ISTY

Implémentation d'une base de données de gestion location de véhicules multi-agences

Encadré par :
Beatrice FINANCE

Présenté par :
Iheb BESBES
Mohamed ilyes EL AJROUD
Sami FAKHFAKH

Introduction

Ce rapport introduit les différentes étapes d'implémentation d'une base données de gestion de location de véhicules d'une compagnie multi-agences. on va détailler dans les différents chapitres le cahier de charges , l'implémentation passant par l'insertion de jeu de données de test .

La base de données traite d'un côté la majorité du processus de location des véhicules commençant par le choix de véhicule, le contrat de location, le retour de véhicule et la facturation .D'un autre côté, la base prend en charge l'interchangeabilité entre différents agences du même groupes. Ceci offre une meilleure gestion du Park de véhicules ainsi qu'une variété d'options pour le client final.

La base de données est implémentée en utilisant ORACLE ENTREPRISE.

Table des matières

Table des Figures

Table des Tables

I. Cahier de charges ;

a. Présentation du projet

Notre projet consiste à implémenter une base de données dédiée à la gestion de location d'un parc de véhicules. La base implémentée est partagée entre plusieurs agences de locations réparties principalement en ile de France mais dans différents départements.

b. Scénario principal :

Un client s'adresse à une agence de location du groupe pour louer un véhicule. Un opérateur le prend en charge (l'opérateur n'est pas géré dans la base) et commence par remplir un contrat (représenté par la table contrat).

Le client choisit le véhicule souhaité parmi le catalogue des véhicules (table véhicule) selon les disponibilités et l'agence.

Le client choisit ensuite le type de contrat .La base de données prend en considération deux types de forfaits :

- **forfait kilométrage limité** : Le client est limité par un nombre de kilomètres précisé dans le contrat (km limite) pendant la période d'emprunt.

-**forfait kilométrage illimité** : Le client n'est pas limité par un nombre de kilomètres précis.

Le client choisit ensuite la date de départ, la date de retour du véhicule ainsi que l'agence de retour.

L'opérateur prend l'immatriculation du véhicule souhaité ainsi que les coordonnés du client (table client).

Notons que le kilométrage de départ véhicule est géré AUTOMATIQUEMENT par notre base : si le

Dès l'enregistrement du contrat dans la base une facture (table facture) provisoire (qui change au fur et à mesure) sera généré AUTOMATIQUEMENT.

Lors du retour du véhicule une fiche retour (table fiche retour) est remplie contenant (attributs table ficheretour).

En se basant sur les différents champs remplis et le type de contrat, la facture provisoire se met à jour et donne le total facture final.

c. Modalités :

- La table véhicule est partiellement saisie manuellement. le prix journalier, le dépôt de garantie et le prix km est calculé automatiquement selon une formule mathématique (que nous avons établie) qui prend en considération l'ancienneté du véhicule, sa puissance et l'usure (nombre de kilomètres parcourus). Ces prix sont mis à jour automatiquement. La table client représente tout les clients de la

base Toutes les agences sont répertoriées dans la table agence et identifiable par un code agence unique.

- on a considéré que le prix du carburant est fixe et précisé dans le contrat.
- Si un client fait un retard lors du retour de véhicule, un supplément va être facturé et le prix journalier majoré de 10% (applicable aussi sur le prix du kilomètre avec une majoration 5%).
- si le client ne retourne pas le véhicule avec un plein de carburant un supplément va être facturé selon le nombre de litres manquants.
- si le client a retourné le véhicule accidenté, un pourcentage de réparation est établi et sera déduit du dépôt de garantie (le processus de réparation n'est pas pris en considération dans la base mais sera considéré comme axe d'amélioration.
- Si le client a retourné le véhicule dans une agence différente que celle citée lors de la signature du contrat, il sera facturé d'un frais supplémentaire fixe de 700 euros.
- La gestion de disponibilités des véhicules se fait par l'intermédiaire d'un opérateur ou un dispatcheur. Notre base de données calcul **les disponibilités approximatives** des véhicules en prenant en considération que les dates prévus (table contrat)
- Chaque agence ajoute une commission à la facture total du client qui a réservé chez elle .Cette commission ne dépasse les 11%

On a essayé de rapprocher notre projet le plus que possible à la réalité cependant il reste plusieurs axes d'améliorations envisageable et certaines conditions non prises en charge par la base par manque de connaissances sur le système réel du système de location de véhicule.

d. modèle MCD

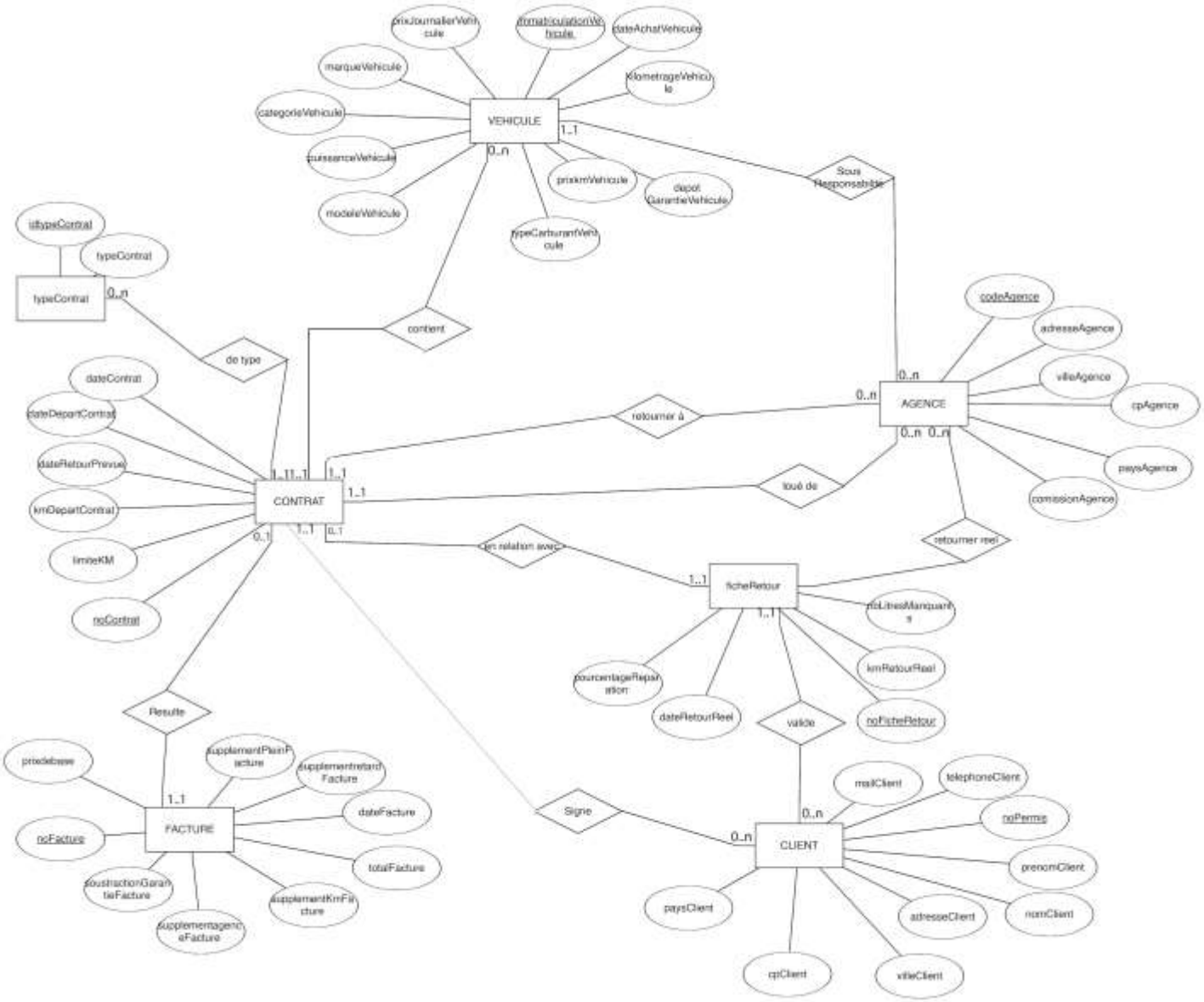

Figure 1 : modèle MCD de la base de données

II. Droit d'accès

La gestion des droits d'accès est très importante dans une base de données. Une faille de droits d'accès dans une base peut engendrer dans certain cas des conduites frauduleuses et même la faillite de l'entreprise.

On définit 4 groupes d'utilisateurs dans notre base :

- **Manager :** Ce groupe est dédié pour les directeurs et chef d'agences

-**responsable :** Ce groupe peut contient les différents responsables des agences

-**finance** : ce groupe dédié spécialement pour le département de la facturation

-**collaborateur** : ce groupe englobe les opérateurs les dispatcheurs et les commerciaux de l'agence

-**client** : ce groupe est dédié pour le Front -end et accessible par le client à tout moment

Tableau 1 droits d'accès à la base

Groupes / Droits	client	finance	collaborateur	responsable	Manager
TABLE CONTRAT					
Ajout			X	X	X
Accès et consultation			X	X	X
Modification annulation			X	X	X
suppression					X
TABLE FACTURE					
Ajout		X		X	X
Accès et consultation		X	X	X	X
Modification annulation		X		X	X
suppression		X			X
TABLE CLIENT					
Ajout		X	X	X	X
Accès et consultation		X	X	X	X
Modification annulation		X	X	X	X
suppression		X			X
TABLE FICHE RETOUR					
Ajout			X	X	X
Accès et consultation		X	X	X	X
Modification annulation		X	X	X	X
suppression					X
TABLE AGENCE					
Ajout				X	X
Accès et consultation		X	X	X	X
Modification annulation					X
suppression					X
TABLE VEHICULE					
Ajout				X	X
Accès et consultation		X	X	X	X
Modification annulation				X	X
suppression					X
TABLE TYPE CONTRAT					
Ajout		X		X	X
Accès et consultation		X	X	X	X
Modification annulation				X	X
suppression					X

Le droit de suppression est très délicat .Si on a ce droit on peut supprimer toutes les lignes d'une tables .C'est pour cela que ce privilège est donné que pour les directeurs.

III. Création de la base de données :

Dans ce chapitre on va entamer la création de notre base de données en se basant sur le modèle MCD.

