentes requêtes et interagir avec les bases de données, ce qui nous a permis de constater les avantages et les différences entre les deux approches.
Cette expérience nous a fourni des connaissances pratiques sur la façon de travailler avec chaque type de base de données et nous a aidés à prendre des décisions éclairées en fonction des besoins spécifiques de nos projets.
Il est important de noter que le choix entre SQL3 et MongoDB dépend des spécificités de votre application et de vos besoins en matière de stockage et de manipulation des données. Les bases de données relationnel