Afin de faciliter la compréhension ci-dessous le modèle relationnel :

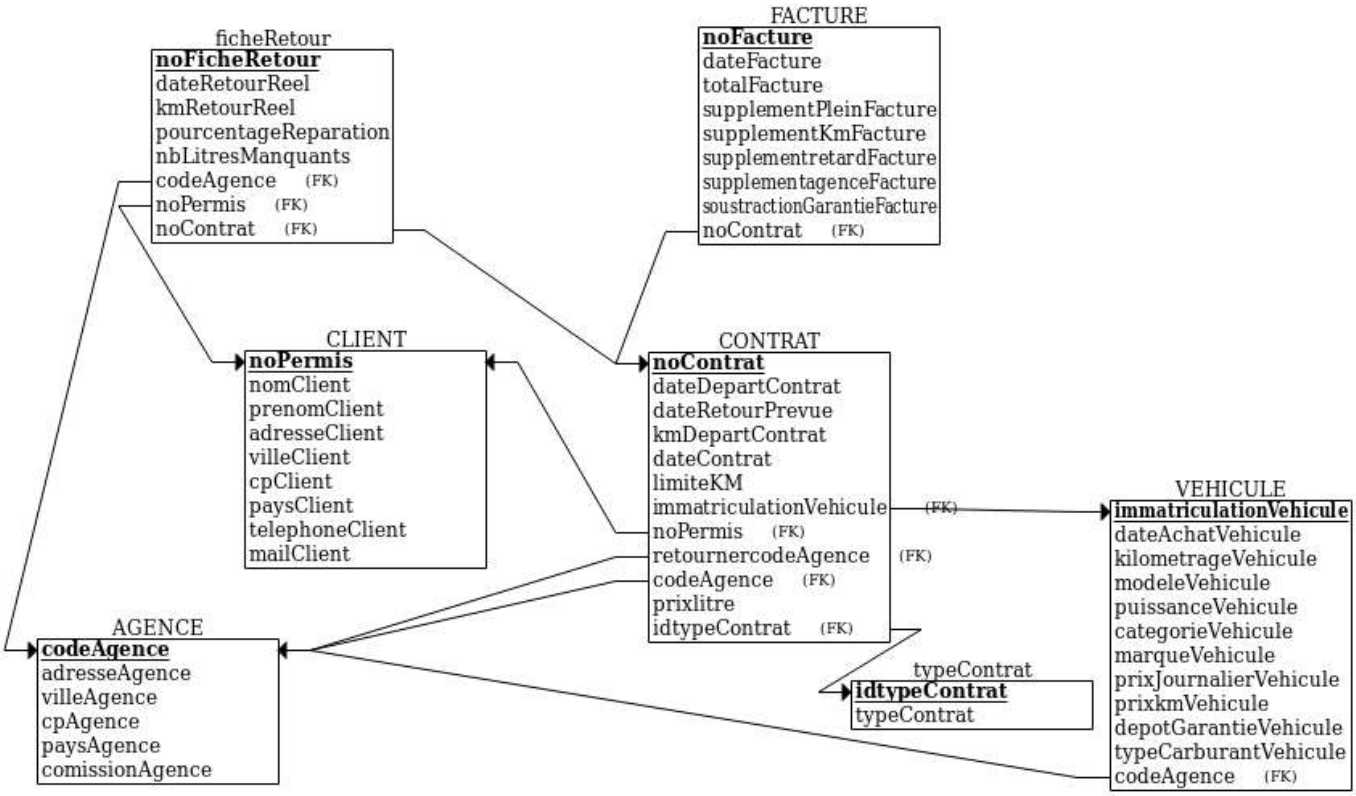

Figure 2 modéle relationnel de la base de données

a. Création des tables :

```
CREATE TABLE client
  (
     nopermis         NUMBER(10) NOT NULL,
     nomclient        VARCHAR(60) NOT NULL,
     prenomclient     VARCHAR(60) NOT NULL,
     adresseclient    VARCHAR(60) NOT NULL,
     villeclient      VARCHAR(60) NOT NULL,
```

```
    cpclient         VARCHAR(60) NOT NULL,
    paysclient       VARCHAR(60) NOT NULL,
    telephoneclient  VARCHAR(60) NOT NULL,
    mailclient       VARCHAR(60) NOT NULL
  );


CREATE TABLE agence
  (
    codeagence       VARCHAR(60) NOT NULL,
    adresseagence    VARCHAR(60) NOT NULL,
    villeagence      VARCHAR(60) NOT NULL,
    cpagence         VARCHAR(60) NOT NULL,
    comissionagence  NUMBER(10) NOT NULL
  );

CREATE TABLE vehicule
  (
    immatriculationvehicule  VARCHAR(60) NOT NULL,
    dateachatvehicule        DATE NOT NULL,
    kilometragevehicule      NUMBER(10) NOT NULL,
    modelevehicule           VARCHAR(60) NOT NULL,
    puissancevehicule        NUMBER(10) NOT NULL,
    categorievehicule        VARCHAR(60) NOT NULL,
    marquevehicule           VARCHAR(60) NOT NULL,
    prixjournaliervehicule   NUMBER(10, 2) NOT NULL,
    prixkmvehicule           NUMBER(10, 2) NOT NULL,
    depotgarantievehicule    NUMBER(10, 2) NOT NULL,
```

```
    typecarburantvehicule   VARCHAR(60) NOT NULL,
    codeagence              VARCHAR(60) NOT NULL
  );

CREATE TABLE facture
  (
    nofacture                   NUMBER(10) NOT NULL,
    datefacture                 DATE NOT NULL,
    totalfacture                NUMBER(10, 2) NOT NULL,
    prixdebase                  NUMBER(10, 2) NOT NULL,
    supplementpleinfacture      NUMBER(10, 2) NOT NULL,
    supplementkmfacture         NUMBER(10, 2) NOT NULL,
    supplementretardfacture     NUMBER(10, 2) NOT NULL,
    supplementagencefacture     NUMBER(10, 2) NOT NULL,
    soustractiongarantiefacture NUMBER(10, 2) NOT NULL,
    nocontrat                   NUMBER(10) NOT NULL
  );
CREATE TABLE ficheretour
  (
    noficheretour         NUMBER(10) NOT NULL,
    dateretourreel        DATE NOT NULL,
    kmretourreel          NUMBER(10) NOT NULL,
    pourcentagereparation NUMBER(10) NOT NULL,
    nblitresmanquants     NUMBER(10) NOT NULL,
    codeagence            VARCHAR(60) NOT NULL,
    nopermis              NUMBER(10) NOT NULL,
    nocontrat             NUMBER(10) NOT NULL
  );
```

```
CREATE TABLE contrat
  (
     nocontrat                NUMBER(10) NOT NULL,
     datedepartcontrat        DATE NOT NULL,
     dateretourprevue         DATE NOT NULL,
     kmdepartcontrat          NUMBER(10) NOT NULL,
     datecontrat              DATE NOT NULL,
     limitekm                 NUMBER(10),
     immatriculationvehicule  VARCHAR(60) NOT NULL,
     nopermis                 NUMBER(10) NOT NULL,
     codeagence               VARCHAR(60) NOT NULL,
     retournercodeagence      VARCHAR(60) NOT NULL,
     prixlitre                NUMBER(10, 2) NOT NULL,
     idtypecontrat            NUMBER(10) NOT NULL
  );


CREATE TABLE typecontrat
  (
     idtypecontrat NUMBER(10) NOT NULL,
     typecontrat   VARCHAR(60) NOT NULL
  );
```

b. Création des contraintes des clé primaires

```
ALTER TABLE client
  ADD CONSTRAINT cleprimaiteclient PRIMARY KEY (nopermis);


ALTER TABLE agence
```

```
  ADD CONSTRAINT cleprimaiteagence PRIMARY KEY (codeagence);


ALTER TABLE vehicule
  ADD CONSTRAINT cleprimairevehicule PRIMARY KEY (immatriculationvehicule);


ALTER TABLE contrat
  ADD CONSTRAINT cleprimairecontrat PRIMARY KEY (nocontrat);


ALTER TABLE facture
  ADD CONSTRAINT cleprimairefacture PRIMARY KEY (nofacture);


ALTER TABLE ficheretour
  ADD CONSTRAINT cleprimairefiche PRIMARY KEY (noficheretour);


ALTER TABLE typecontrat
  ADD CONSTRAINT cleprimairetypecontrat PRIMARY KEY (idtypecontrat);
```

1. création de contraintes des clés étrangères

```
ALTER TABLE ficheretour
  ADD CONSTRAINT cleetrangerefiche FOREIGN KEY (codeagence) REFERENCES
agence(
  codeagence);


ALTER TABLE ficheretour
  ADD CONSTRAINT cleetrangerefiche1 FOREIGN KEY (nopermis) REFERENCES
client(
  nopermis);


ALTER TABLE ficheretour
  ADD CONSTRAINT cleetrangerefiche2 FOREIGN KEY (nocontrat) REFERENCES
contrat(
```

```
  nocontrat);


ALTER TABLE vehicule

  ADD CONSTRAINT cleetrangerevehicule FOREIGN KEY (codeagence) REFERENCES
agence

  (codeagence);


ALTER TABLE contrat

  ADD CONSTRAINT cleetrangerecontrat1 FOREIGN KEY (immatriculationvehicule)

  REFERENCES vehicule(immatriculationvehicule);


ALTER TABLE contrat

  ADD CONSTRAINT cleetrangerecontrat2 FOREIGN KEY (nopermis) REFERENCES
client(

  nopermis);


ALTER TABLE contrat

  ADD CONSTRAINT cleetrangerecontrat4 FOREIGN KEY (codeagence) REFERENCES
agence

  (codeagence);


ALTER TABLE contrat

  ADD CONSTRAINT cleetrangerecontrat5 FOREIGN KEY (retournercodeagence)

  REFERENCES agence(codeagence);


ALTER TABLE contrat

  ADD CONSTRAINT cleetrangeretypecontrat FOREIGN KEY (idtypecontrat)
REFERENCES

  typecontrat(idtypecontrat);
```

```
ALTER TABLE facture

  ADD CONSTRAINT cleetrangerefacture FOREIGN KEY (nocontrat) REFERENCES
contrat

  (nocontrat);
```

IV. Contrôle et gestion de la base de données :

Dans ce paragraphe on va détailler les différents mécanismes mis en place pour assurer la consistance et la fiabilité de notre base.

a. Mise en place de contraintes d'intégrités

```
ALTER TABLE contrat ADD CONSTRAINT dates CHECK ( datedepartcontrat >= datecontrat

AND         datedepartcontrat <= dateretourprevue );ALTER TABLE typecontrat ADD
CONSTRAINT veriftypecontrat CHECK ( typecontrat IN ('kilometre',


'jour'));ALTER TABLE vehicule ADD CONSTRAINT verifcarbur CHECK
(typecarburantvehicule IN ('essence',


'diesel'));ALTER TABLE agence ADD CONSTRAINT commission CHECK ( comissionagence
BETWEEN 0 AND 11);
```

b. Génération séquentielle des clés primaires

On a choisi les tables facture, contrat et ficheRetour pour intégrer un trigger qui permet d'insérer automatiquement la clé primaire (en écrasant si nécessaire la clé primaire erronée lors de l'insertion d'un tuple) en s'appuyant sur des séquences .

ci-dessous le code sql des séquences :

- Séquence de la clé noContrat de la table contrat

```
BEGIN

    DECLARE

        seqval contrat.nocontrat%TYPE;

    BEGIN

        SELECT Max(nocontrat)

        INTO   seqval

        FROM   contrat;


        EXECUTE IMMEDIATE(
```

```
        'CREATE SEQUENCE seq_contrat MINVALUE 201000000 START WITH '||
        seqval||' INCREMENT BY 1 CACHE 20');
    END;
END;

/
CREATE OR replace TRIGGER trig_nocontrat
  BEFORE INSERT ON contrat
  FOR EACH ROW
BEGIN
    SELECT seq_contrat.NEXTVAL
    INTO   :new.nocontrat
    FROM   dual;
END;

/
```

- Séquence de la clé noFicheRetour de la table ficheRetour

```
BEGIN
    DECLARE
        seqval ficheretour.noficheretour%TYPE;
    BEGIN
        SELECT Max(noficheretour)
        INTO   seqval
        FROM   ficheretour;

        EXECUTE IMMEDIATE(
        'CREATE SEQUENCE seq_FICHERETOUR MINVALUE 5000 START WITH '||
        seqval||' INCREMENT BY 1 CACHE 20');
    END;
END;
```

```
/

CREATE OR replace TRIGGER trig_noficheretour

  BEFORE INSERT ON ficheretour

  FOR EACH ROW

BEGIN

    SELECT seq_ficheretour.NEXTVAL

    INTO   :new.noficheretour

    FROM   dual;

END;


/
```

- Séquence de la clé noFacture de la table facture

```
CREATE SEQUENCE seq_facture

MINVALUE 1

 START WITH 1

 INCREMENT BY 1

  CACHE 20;
```

Le trigger d'insertion de la clé primaire de la table facture a été intégré précédemment dans le trigger premfact quand va définir dans le prochain paragraphe.

c. Intégration de triggers

La plupart des méthodes de contrôle et de saisie automatique ont été faite par l'intermédiaire des triggers.

```
-- Verification si le vehicule est deja loue (en utilisant l'immatriculation et
dates) :

CREATE

OR

replace TRIGGER kmcompteurcontrat beforeINSERT

on CONTRAT for each row

DECLARE kmcompteur vehicule.kilometragevehicule%type;

carb vehicule.typecarburantvehicule%type;tcont typecontrat.typecontrat%type;controw
contrat%rowtype;testdisp number(2):=0;CURSOR cont ISSELECT *
```

```
  FROM   contrat
  WHERE  immatriculationvehicule = :NEW.immatriculationvehicule;BEGIN
  OPEN cont ;
  loop
  FETCH cont
  INTO  controw;

  IF cont%notfound then
  EXIT;
ENDIF;IF controw.datedepartcontrat >= :NEW.datedepartcontrat
AND
controw.datedepartcontrat <= :NEW.dateretourprevue then
testdisp := 1;ENDIF ;IF controw.dateretourprevue >= :NEW.datedepartcontrat
AND
controw.dateretourprevue <= :NEW.dateretourprevue then
testdisp := 1;ENDIF ;IF testdisp = 1 then
EXIT;ENDIF ;END
loop;CLOSE cont;IF testdisp = 1 then
raise_application_error (-20011, 'vehicule indisponible');ENDIF;
-- Exception si le vehicule est deja loue
-- Lors de l'ajout d'un vehicule dans le contrat, son kilometrage est ajouté
aussi.SELECT kilometragevehicule
INTO   kmcompteur
FROM   vehicule
WHERE
immatriculationvehicule=:NEW.immatriculationvehicule;:NEW.KMDEPARTCONTRAT:=kmcompte
ur;
-- Definition du prix du litre de plein dans le contrat suivant le type de
carburant
SELECT typecarburantvehicule
INTO   carb
```

```
FROM   vehicule

WHERE  immatriculationvehicule=:NEW.immatriculationvehicule;IF carb = 'essence'
then

:NEW.prixlitre := 1.8;

ELSE :NEW.prixlitre := 1.6;ENDIF;

-- Verification de la presence du kilometrage du vehicule de retour si le type de
contrat est kilometreSELECT typecontrat

INTO   tcont

FROM   typecontrat

WHERE  idtypecontrat = :NEW.idtypecontrat;IF ( tcont ='kilometre'

AND

(

  :NEW.limitekm = 0

  OR

  :NEW.limitekm IS NULL

)

) then

raise_application_error (-20003, 'viol integrité table CONTRAT
(immatriculationVehicule)');ENDIF ;END;/

-- Calcul du prix de base (prix estime de la location) lors de la signature d'un
contrat

-- Le prix total sera calcule lorsque la fiche de retour du vehicule est signe
(trigger calfact et factfinal)

CREATE

OR

replace TRIGGER premfact afterINSERT

on contrat for each row

DECLARE prixj vehicule.prixjournaliervehicule%type;

prixk vehicule.prixkmvehicule%type;tcont typecontrat.typecontrat%type;totfact
facture.totalfacture%type;id facture.nofacture%type;BEGIN

  --Extraction du type de contrat

  SELECT typecontrat
```

```
    INTO   tcont
    FROM   typecontrat
    WHERE  idtypecontrat = :NEW.idtypecontrat;

    -- dans le cas ou le type de contrat est jour
    IF tcont = 'jour' then
    -- Extraction du prix journalier d'un vehicule
    SELECT prixjournaliervehicule
    INTO   prixj
    FROM   vehicule
    WHERE  immatriculationvehicule = :NEW.immatriculationvehicule;

    -- Calcul du prix estime de la location
    totfact := prixj * ( :NEW.dateretourprevue - :NEW.datedepartcontrat );
    else
    -- dans le cas d'un contrat kilometre
    -- Extraction du prix de kilometre
    SELECT prixkmvehicule
    INTO   prixk
    FROM   vehicule
    WHERE  immatriculationvehicule = :NEW.immatriculationvehicule;

    -- Calcul du prix estime de la location
    totfact := prixk * (:NEW.limitekm - :NEW.kmdepartcontrat);
ENDIF;
-- Recuperation de la valeur suivante de la sequence
SELECT seq_facture.nextval
INTO   id
FROM   dual;
```

```
-- Creation du tuple contenant les details de la factureINSERT INTO facture VALUES
            (
                        id,
                        sysdate,
                        totfact,
                        0,0,0,0,0,
                        :NEW.nocontrat,
                        totfact
            );END;/
-- Calcul des supplements a rajouter dans la facture lors
-- de la signature de la fiche de retour d'un vehicule
CREATE
OR
replace TRIGGER factfinal beforeINSERT
on ficheretour for each row
DECLARE tcont typecontrat.typecontrat%type;
datret contrat.dateretourprevue%type;kmret contrat.kmdepartcontrat%type;prixj
vehicule.prixjournaliervehicule%type;prixk vehicule.prixkmvehicule%type;prixl
contrat.prixlitre%type;nvprix facture.supplementretardfacture%type;nvprixk
facture.supplementkmfacture%type;nvprixl
facture.supplementpleinfacture%type;cagence agence.codeagence%type;depgar
vehicule.depotgarantievehicule%type;nvgar
facture.soustractiongarantiefacture%type;nocont
ficheretour.noficheretour%type;BEGIN
  -- Mise a jour du nouveau kilometrage d'un vehicule lors de son retour
  UPDATE vehicule
  SET    kilometragevehicule = :NEW.kmretourreel
  WHERE  immatriculationvehicule =
         (
                SELECT immatriculationvehicule
                FROM   contrat
                WHERE  nocontrat = :NEW.nocontrat );
```

```
-- Extraction type contrat
SELECT typecontrat
INTO   tcont
FROM   typecontrat c ,
       contrat cc
WHERE  cc.idtypecontrat = c.idtypecontrat
AND    cc.nocontrat = :NEW.nocontrat ;

-- dans le cas ou le type de contrat est kilometre
-- Calcul exces kilometres si existant
IF tcont = 'kilometre' then
-- selection du kilometrage prevu du vehicule
SELECT limitekm
INTO   kmret
FROM   contrat
WHERE  nocontrat = :NEW.nocontrat;

-- Dans le cas d'un exces
IF kmret < :NEW.kmretourreel then
-- selection du prix du kilometre en exces
SELECT prixkmvehicule
INTO   prixk
FROM   vehicule v,
       contrat c
WHERE  v.immatriculationvehicule = c.immatriculationvehicule
AND    c.nocontrat = :NEW.nocontrat;

-- calcul du suppelemnt kilometre
```

```
  nvprixk := prixk * (1.05) * (:NEW.kmretourreel - kmret);
  -- mise a jour de la table facture par le nouvel supplement
  UPDATE facture
  SET    supplementkmfacture = nvprixk
  WHERE  nocontrat = :NEW.nocontrat;

ENDIF ;ENDIF;
-- selection de la date retour prevu du vehiculeSELECT dateretourprevue
INTO   datret
FROM   contrat
WHERE  nocontrat = :NEW.nocontrat;

-- dans le cas de retardIF datret < :NEW.dateretourreel then
-- selection du prix journalier du vehicule
SELECT prixjournaliervehicule
INTO   prixj
FROM   vehicule v,
       contrat c
WHERE  v.immatriculationvehicule = c.immatriculationvehicule
AND    c.nocontrat = :NEW.nocontrat;

-- calcul supplement retardnvprix := prixj * (1.1) * ( :NEW.dateretourreel -
datret);
-- mise a jour de la table facture par le nouvel supplementUPDATE facture
SET    supplementretardfacture = nvprix
WHERE  nocontrat = :NEW.nocontrat;ENDIF;
-- Dans le cas ou le reservoir du vehicule rendu est non pleinIF
:NEW.nblitresmanquants > 0 then
-- Selection du prix du litre
SELECT prixlitre
```

```
INTO   prixl
FROM   contrat
WHERE  nocontrat = :NEW.nocontrat;

-- Calcul du suppelement carburantnvprixl := :NEW.nblitresmanquants * prixl ;
-- mise a jour de la table facture par le nouvel supplementUPDATE facture
SET    supplementpleinfacture = nvprixl
WHERE  nocontrat = :NEW.nocontrat;ENDIF;
-- Selection du code agence ou le vehicule est renduSELECT retournercodeagence
INTO   cagence
FROM   contrat
WHERE  nocontrat = :NEW.nocontrat;

--  dans le cas ou le vehicule est rendu dans une agence differente du contratIF
cagence <> :NEW.codeagence then
-- mise a jour de la table facture par le nouvel supplement
UPDATE facture
SET    supplementagencefacture = 700
WHERE  nocontrat = :NEW.nocontrat;

-- mise a jour de la table vehicule par la nouvelle position du vehiculeUPDATE
vehicule
SET    codeagence = cagence
WHERE  immatriculationvehicule =
       (
              SELECT immatriculationvehicule
              FROM   contrat
              WHERE  nocontrat = :NEW.nocontrat );ENDIF;
-- selection du depot de garantie du vehiculeSELECT depotgarantievehicule
INTO   depgar
```

```
FROM   vehicule v ,
       contrat c
WHERE  v.immatriculationvehicule = c.immatriculationvehicule
AND    c.nocontrat = :NEW.nocontrat;


-- Calcul du cout de repartion suivant le pourcentage du degatnvgar := (depgar *
:NEW.pourcentagereparation / 100);
-- mise a jour de la table facture par le nouvel supplementUPDATE facture
SET    soustractiongarantiefacture = nvgar
WHERE  nocontrat = :NEW.nocontrat;END;/
-- Calcul du total facture lors de la mise a jour de la table facture
-- par un/des suppelements
CREATE
OR
replace TRIGGER calfact beforeUPDATE
on facture for each rowDECLAREBEGIN
  :NEW.totalfacture := :NEW.prixdebase + :NEW.supplementpleinfacture +
:NEW.supplementkmfacture + :NEW.supplementretardfacture +
:NEW.supplementagencefacture + :NEW.soustractiongarantiefacture;
end;/
--comission agence ne doit pas depasser 11%
-- la date d'achat de voiture ne doit pas depasser 2014
CREATE
OR
replace TRIGGER dateachatveh beforeINSERT
or
UPDATE
on vehicule FOR each row BEGIN
IF ( :NEW.dateachatvehicule < to_date('01-JAN-14', 'DD-MON-YY') ) THEN
raise_application_error (-20011, 'Date vehicule ilegale  (dateAchatVehicule)');
endIF ;END;/
```

```
-- a tester

--trigger foreign key

--
********************************************************************************

-- table CONTRAT

-- Triggers de verification de l'existance de la cle etrangere

-- en tant que cle primaire dans la table "mere"

CREATE

OR

replace TRIGGER verif_contrat beforeINSERT

or

UPDATE

on contrat FOR each row

DECLARE

CURSOR immatveh IS

  SELECT immatriculationvehicule

  FROM   vehicule;immat contrat.immatriculationvehicule%type;testImmat number(2) :=
0;CURSOR npermis ISSELECT nopermis

  FROM   client;permis contrat.nopermis%type;testPermis number(2) := 0;CURSOR
cagence ISSELECT codeagence

  FROM   agence;agencec agence.codeagence%type;testagence  number(2) :=
0;testagence2 number(2) := 0;CURSOR tcontr ISSELECT idtypecontrat

  FROM   typecontrat;contr typecontrat.idtypecontrat%type;testcont number(2) :=
0;BEGIN

  -- Ouverture du curseur les cle primaire de la table "mere"

  OPEN tcontr;

  loop

  FETCH tcontr

  INTO  contr;


  IF tcontr%notfound then

  EXIT;
```

```
ENDIF;

-- Test : cle etrangere égale t elle la cle primaire

-- Si oui la cle etrangere existe bien dans la table "mere"

-- Une variable de test changera et la boucle s'arreteraIF (:NEW.idtypecontrat =
contr) then

testcont := 1;ENDIF ;IF (testcont =1 ) then

EXIT;ENDIF;END

loop;CLOSE tcontr;

-- Si la variable de test reste inchange =>Cle absente exceptionIF testcont = 0
then

raise_application_error (-20003, 'viol integrité table CONTRAT
(idtypeContrat)');ENDIF;OPEN immatveh;LOOPFETCH immatveh

INTO  immat;IF immatveh%notfound then

EXIT;ENDIF;IF (:NEW.immatriculationvehicule = immat) then

testimmat := 1;ENDIF ;IF (testimmat =1 ) then

EXIT;ENDIF;END

loop;CLOSE immatveh;IF testimmat = 0 then

raise_application_error (-20003, 'viol integrité table CONTRAT
(immatriculationVehicule)');ENDIF;OPEN npermis;LOOPFETCH npermis

INTO  permis;IF npermis%notfound then

EXIT;ENDIF;IF (:NEW.nopermis = permis) then

testpermis := 1;ENDIF ;IF (testpermis =1 ) then

EXIT;ENDIF;END

loop;CLOSE npermis;IF testpermis = 0 then

raise_application_error (-20004, 'viol integrité table CONTRAT
(noPermis)');ENDIF;OPEN cagence;LOOPFETCH cagence

INTO  agencec;IF cagence%notfound then

EXIT;ENDIF;IF (:NEW.retournercodeagence = agencec) then

testagence := 1;ENDIF ;IF (:NEW.codeagence = agencec) then

testagence2 := 1;ENDIF ;IF (testagence =1

AND

testagence2=1 ) then
```

```
EXIT;ENDIF;END

loop;CLOSE cagence;IF testagence = 0 then

raise_application_error (-20003, 'viol integrité table CONTRAT
(retournercodeAgence)');ENDIF;IF testagence2 = 0 then

raise_application_error (-20003, 'viol integrité table CONTRAT
(codeAgence)');ENDIF;END;/

--
********************************************************************************

-- table VERICULE

CREATE

OR

replace TRIGGER verif_vehicule beforeINSERT

or

UPDATE

on vehicule FOR each row

DECLARE

CURSOR cgence IS

  SELECT codeagence

  FROM   agence;agence vehicule.codeagence%type;testagence number(2) := 0;BEGIN

  SELECT codeagence

  INTO   agence

  FROM   agence

  WHERE  codeagence = :NEW.codeagence;


  IF agence <> :NEW.code

  OPEN cgence;

  loop

  FETCH cgence

  INTO  agence;


  IF cgence%notfound then
```

```
  EXIT;
ENDIF;IF (:NEW.codeagence = agence) then
testagence := 1;ENDIF ;IF (testagence =1 ) then
EXIT;ENDIF;END
loop;CLOSE cgence;IF testagence = 0 then
raise_application_error (-20008, 'viol integrité table VEHICULE
(codeAgence)');ENDIF;END;/
--
*********************************************************************************
***********************************
CREATE
OR
replace TRIGGER verif_ficheretour beforeINSERT
or
UPDATE
on ficheretour FOR each row
DECLARE
CURSOR npermis IS
  SELECT nopermis
  FROM   client;permis contrat.nopermis%type;testPermis number(2) := 0;CURSOR
cagence ISSELECT codeagence
  FROM   agence;agencec agence.codeagence%type;testagence number(2) := 0;CURSOR
ncont ISSELECT nocontrat
  FROM   contrat;cont contrat.nocontrat%type;testcont number(2) := 0;BEGIN
  OPEN ncont;
  loop
  FETCH ncont
  INTO  cont;

  IF ncont%notfound then
  EXIT;
ENDIF;IF (:NEW.nocontrat = cont) then
```

```
testcont := 1;ENDIF ;IF (testcont =1 ) then

EXIT;ENDIF;END

loop;CLOSE ncont;IF testcont = 0 then

raise_application_error (-20004, 'viol integrité table CONTRAT
(noContrat)');ENDIF;OPEN npermis;LOOPFETCH npermis

INTO  permis;IF npermis%notfound then

EXIT;ENDIF;IF (:NEW.nopermis = permis) then

testpermis := 1;ENDIF ;IF (testpermis =1 ) then

EXIT;ENDIF;END

loop;CLOSE npermis;IF testpermis = 0 then

raise_application_error (-20004, 'viol integrité table CONTRAT
(noPermis)');ENDIF;OPEN cagence;LOOPFETCH cagence

INTO  agencec;IF cagence%notfound then

EXIT;ENDIF;IF (:NEW.codeagence = agencec) then

testagence := 1;ENDIF ;IF (testagence =1) then

EXIT;ENDIF;END

loop;CLOSE cagence;IF testagence = 0 then

raise_application_error (-20003, 'viol integrité table CONTRAT
(retournercodeAgence)');ENDIF;END;/
```

V. Implémentation et Génération du jeu de données :

a. Génération du jeu de données:

Le jeu de données est essentiel pour vérifier si notre base de données fait bien ce qu'il supposé faire.

Néanmoins, ce jeu de données doit être bien choisi et vérifie bien les contraintes d'intégrités cité ci-dessus.

On a utilisé différents sources pour assembler notre base de données :

1. générateur automatique de données de test www.databasetestdata.com : cet outil nous permis de générer notre liste de client avec les différents attributs voulus.
2. Base de données de station Vélib en ile de France référence site : opendata.paris.fr : Afin de s'approcher au plus près de la réalité on a préféré prendre des coordonnées réelles en ile de France pour représenter nos agences. En effet on a pris des vraies adresses de stations Vélib en ile de France et on les est considéré comme étant nos agences de location
3. Base de données de véhicules. On a pris une base de données de véhicules avec de vraies marques de voiture ainsi que de vrais modèles. On a affiné cette base en ajoutant une immatriculation de véhicule française (En respectant la norme européenne). Le type et la puissance de véhicule sont mis au hasard
4. Pour la génération des intervalles de dates ,les dates de contrats , les dates de retours réel, les kilométrages de de véhicules , on a préféré utiliser des scripts en python afin de garantir qu'on a bien des dates 'Logique' et non pas au hasard . Ceci s'applique aussi au kilométrage où on a essayé au maximum de se rapprocher à la réalité en générant un nombre de KM parcourus proportionnel au nombre de jours d'emprunts
5. Le Générateur aléatoire d'Excel : Pour des valeurs moins critique (pourcentage de réparation, commission agence …) on à générer des nombres aléatoires avec des intervalles précis
6. Afin de vérifier l'unicité de nos clés primaire on a fait un script python qui teste cette spécification.

PHOTO SCRIPT PYTHON

```python
import csv
import random
month=['JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP','OCT','NOV','DEC']
# csv file name
filename = "dates.csv"

# initializing the titles and rows list
fields = []
rows = []
result=[]
# reading csv file
with open(filename, 'r') as csvfile:
    # creating a csv reader object
    csvreader = csv.reader(csvfile)
    # extracting field names through first row
    fields = csvreader.next()
    # extracting each data row one by one
    for row in csvreader:
        rows.append(row)
    # get total number of rows
    print("Total no. of rows: %d"%(csvreader.line_num))
i=0
while i<=100:
    ok=1
    while(ok==1):
        a=rows[random.randint(0,csvreader.line_num-10)]#begin date
        b=rows[random.randint(0,csvreader.line_num-10)]#end date
        montha=month.index(a[1]) # extract month to be compared
        monthb=month.index(b[1])
        #print "month of a = ",montha,"month of b",monthb,"\n"
        if a[2]<=b[2] and a[0]<=b[0] and montha<=monthb :
            ok=0
            facture=str(abs(random.randint(1,int(a[0]))))+'.'+month[montha]+'.'+str(a[2])
            dayreturn=(int(b[0])+abs(random.randint(0,2)))
            if dayreturn>31:
                dayreturn=31 #no month exceeds 31
            if dayreturn>28 and month[monthb]=='FEB':
                dayreturn=28 # for feb month
            reel=str(dayreturn)+'.'+month[monthb]+'.'+str(b[2])
            #calculate number of days
            if a[2]==b[2]:
                days=abs(int(a[0])-dayreturn+(monthb-montha)*31)
            else:
                days=abs(int(a[0])-dayreturn+(monthb+montha)*31)
            kilometreparcouru=random.randint(20,80)
            for x in range(1,days):
                kilometreparcouru+=random.randint(20,80)
            result.append([facture,a[3],b[3],reel,days,kilometreparcouru])
```

Figure 3 Exemple de script python utilisé pour le jeu de données

b. Échantillon du Jeu de données

Ci-dessous un Échantillon du JEU DE DONNÉES générées pour chaque table. On a mis en gras la clé primaire pour la lisibilité du texte et on a mis l'attribut de chaque table en entête pour faciliter l'interprétation de ce jeu de données.

Notons bien que la table facture est généré automatiquement (avec un trigger) .On a pas donc besoin de générer un jeu de données pour cette table.

Table **agence** :

Tableau 2 échantillon de jeu de données table agence

codeAgence	adresseAgence	villeAgence	cpAgence	comissionAgence
7512022	3 rue Harpignies	Paris	75020	6
9202505	16 Avenue Henri Barbusse	Colombes	92700	10
9203613	125 Avenue Louis Roche	Gennevilliers	92230	9
9205030	123 Rue Salvador Allende	Nanterre	92000	8
7511420	59 avenue Reille	Paris	75014	9
75115PD	81 rue Falguiere	Paris	75015	8
7511308	245 rue de Tolbiac	Paris	75013	8
9307012	44 avenue du Capitaine Glarner	SaintOuen	93400	10
7510620	8 rue Danton	Paris	75006	10
9202406	122 boulevard Jean Jaurès	Clichylagarenne	92210	10

Table **contrat** :

Tableau 3 échantillon de jeu de données table Contrat

noContrat	dateDepartContrat	dateRetourPrevue	kmDepartContrat	dateContrat	limiteKM	immatriculationVehicule	noPermis	codeAgence	retournercodeAgence	prixlitre	idtypeContrat
201800001	12.JUL.2018	19.JUL.2019	44390	11.JUL.2018	62500	AA-236-UGC	67454398	7512022	7512022	0	6001
201800002	21.MAR.2018	26.APR.2018	39654	14.MAR.2018	0	AB-530-SUM	74323363	7510501	9202505	0	6002
201800003	16.FEB.2019	18.DEC.2019	12834	5.FEB.2019	29000	AD-699-URU	48003869	9202406	9203613	0	6001
201800004	11.APR.2018	24.MAY.2018	28675	2.APR.2018	0	AE-607-WID	56101060	9300113	9205030	0	6002
201800005	11.APR.2018	28.MAY.2019	35927	3.APR.2018	46000	AH-669-NWY	60770453	9204803	7511420	0	6001
201800006	7.MAY.2019	8.MAY.2019	72569	3.MAY.2019	74700	AJ-253-ASE	62576410	75115PD	75115PD	0	6001
201800007	13.JAN.2019	13.FEB.2019	26796	9.JAN.2019	0	AK-302-DXM	29901240	9200413	7511308	0	6002
201800008	13.JAN.2018	22.MAY.2019	15063	7.JAN.2018	21000	AK-435-DYX	81344269	9207308	9307012	0	6001
201800009	25.MAY.2018	8.DEC.2018	19977	20.MAY.2018	0	AN-501-MTI	54827006	9401804	7510620	0	6002
201800010	10.JUN.2018	10.JUL.2019	28403	6.JUN.2018	45000	AN-995-SQH	97938617	9203506	9202406	0	6001

Table **véhicule** :

Tableau 4 Echantillon de jeu de données table véhicule

immatriculationVehicule	dateAchatVehicule	kilometrageVehicule	modeleVehicule	puissanceVehicul	categorieVehicule	marqueVehicule	prixJournalierVehicu	prixkmVehicule	depotGarantieVehicule	typeCarburantVehicule	codeAgence
AA-236-UGC	2.JUL.2017	62781	C-Class	340	Berline	Mercedes-Benz	0	0	0	diesel	7512022
AB-530-SUM	21.MAR.2017	40844	Camaro	387	Berline	Chevrolet	1	1	1	diesel	9204803
AD-699-URU	1.FEB.2018	28005	GL-Class	63	Berline	Mercedes-Benz	2	2	2	essence	7511903
AE-607-WID	1.APR.2017	29404	Mazda5	173	Coupe	Mazda	3	3	3	essence	75115PD
AH-669-NWY	11.APR.2017	45277	Tundra	319	Monospace	Toyota	4	4	4	essence	9204807
AJ-253-ASE	7.MAY.2018	72634	F450	341	Coupe	Ford	5	5	5	essence	9202302
AK-302-DXM	3.JAN.2018	28324	Cooper Countryma	293	Monospace	MINI	6	6	6	diesel	9401602
AK-435-DYX	3.JAN.2017	20827	G-Class	160	Monospace	Mercedes-Benz	7	7	7	diesel	9401602
AN-501-MTI	25.MAY.2017	31397	Frontier	64	Berline	Nissan	8	8	8	diesel	9164502
AN-995-SQH	1.JUN.2017	45056	Lancer Evolution	75	Coupe	Mitsubishi	9	9	9	essence	9202505
AP-489-QOS	18.SEP.2017	46490	Pilot	325	Berline	Honda	10	10	10	essence	7510501
AQ-466-CDG	11.APR.2017	65064	LS	81	Berline	Lexus	11	11	11	essence	9209205

Table **client** :

Tableau 5 Echantillon de jeu de données client

noPermis	nomClient	prenomClient	adresseClient	villeClient	cpClient	paysClient	telephoneClient	mailClient
67454398	Beahan	Ashleigh	1747 Adell Avenue	East Deionburgh	72326	Syria	3430387625	Vada@jay.us
74323363	Connelly	Weston	80069 Cydney Haven	East Florence	98701	Ukraine	4879109630	Jazlyn_Predovic@madeline.biz
48003869	Macejkovic	Brandy	4718 Mack Stravenue	Port Joesph	47193	New Caledonia	9330627026	Friedrich@isaiah.co.uk
56101060	Johnston	Nayeli	604 Zulauf Tunnel	South Lavinachester	09856-8175	Ireland	7316111024	Angelina_Schaefer@neha.com
60770453	Ryan	Libby	928 Berge Mission	North Ardithberg	10518-7297	Cayman Islands	7054766185	Gisselle@carlos.tv
62576410	Schumm	Eino	24900 Schmeler Shoals	Lake Kasandraview	55308	Uganda	8345017528	Nettie.Ullrich@joelle.co.uk
29901240	Bartoletti	Maximilian	42758 Wisozk Wells	West Danteland	43368	Nauru	1400563813	Kylie_Okuneva@foren.io
81344269	Bednar	Donato	911 Garrick Mountains	Bergnaumville	61108-9590	Bhutan	2412711285	Randall@kiera.biz
54827006	Jenkins	Kole	1228 Pollich Parks	South Reva	75934-5501	Bolivia	8922291051	Sterling_Goodwin@ruthie.biz
97938617	Hegmann	Misael	75709 Gibson Field	Fisherville	21371	Vanuatu	8603920330	Fleta_Daugherty@helmer.tv

Table **typecontrat :**

Tableau 6 Echantillon de jeu de données type contrat

idtypeContrat	typeContrat
6001	kilometre
6002	jour

table **ficheretour** :

Tableau 7 Echantillon de jeu de données table fiche retour

noFicheRetour	dateRetourReel	kmRetourReel	pourcentageReparation	nbLitresManquants	codeAgence	noPermis	nocontrat
5000	21.JUL.2019	62781	16	19	7512022	67454398	201800001
5001	27.APR.2018	40844	18	1	9204803	74323363	201800002
5002	19.DEC.2019	28005	8	0	7511903	48003869	201800003
5003	25.MAY.2018	29404	16	6	75115PD	56101060	201800004
5004	30.MAY.2019	45277	0	5	9204607	60770453	201800005
5005	8.MAY.2019	72634	16	18	9202302	62576410	201800006
5006	14.FEB.2019	28324	20	7	9401602	29901240	201800007
5007	23.MAY.2019	20927	8	16	9401602	81344269	201800008
5008	10.DEC.2018	31397	9	15	9164502	54827006	201800009
5009	12.JUL.2019	45056	7	15	9202505	97938617	201800010
5010	21.SEP.2019	46490	12	16	7510501	37497384	201800011
5011	7.JUL.2019	65064	3	5	9205005	57448877	201800012

c. Insertion du jeu de données dans la base:

L'insertion de ce jeu de données peut se faire facilement avec plusieurs méthode cependant il faut faire attention à l'ordre d'insertion et prendre en considération les dépendances entre les tables.

On cite 3 méthodes d'insertion :

1. **Insertion classique** tuple par tuple en générant les requêtes d'insertion avec un script:

```
INSERT INTO client VALUES
            (
                        67454398,
                        'beahan',
                        'ashleigh',
                        '1747 adell avenue',
                        'east deionburgh',
                        72326,
                        'syria',
                        3430387625,
                        'vada@jay.us'
            );
INSERT INTO agence VALUES
            (
                        7512022,
                        '3 rue harpignies',
                        'paris',
                        75020,6
            );
INSERT INTO typecontrat VALUES
            (
                        6001,
                        'kilometre'
            );
INSERT INTO vehicule VALUES
            (
                        'aa-236-ugc',
                        '2.jul.2017',
```

```
                        62781,
                        'c-class',
                        340,
                        'berline',
                        'mercedes-benz',
                        0,0,0,
                        'diesel',
                        7512022
            );
INSERT INTO contrat VALUES
            (
                        201800001,
                        '12.jul.2018',
                        '19.jul.2019',
                        44390,
                        '11.jul.2018',
                        62500,
                        'aa-236-ugc',
                        67454398,7512022,7512022,0,6001
            );
INSERT INTO ficheretour VALUES
            (
                        5000,
                        '21.jul.2019',
                        62781,16,19,7512022,67454398,201800001
            );
```

2. Insertion avec un script SQL/PLSQL :

```
DECLARE
```

```
    nc                    contrat.nocontrat%TYPE;
    dateretourreel        ficheretour.dateretourreel%TYPE;
    kmretourreel          ficheretour.kmretourreel%TYPE;
    pourcentagereparation ficheretour.pourcentagereparation%TYPE;
    nblitresmanquants     ficheretour.nblitresmanquants%TYPE;
    codeagence            agence.codeagence%TYPE;
    permis                client.nopermis%TYPE;
    nofiche               ficheretour.noficheretour%TYPE;
BEGIN
    nc := '&nc';

    dateretourreel := To_date('&dateRetourReel');

    kmretourreel := '&kmRetourReel';

    pourcentagereparation := '&pourcentageReparation';

    nblitresmanquants := '&nbLitresManquants';

    codeagence := '&codeAgence';

    SELECT DISTINCT nopermis
    INTO   permis
    FROM   contrat
    WHERE  contrat.nocontrat = nc;

    SELECT seq_ficheretour.NEXTVAL
    INTO   nofiche
```

```
    FROM   dual;


    INSERT INTO ficheretour
    VALUES      (nofiche,
                 To_date(dateretourreel, 'dd.mm.yy'),
                 kmretourreel,
                 pourcentagereparation,
                 nblitresmanquants,
                 codeagence,
                 permis,
                 nc);
END;
/
```

3. **Insertion avec SQL LOADER** en utilisant la commande sqlldr :

La méthode la plus rapide et la plus pratique dans notre cas est l'insertion de tuples à partir de fichiers csv . on a un fichier load.ctl contenant :

```
LOAD data infile '/home/oracle/Desktop/projet/client.csv' append INTO TABLE client
fields terminated BY ',' (
nopermis,nomclient,prenomclient,adresseclient,villeclient,cpclient,paysclient,telep
honeclient,mailclient )


LOAD data infile '/home/oracle/Desktop/projet/agencefinal.csv' append INTO TABLE
agence fields terminated BY ',' (
codeagence,adresseagence,villeagence,cpagence,comissionagence )


LOAD data infile '/home/oracle/Desktop/projet/TYPECONTRAT.csv' append INTO TABLE
typecontrat fields terminated BY ',' ( idtypecontrat,typecontrat )


LOAD data infile '/home/oracle/Desktop/projet/voiturefinal.csv' append INTO TABLE
vehicule fields terminated BY ',' (
immatriculationvehicule,dateachatvehicule,kilometragevehicule,modelevehicule,puissa
```

```
ncevehicule,categorievehicule,marquevehicule,prixjournaliervehicule,prixkmvehicule,
depotgarantievehicule,typecarburantvehicule,codeagence )


LOAD data infile '/home/oracle/Desktop/projet/contrat.csv' append INTO TABLE
contrat fields terminated BY ',' (
nocontrat,datedepartcontrat,dateretourprevue,kmdepartcontrat,datecontrat,limitekm,i
mmatriculationvehicule,nopermis,codeagence,retournercodeagence,prixlitre,idtypecont
rat )


LOAD data infile '/home/oracle/Desktop/projet/ficheRetour.csv' append INTO TABLE
ficheretour fields terminated BY ',' (
noficheretour,dateretourreel,kmretourreel,pourcentagereparation,nblitresmanquants,c
odeagence,nopermis,nocontrat )
```

la cimmande utilize est : sqlldr userid=nom de l'utilisateur control=load.ctl

d. Manipulation du jeu de données et formulation des requêtes en SQL:

Dans ce paragraphe on va essayer de balayer sur les différentes fonctions de la base . Ces requêtes sont données a titre indicatif et ne peuvent pas résumer toutes les fonctionnalités offert par notre base de données .

```
PROCEDURE de rendement de vehicule
--les marques de vehicules préférées des clients
--
 premier select pour afficher la marque de vehicule qui a le max de contrat
 .
--
le deuxieme et troixieme select pour extraire le max du nb de contrat selon
 la marque

SELECT    Count(nopermis) AS nb,
          marquevehicule
FROM      contrat C ,
          vehicule v
WHERE     c.immatriculationvehicule=v.immatriculationvehicule
GROUP BY( marquevehicule)
HAVING    Count(nopermis) IN
          (
                SELECT Max(nb)
                FROM   (
                              SELECT    Count(nopermis) AS nb,
```

```
                                        marquevehicule
                                FROM    contrat C ,
                                        vehicule v
                                WHERE   c.immatriculationvehicule=v.immatr
iculationvehicule
                                GROUP BY( marquevehicule)));

--
les noms des clients qui ont reservés(ou retourner) dans le département 78

SELECT nomclient,
       prenomclient
FROM   client c ,
       contrat co
WHERE  co.nopermis=c.nopermis
AND    (
              retournercodeagence IN
              (
                     SELECT codeagence
                     FROM   agence
                     WHERE  cpagence LIKE '78%' )
       OR     codeagence IN
              (
                     SELECT codeagence
                     FROM   agence
                     WHERE  cpagence LIKE '78%' ) );

--
les vehicules de type coupé qui n'ont pas été loué À PARTIR DE 1 JANVIER 20
19

SELECT immatriculationvehicule
FROM   vehicule
WHERE  categorievehicule='Coupe'
AND    immatriculationvehicule NOT IN
       (
              SELECT immatriculationvehicule
              FROM   contrat
              WHERE  datedepartcontrat >To_date('01-JAN-2019', 'DD-MON-
RR'));

--
 le pourcentage  du cout des pleins de l'essence par rapport au total des c
ontrats

SELECT Sum(c.prixlitre*f.nblitresmanquants)/Sum(fa.totalfacture)*100
FROM   contrat c,
       ficheretour f,
```

```
       facture fa
WHERE  c.nocontrat=f.nocontrat
AND    fa.nocontrat=c.nocontrat;

--les département qui n'a eu ni un retour ni un contrat

SELECT Substr(cpagence,0,2)
FROM   agence
WHERE  codeagence NOT IN
       (
              SELECT codeagence
              FROM   contrat
              UNION
              SELECT codeagence
              FROM   ficheretour );

--
les immatriculation de tout les vehicules qui ont ete loués sur tout les dé
partement
-- Codes des deparmetements ou il existe des agences X1

SELECT Substr(cpagence,0,2)SELECT   Substr(cpagence,0,2) AS departement
FROM     agence
GROUP BY Substr(cpagence,0,2);

--immatricualation des vehicules accompagnes par les codes departement X2

SELECT immatriculationvehicule,
       Substr(cpagence,0,2) AS departement
FROM   contrat c ,
       agence a
WHERE  a.codeagence = c.codeagence;

--
 X2 div X1 : les immatriculation de  tout les vehicules qui ont ete loués s
ur tout les département

SELECT DISTINCT immatriculationvehicule
FROM           (
                      SELECT immatriculationvehicule,
                             Substr(cpagence,0,2) AS departement
                      FROM   contrat c ,
                             agence a
                      WHERE  a.codeagence = c.codeagence) rq1
WHERE          NOT EXISTS
               (
                      SELECT *
                      FROM   (
```

```
                                        SELECT   Substr(cpagence,0,2) AS dep
artement
                                        FROM     agence
                                        GROUP BY Substr(cpagence,0,2)) rq2
                        WHERE  NOT EXISTS
                               (
                                      SELECT *
                                      FROM   (
                                                    SELECT immatriculationve
hicule,
                                                           Substr(cpagence,0
,2) AS departement
                                                    FROM   contrat c ,
                                                           agence a
                                                    WHERE  a.codeagence = c.
codeagence) rq3
                                      WHERE  rq3.departement = rq2.departeme
nt
                                      AND    rq3.immatriculationvehicule = r
q1.immatriculationvehicule ) );

--le type de vehicule le plus rentable en été de 2018
-- la sous-
table somme contient le total des facture selon l'immatriculation dans la p
eriode de l'été

SELECT categorievehicule
FROM   vehicule
WHERE  immatriculationvehicule IN ( WITH somme AS
                                    (
                                           SELECT   immatriculationvehicul
e,
                                                    Sum(totalfacture) sm
                                           FROM     contrat c
                                           JOIN     facture f
                                           ON       f.nocontrat=c.nocontra
t
                                           WHERE    datedepartcontrat BETW
EEN To_date('01-JUN-2018', 'DD-MON-RR') AND      To_date('31-AUG-
2018', 'DD-MON-RR')
                                           GROUP BY (immatriculationvehicu
le))SELECT immatriculationvehicule
                           FROM   somme
                           WHERE  sm=
                                  (
                                         SELECT Max(sm)
                                         FROM   somme));
```

```
--
les clients qui ont pris des vehicules dans une agence et l'on rendu dans u
ne autre agence

SELECT nomclient,
       prenomclient
FROM   contrat c,
       client cl,
       ficheretour f
WHERE  c.nopermis=cl.nopermis
AND    c.nocontrat=f.nocontrat
AND    c.codeagence <> f.codeagence;

--
les clients qui ont reservé une vehicule  de type monospace et de puissance
 supérieure à 100

SELECT DISTINCT nomclient,
                prenomclient
FROM            contrat c,
                client cl,
                vehicule vl
WHERE           c.nopermis=cl.nopermis
AND             c.immatriculationvehicule=vl.immatriculationvehicule
AND             vl.categorievehicule='Monospace'
AND             vl.puissancevehicule>100;

--le prix moyen de contrat pour les véhicules achetés en 2018

SELECT   vehicule.immatriculationvehicule,
         Avg(totalfacture)
FROM     contrat,
         vehicule,
         facture
WHERE    contrat.immatriculationvehicule=vehicule.immatriculationvehicule
AND      facture.nocontrat=contrat.nocontrat
AND      vehicule.dateachatvehicule >= '01JAN2018'
AND      vehicule.dateachatvehicule < '01JAN2019'
GROUP BY vehicule.immatriculationvehicule;

--le pays des clients qui ont une facture de plus de 1000 euros en 2018

SELECT DISTINCT paysclient
FROM            client,
                contrat ,
                facture
WHERE           client.nopermis=contrat.nopermis
```

```
AND             facture.nocontrat=contrat.nocontrat
AND             contrat.datecontrat >= '01JAN2018'
AND             contrat.datecontrat < '01JAN2019'
AND             facture.totalfacture>1000
-- l'agence qui a reservé le plus de contrats entre 2017 et 2018

SELECT   numero,
         Max(nombredecontrat)
FROM     (
                  SELECT   contrat.codeagence AS numero,
                           Count(*)           AS nombreDeContrat
                  FROM     contrat
                  WHERE    contrat.datecontrat >= '01JAN2017'
                  AND      contrat.datecontrat < '01JAN2019'
                  GROUP BY contrat.codeagence )
GROUP BY numero;

--le code de l'agence qui a fait que des retours (sans faire aucun contrat)

SELECT codeagence
FROM   agence
WHERE  codeagence IN
       (
              SELECT codeagence
              FROM   ficheretour)
AND    codeagence NOT IN
       (
              SELECT codeagence
              FROM   contrat);

--le nom des clients qui ont plus de 5 jours de retard

SELECT nomclient,
       prenomclient
FROM   client c ,
       contrat co
WHERE  c.nopermis=co.nopermis
AND    sysdate-dateretourprevue > 5 ;

--
 le nombre de contrat qui ont un depot de garantie de plus de 10% du prix t
otal de la facture

SELECT Count(*)
FROM   facture ,
       contrat ,
       vehicule
WHERE  facture.nocontrat=contrat.nocontrat
```

```
AND    contrat.immatriculationvehicule=vehicule.immatriculationvehicule
AND    facture.totalfacture*0.1 < vehicule.depotgarantievehicule;

--
 le nom du client le plus fidéle(le client qui ont reservés plus que tout l
es clients )SELECT nomclient
FROM   (
                SELECT   Count(nopermis) AS nb,
                         nopermis
                FROM     contrat
                GROUP BY nopermis) t,
       client c
WHERE  c.nopermis=t.nopermis
AND    t.nb =
       (
              SELECT Max(nb)
              FROM   (
                              SELECT   Count(nopermis) AS nb,
                                       nopermis
                              FROM     contrat
                              GROUP BY nopermis)) ;

-- le type préféré d'un client donnée

SELECT categorievehicule
FROM   (
                SELECT   Count(categorievehicule ) AS nb,
                         categorievehicule
                FROM     vehicule v,
                         contrat c
                WHERE    c.immatriculationvehicule = v.immatriculationvehic
ule
                AND      c.nopermis = 67454398
                GROUP BY categorievehicule )
WHERE  nb =
       (
              SELECT Max(nb)
              FROM   (
                              SELECT   Count(categorievehicule ) AS nb,
                                       categorievehicule
                              FROM     vehicule v,
                                       contrat c
                              WHERE    c.immatriculationvehicule = v.immatr
iculationvehicule
                              AND      c.nopermis = 67454398
                              GROUP BY categorievehicule) );
```

```
-- le mois le plus rentable en 2018

WITH stat AS
(
         SELECT   Extract(month FROM datecontrat) AS mois,
                  Sum(totalfacture)               AS recettes
         FROM     contrat
         JOIN     facture
         ON       contrat.nocontrat=facture.nocontrat
         WHERE    datecontrat BETWEEN To_date('01-JAN-2018', 'DD-MON-
RR') AND      To_date('31-DEC-2018', 'DD-MON-RR')
         GROUP BY Extract(month FROM datecontrat) )
SELECT To_char(To_date(mois, 'MM'), 'MONTH')AS mois,
       recettes
FROM   stat
WHERE  recettes =
       (
              SELECT Max(recettes)
              FROM   stat);

--la liste des vehicule , l'agence de location et l'argence de retour

SELECT c.immatriculationvehicule ,
       c.codeagence,
       fr.codeagence
FROM   contrat c
JOIN   ficheretour fr
ON     c.nocontrat=fr.nocontrat;

--supprimer les clients qui n'ont jamais reservé
DELETE
FROM   client
WHERE  nopermis NOT IN
                        (
                        SELECT DISTINCT nopermis
                        FROM            contrat);


--
reduire la comission des agences qui n'ont fait aucun contrat de location d
e 10%UPDATE agence
SET    comissionagence = comissionagence * 0.9
WHERE  codeagence NOT                    IN
                                             (
                                             SELECT DISTINCT codeagence
                                             FROM            contrat);

--
```

```
majorer la comission de toute les agences de 10%UPDATE agence comissionAgen
ce = comissionagence * 1.1;

--supprimer les client qui n'ont pas d'adresse mail valideDELETE
FROM   client
WHERE  mailclient NOT LIKE '%@%.%';

--
 rajouter 10% au prix journalier d'un vehicule qui a un prix journalier inf
erieur a 50 euroUPDATE vehicule
SET    prixjournaliervehicule = prixjournaliervehicule*(1.1)
WHERE  prixjournaliervehicule < 50;
```

VI. les vues

a. déclaration des vues :

```
-- la localisation des vehicules qui n'ont jamais été loué
CREATE VIEW localisationvehiculenonloue
AS
  SELECT V.immatriculationvehicule,
         V.marquevehicule,
         V.modelevehicule,
         A.adresseagence,
         A.cpagence
  FROM   vehicule v,
         agence a
  WHERE  immatriculationvehicule IN(SELECT immatriculationvehicule
                                    FROM   vehicule
                                    WHERE  immatriculationvehicule NOT IN
                                           (SELECT
                                           immatriculationvehicule
                                                                    FROM
                                           contrat
                                           )
                                   )
         AND a.codeagence = V.codeagence;

-- la liste des clients qui ont loué au moins une voiture Diesel (peut être
utiliser comme statistiques pour diminuer le nombre de vehicule diesel )
CREATE VIEW listeclientdiesel
AS
```

```
  SELECT DISTINCT nomclient,
                  prenomclient
  FROM   client c,
         contrat co,
         vehicule v
  WHERE  c.nopermis = co.nopermis
         AND co.immatriculationvehicule = v.immatriculationvehicule
         AND v.typecarburantvehicule = 'diesel';


-- Liste des agence qui au moins fait une location qui part et retourne dans la
meme agence
CREATE VIEW listeagence
AS
  SELECT DISTINCT c.codeagence
  FROM   contrat c,
         agence a,
         ficheretour fr
  WHERE  a.codeagence = c.codeagence
         AND fr.nocontrat = c.nocontrat
         AND c.codeagence = fr.codeagence;


--la moyenne des factures selon le type de vehicules
CREATE VIEW moyfacturetype
AS
  SELECT vehicule.categorievehicule,
         Avg(totalfacture) "moyenneFacture"
  FROM   contrat,
         vehicule,
         facture
  WHERE  contrat.immatriculationvehicule = vehicule.immatriculationvehicule
```

```
        AND facture.nocontrat = contrat.nocontrat
  GROUP  BY vehicule.categorievehicule;

--nombre de contrats de chaque client
CREATE VIEW info_client
(nomclient, prenomclient, nombrecontrat )
AS
  SELECT nomclient,
         prenomclient,
         Count(c.nopermis)
  FROM   contrat c,
         client a
  WHERE  c.nopermis = a.nopermis
  GROUP  BY c.nopermis,
            nomclient,
            prenomclient
  ORDER  BY Count(c.nopermis) DESC;

--le nombre de jours de retard pour chaque client
CREATE VIEW retard
(nocontrat, nomclient, prenomclient, nombrejours )
AS
  SELECT c.nocontrat,
         a.nomclient,
         a.prenomclient,
         Trunc(To_date(SYSDATE, 'dd/mm/yyyy')) -
         To_date(c.dateretourprevue, 'dd/mm/yyyy'
         )
  FROM   contrat c
```

```
        left join ficheretour fr
                ON c.nocontrat = fr.nocontrat,
        client a
  WHERE  c.nopermis = a.nopermis
        AND fr.dateretourreel IS NULL;

--résumé de chaque contrat
CREATE VIEW detailcontrat (nocontrat, nofacture, nomclient,
immatriculationvehicule, datedepartcontrat, dateretourprevue, kmdepartcontrat,
limitekm, typecontratretourner, adresseageneretour, cpagenceretour, totalfacture
)
AS
  SELECT c.nocontrat,
        nofacture,
        nomclient,
        c.immatriculationvehicule,
        datedepartcontrat,
        dateretourprevue,
        kmdepartcontrat,
        limitekm,
        tc.typecontrat,
        ag.adresseagence,
        ag.cpagence,
        totalfacture
  FROM   contrat c
        LEFT JOIN ficheretour fr
                ON c.nocontrat = fr.nocontrat,
        client a,
        vehicule v,
```

```
         facture f,
         typecontrat tc,
         agence ag
  WHERE  c.nopermis = a.nopermis
         AND v.immatriculationvehicule = c.immatriculationvehicule
         AND f.nocontrat = c.nocontrat
         AND c.idtypecontrat = tc.idtypecontrat
         AND c.retournercodeagence = ag.codeagence;

--afficher la disponibilité des véhicules à la date du système
CREATE OR replace VIEW vehiculesdisponibles
AS
  SELECT vehicule.immatriculationvehicule
  FROM   vehicule
  WHERE  vehicule.immatriculationvehicule NOT IN (SELECT immatriculationvehicule
                                                  FROM   contrat,
                                                         ficheretour
                                                  WHERE  contrat.nocontrat =
                                                         ficheretour.nocontrat
                                                         AND
                 SYSDATE < contrat.datedepartcontrat
                                                          OR (
                 ficheretour.dateretourreel IS NULL
                 AND SYSDATE > contrat.dateretourprevue )
                                                          OR ( SYSDATE >
  ficheretour.dateretourreel ));

--afficher les détails de chaque facture
CREATE OR replace VIEW detailfacture
```

```
AS
  SELECT f.nofacture,
         ( fr.kmretourreel - co.kmdepartcontrat )     AS kmParcouru,
         co.limitekm,
         f.supplementkmfacture,
         To_date(co.dateretourprevue, 'dd/mm/yyyy') -
         To_date(co.datedepartcontrat, 'dd/mm/yyyy') AS nbJoursprevu,
         v.prixjournaliervehicule                    AS prixJournalier,
         To_date(fr.dateretourreel, 'dd/mm/yyyy') -
         To_date(co.dateretourprevue, 'dd/mm/yyyy')  AS nbJoursretard,
         f.supplementretardfacture,
         fr.pourcentagereparation,
         v.depotgarantievehicule,
         f.soustractiongarantiefacture               asgarantieRetenu,
         f.supplementagencefacture
  FROM   facture f,
         contrat co,
         ficheretour fr,
         vehicule V
  WHERE  f.nocontrat = co.nocontrat
         AND fr.nocontrat = co.nocontrat
         AND co.immatriculationvehicule = v.immatriculationvehicule;
```

b. droit d'accès au vues

Tableau 8 droits d'accès aux vues

groupe d'accés / nom de la vue	client	collaborateur	finance	Responsable /manage
localisationVehiculeNonLoue		X		X
listeClientDiesel			X	X
listeAgence				X
moyfacturetype			X	X
info_client	X	X	X	X
retard		X		X
detailcontrat	X	X	X	X
VEHICULESDISPONIBLES		X		X
DETAILFACTURE		X	X	X

Notons bien que si on a donné le droit au groupe d'utilisateurs 'client ' un droit d'accés à la vue detailContrat cela ne veut pas dire qu'un client a accès à tous les autres clients . C'est la mission du Front end ou plutôt la couche supérieur qui gère ces restrictions .

les détails de la facture d'un client ne seront disponible qu'à sa demande .c'est pour cela que le groupe utilisateurs n'a pas un accés directe à la vue .

VII. les méta-données

```
-- Creation d'une table qui contiendra la liste des contraintes
CREATE TABLE contraintes
  (
     constraint_name  VARCHAR2(30),
     constraint_type  VARCHAR2(30),
     table_name       VARCHAR2(30),
     search_condition CLOB
  );

-- liste_ora_constraints
-- Script de recuperation des contraintes
DECLARE
BEGIN
    -- Vider les anciennes valeurs
    DELETE FROM contraintes;

    -- Recuperer les contraintes suivant l'utilisateur tries par les nom des
tableaux et types de contraintes
    INSERT INTO contraintes
                (constraint_name,
                 constraint_type,
                 table_name,
                 search_condition)
    SELECT constraint_name,
           Decode (constraint_type, 'P', 'cle primaire',
```

```
                                        'R', 'cle etrangere',
                                        'C', 'check',
                                        'U', 'unique',
                                        'V', 'check view',
                                        'O', 'lecture seule') TYPE,
            table_name,
            To_lob(search_condition)
    FROM    user_constraints
    WHERE   table_name IN (SELECT table_name
                           FROM   user_tables)
    ORDER   BY table_name,
               constraint_type DESC;
END;

/
-- Tables contenant les triggers
CREATE TABLE TRIGGERS
  (
     trigger_name VARCHAR2(30),
     table_name   VARCHAR2(30)
  );

-- Script de recuperation des triggers
DECLARE
BEGIN
    -- Vider les anciennes valeurs
    DELETE FROM TRIGGERS;

    -- Selection des triggers
```

```
    INSERT INTO TRIGGERS
                (trigger_name,
                 table_name)
    SELECT trigger_name,
           table_name
    FROM   all_triggers
    WHERE  table_name IN (SELECT table_name
                          FROM   user_tables)
    ORDER  BY table_name;
END;

/
-- Creation d'une table qui contiendra la liste des cle primaires
CREATE TABLE cle_primaire
  (
     constraint_name VARCHAR2(30),
     table_name      VARCHAR2(30),
     column_name     VARCHAR2(4000)
  );

-- liste cle primaire de chaque table et nom contrainte
DECLARE
BEGIN
    DELETE FROM cle_primaire;

    INSERT INTO cle_primaire
                (constraint_name,
                 table_name,
                 column_name)
```

```
    SELECT c.constraint_name,
           c.table_name,
           a.column_name
    FROM   user_constraints c,
           all_cons_columns a
    WHERE  c.constraint_name = a.constraint_name
           AND c.table_name = a.table_name
           AND c.constraint_type = 'P'
           AND c.table_name IN (SELECT table_name
                                FROM   user_tables)
    ORDER  BY c.table_name;
END;

/
-- Creation d'une table qui contiendra la liste des cle etrangeres
CREATE TABLE cle_etrangere
  (
     constraint_name VARCHAR2(30),
     table_name      VARCHAR2(30),
     column_name     VARCHAR2(4000),
     r_table_name    VARCHAR2(30)
  );

-- liste cle etrangeres les tables et colonnes qui les referents
DECLARE
BEGIN
    DELETE FROM cle_etrangere;

    INSERT INTO cle_etrangere
```

```
                    (constraint_name,
                     table_name,
                     column_name,
                     r_table_name)
        SELECT c.constraint_name,
               c.table_name,
               a.column_name,
               (SELECT r.table_name
                FROM   user_constraints r
                WHERE  r.constraint_name = c.r_constraint_name) r_table_name
        FROM   user_constraints c,
               all_cons_columns a
        WHERE  c.constraint_name = a.constraint_name
               AND c.table_name = a.table_name
               AND constraint_type = 'R'
               AND c.table_name IN (SELECT table_name
                                    FROM   user_tables)
        ORDER  BY c.table_name;
    END;

    /
```

VIII. EXPLAIN PLAN

a. Requete 1 :

```
FOR SELECT DISTINCT immatriculationvehicule FROM (SELECT immatriculationveh
icule
, substr(cpagence, 0, 2) AS departement FROM contrat c, agence a WHERE
a.codeagence = c.codeagence) rq1 WHERE NOT EXISTS ( SELECT * FROM (SELECT s
ubstr(cpagence, 0
, 2) AS departement FROM agence GROUP BY substr(cpagence, 0, 2)) rq2 WHERE
NOT
EXISTS ( SELECT * FROM (SELECT immatriculationvehicule, substr(cpagence, 0,
 2)
AS departement FROM contrat c, agence a WHERE a.codeagence = c.codeagence)
rq3
WHERE rq3.departement = rq2.departement AND rq3.immatriculationvehicule =
rq1.immatriculationvehicule ) );

SELECT *
FROM   TABLE(dbms_xplan.display);
```

```
-------------------------------------------------------------------------------------------
| Id  | Operation              | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |              |    20 |   640 |    20   (5)| 00:00:01 |
|   1 |  HASH UNIQUE           |              |    20 |   640 |    20   (5)| 00:00:01 |
|*  2 |   FILTER               |              |       |       |            |          |
|   3 |    TABLE ACCESS FULL   | CONTRAT      |    33 |  1056 |     3   (0)| 00:00:01 |
|*  4 |   FILTER               |              |       |       |            |          |
|   5 |    TABLE ACCESS FULL   | AGENCE       |    34 |  1088 |     3   (0)| 00:00:01 |
|   6 |    NESTED LOOPS        |              |       |       |            |          |
|   7 |     NESTED LOOPS       |              |     1 |   128 |     3   (0)| 00:00:01 |
|*  8 |      TABLE ACCESS FULL | CONTRAT      |     1 |    64 |     3   (0)| 00:00:01 |
```

|* 9 | INDEX UNIQUE SCAN | CLEPRIMAITEAGENCE | 1 | | 0 (0)| 00:00:01 |

|* 10 | TABLE ACCESS BY INDEX ROWID| AGENCE | 1 | 64 | 0 (0)| 00:00:01 |

Predicate Information (identified by operation id):

---

2 - filter(NOT EXISTS (SELECT 0 FROM "AGENCE" "SYS_ALIAS_1" WHERE NOT EXISTS (SELECT 0
FROM "AGENCE" "A","CONTRAT" "C" WHERE "IMMATRICULATIONVEHICULE"=:B1 AND
"A"."CODEAGENCE"="C"."CODEAGENCE" AND SUBSTR("CPAGENCE",0,2)=SUBSTR(:B2,0,2))))

4 - filter(NOT EXISTS (SELECT 0 FROM "AGENCE" "A","CONTRAT" "C" WHERE
"IMMATRICULATIONVEHICULE"=:B1 AND "A"."CODEAGENCE"="C"."CODEAGENCE" AND
SUBSTR("CPAGENCE",0,2)=SUBSTR(:B2,0,2)))

8 - filter("IMMATRICULATIONVEHICULE"=:B1)

9 - access("A"."CODEAGENCE"="C"."CODEAGENCE")

10 - filter(SUBSTR("CPAGENCE",0,2)=SUBSTR(:B1,0,2))

Note

- dynamic sampling used for this statement (level=2)

b. Analyse requête :

Nous pouvons voir d'après le tableau généré par explain plan, l'ensemble de tables qu' SQL a consulté lors de l'exécution

De la requête qui sont (CONTRAT, AGENCE, AGENCE).

La colonne opération nous indique la nature de ces accès ;

Nous pouvons constater d'après le tableau explain plan que les deux tables CONTRAT et AGENCE ont eu un TABLE ACCESS FULL,

C'est l'accès le plus couteux puisque SQL va devoir consulter toutes les lignes et colonnes (cela prendra du temps si on

Parle de tables contenant un grand nombre de lignes/colonnes (au-delà de 1 000 000)). Par accès le plus couteux,

Nous parlons dans le contexte de temps, mémoire et cpu, le ralentissement du temps d'exécution n'est pas visible

Dans ce tableau vu que la table contient une trentaine de tuples, nous pouvons voir le cout mémoire du chargement

De table qui est supérieur aux méthodes d'accès, le cout de cpu est élevé aussi par rapport aux autres opérations,

Ce cout reste enfin relatif à la puissance de la machine virtuelle.

Vu la nature de la requête on juge que l'optimiseur SQL n'a pas pu choisir une autre méthode d'accès pour accomplir

La requête.

Nous pouvons voir qu'il y'avait un accès TABLE ACCESS BY INDEX ROWID sur la table agence, en effet dans ce

Cas nous voyons l'optimiseur SQL en plein action, c'est un accès indexé par ROWID, ce ROWID est récupère

D'une recherche dans l'index, SQL accède alors directement à la ligne désiré en utilisant ce ROWID

Ce qui évite le parcours de la table et qui permet de gagner temps exécution et espace mémoire.

L'opération HASH UNIQUE (cette opération est du à l'évocation du mot distinct dans la requête),

Regroupe les clés désirées à être uniques dans une table de hachage dans un ordre quelconque (absence d'order by),

Cette opération permet de gagner du temps exécution (le temps d'accès à une table de hachage est inferieur à celui d'une table ordinaire

O (nlogn) < O(n) ou n est la taille du tableau), en contrepartie cette opération est peu couteuse en mémoire.

L'opération INDEX UNIQUE SCAN est exécuté vu la présence de la contrainte CLEPRIMAITEAGENCE qui décrit

La clé primaire de l'agence, c'est un scan unique vu que la clé primaire est unique d'où sa rapidité d'exécution

L'optimiseur SQL a exploité cette contrainte alors pour réduire les charges de parcours

Le schéma d'exécution commence par l'opération 10 vers l'opération 0 suivant l'arbre d'exécution d'explain plan.

Nous voyons le mot FILTER dans les opérations et les informations complémentaires de la table aussi,

En effet ces filtres sont exécutés sur chaque ligne de la table consultée, celui-ci permettra d'appliquer

Les conditions de sélection présentes dans la requête, il arrive de remplacer certain champs dans les filtres

Par des alias, si cet alias est présent dans plusieurs filtre, en conclusion le moins il existe de filtre

Le plus rapide une requête s'exécutera, le plus une méthode d'accès au tables est optimisé le moins de tuples ou seront

Les filtres appliqués et le moins le cout d'exécution coutera.

L'opération NESTED LOOPS est appelé du a la présence d'une jointure dans la requête en réalité

SQL récupère une table et rajoute les colonnes à chaque ligne de cette table en parcourant

La deuxième table entière et appliquant la condition de jointure sur chaque tuples. Nous parlons ici d'une

Boucle imbriqué de parcours le temps d'exécution est alors le produit du temps du parcours du premier tableau

Et le temps du parcours de la deuxième table.

c. Requête 2

```
FOR WITH stat AS ( SELECT EXTRACT(MONTH FROM datecontrat) AS mois, sum(
totalfacture) AS recettes FROM contrat JOIN facture ON
contrat.nocontrat=facture.nocontrat WHERE datecontrat BETWEEN to_date('01-
JAN-2018', 'DD-MON-RR') AND to_date
('31-DEC-2018', 'DD-MON-
RR') GROUP BY EXTRACT(MONTH FROM datecontrat) ) SELECT
to_char(to_date(mois, 'MM'), 'MONTH')AS mois, recettes FROM stat WHERE rece
ttes
= (SELECT max(recettes) FROM stat);
```

```
SELECT *
FROM   TABLE(dbms_xplan.display);
```

```
-----------------------------------------------------------------------------------------------------
| Id  | Operation                | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                     |    24 |   624 |     8  (13)| 00:00:01 |
|   1 |  TEMP TABLE TRANSFORMATION     |                     |       |       |            |          |
|   2 |   LOAD AS SELECT         | SYS_TEMP_0FD9D6605_A08741 |       |       |            |          |
|   3 |    HASH GROUP BY         |                     |    24 |  1584 |     4  (25)| 00:00:01 |
|   4 |     NESTED LOOPS         |                     |       |       |            |          |
|   5 |      NESTED LOOPS        |                     |    24 |  1584 |     3   (0)| 00:00:01 |
|   6 |       TABLE ACCESS FULL      | FACTURE             |    33 |   858 |     3   (0)| 00:00:01 |
|*  7 |       INDEX UNIQUE SCAN      | CLEPRIMAIRECONTRAT      |     1 |       |     0   (0)| 00:00:01 |
|*  8 |      TABLE ACCESS BY INDEX ROWID| CONTRAT             |     1 |    40 |     0   (0)| 00:00:01 |
|*  9 |    VIEW                  |                     |    24 |   624 |     2   (0)| 00:00:01 |
|  10 |     TABLE ACCESS FULL        | SYS_TEMP_0FD9D6605_A08741 |    24 |   624 |     2   (0)| 00:00:01 |
|  11 |     SORT AGGREGATE       |                     |     1 |    13 |            |          |
|  12 |      VIEW                |                     |    24 |   312 |     2   (0)| 00:00:01 |
|  13 |       TABLE ACCESS FULL      | SYS_TEMP_0FD9D6605_A08741 |    24 |   624 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------
```

Predicate Information (identified by operation id):

```
---------------------------------------------------

   7 - access("CONTRAT"."NOCONTRAT"="FACTURE"."NOCONTRAT")
   8 - filter("CONTRAT"."DATECONTRAT">=TO_DATE(' 2018-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND
```

"CONTRAT"."DATECONTRAT"<=TO_DATE(' 2018-12-31 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND

"DATEDEPARTCONTRAT">=TO_DATE(' 2018-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND

"DATERETOURPREVUE">=TO_DATE(' 2018-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))

9 - filter("RECETTES"= (SELECT MAX("RECETTES") FROM (SELECT /*+ CACHE_TEMP_TABLE ("T1") */ "C0"

"MOIS","C1" "RECETTES" FROM "SYS"."SYS_TEMP_0FD9D6605_A08741" "T1") "STAT"))

Note

- dynamic sampling used for this statement (level=2)

d. Analyse requête 2:

Dans une autre requête nous pouvons voir d'autres opérations,

Whith stats as cette ligne présente dans la requête permet de transformer la sous requête qui la suit en

Une table temporaire libellé stats qu'on exploitera après dans la requête.

TEMP TABLE TRANSFORMATION décrit la transformation de la sous requête en table temporaire

LOAD AS SELECT chargement de la table par un select la table est appelé SYS_TEMP_0FD9D6605_A08741.

L'opération SORT AGGREGATE, c'est une opération qui permet le calcul des fonctions d'agrégat,

Sum dans le cas de cette requête.

VIEW, ce mots indique que SQL a dû lancer la création une table intermédiaire

Temporaire depuis une autre en parallèle de l'exécution de la requête

En utilisant les opérations sous les mots VIEW cette table permettra est utilisé dans

L'exécution de la requête. Nous voyons que les deux VIEW exécuté sont exécuté sur la même

Table mais ils n'ont pas le même en effet en appelant VIEW SQL ne va pas forcement parcourir toute la

Table il l'a parcours suivant ses besoins.

IX. Conclusion

Ce projet est le résultat de plusieurs jours de travail ainsi qu'une recherche assez approfondi dans le domaine de la location de véhicule afin de s'approcher au maximum du monde réel .Il reste néanmoins des corrections et des améliorations à envisager . La conception de départ nous permet d'apporter différents ajustement selon les exigences du client final tel que par exemple un pourcentage de réduction dans les contrats ou même étendre la base pour prendre en charge le processus de réparation des véhicules.

On envisage aussi ajouter