Rapport De Fin De Projet

Date: 2-25-2021

Année académique: 2020-2021

Département: Informatique

Matière: Base De Données 2

Code: I3306

Docteur Encadrant: Dr. Hamsa Hassrouny

Etudiants: Michel Sleiman Rahme 56312

Michael Saba 54851

Donnée

On veut établir la base de données d'une société de tourisme spécialisée dans les excursions en autobus. La base de données doit permettre de gérer les clients, de donner des informations sur les excursions du catalogue de la société, de gérer le parc de véhicules et le personnel de la société, et d'affecter les chauffeurs et hôtesses d'accompagnement ainsi que les clients ayant réservé une place et les autobus à chaque voyage, un voyage étant la réalisation d'une excursion à une date précise.

CLIENTS (numclient, nom, adresse, tel, email, **nb_reserv,** solde) C5, C20, C30, C10, C30, N
EXCURSION (nomex, libellé, départ, destination, descriptif, durée, prix) C20, C100, C20, C20, C100, N, N
EMPLOYE (numemp, nom, prenom, datenais, salaire) C5, C20, C20, D, N
BUS (numb, modele, annee, typebus) C10, C20, N, C10
RESERVATION (idr, idv, dater, numclient) N, N, D, C5
VOYAGE (idv, nomex, datev, bus, chauffeur, hôtesse, nbmax) N, C20, D, C10, C5, C5, N
PAYEMENT (nump, datep, montant, numc) N, D, N, C5

1. Ajouter les attributs en gras aux tables du schéma.
2. Créer une table **HIST_EX** qui contient pour chaque excursion et par année le nombre de voyages faits pour cette excursion, le nombre de réservations et le revenu

```
CREATE TABLE CLIENTS (
        numclient NUMBER(5) NOT NULL ,
        nom VARCHAR(20) NOT NULL,
        adresse VARCHAR(30) NOT NULL,
        tel NUMBER(8) NOT NULL),
        email VARCHAR(30) NOT NULL,
        nb_reserv NUMBER  NULL,
        solde NUMBER  NULL,

        CONSTRAINT PK_clients PRIMARY KEY (numclient)

);

CREATE TABLE EXCURSION (
        nomex VARCHAR(20) NOT NULL ,
        libelle VARCHAR(100) NOT NULL,
        depart varchar(20) NOT NULL,
        destination varchar(20) NOT NULL,
```

```
        descriptif varchar(100) NOT NULL,
        duree TIMESTAMP NULL,
        prix NUMBER  NULL,

        CONSTRAINT PK_excursion PRIMARY KEY (nomex)

);

CREATE TABLE EMPLOYEE (
        numemp NUMBER(5) NOT NULL ,
        nom VARCHAR(20) NOT NULL,
        prenom VARCHAR(20) NOT NULL,
        datenais DATE NOT NULL,
        salaire  NUMBER NULL,

        CONSTRAINT PK_employee PRIMARY KEY (numep)
);

CREATE TABLE BUS (
        numb NUMBER(10) NOT NULL ,
        modele VARCHAR(20) NOT NULL,
        annee NUMBER(4) NOT NULL,
        typebus VARCHAR(10) NOT NULL,

        CONSTRAINT PK_bus PRIMARY KEY (numb)
);

CREATE TABLE RESERVATION (
        idr NUMBER(5) NOT NULL ,
        idv NUMBER(5) NOT NULL,
        dater DATE NOT NULL,
        numclient NUMBER(5) NOT NULL,

        CONSTRAINT PK_reservation PRIMARY KEY (idr),

        CONSTRAINT FK_reservation FOREIGN KEY (idv)
   REFERENCES VOYAGE(idv)
);

CREATE TABLE VOYAGE (
        idv NUMBER(5) NOT NULL ,
        nomex VARCHAR(20) NOT NULL,
        datev DATE NOT NULL,
        bus NUMBER(10) NOT NULL,
        chauffeur NUMBER(5) NOT NULL,
        hotesse NUMBER(5) NOT NULL,
        nbmax NUMBER NULL,

        CONSTRAINT PK_voyage PRIMARY KEY (idv),

        CONSTRAINT FK_voyage0 FOREIGN KEY (nomex)
   REFERENCES EXCURSION(nomex),

   CONSTRAINT FK_voyage1 FOREIGN KEY (bus)
   REFERENCES BUS(numb),
```

```
   CONSTRAINT FK_voyage2 FOREIGN KEY (chauffeur)
   REFERENCES EMPLOYEE(numemp),

   CONSTRAINT FK_voyage3 FOREIGN KEY (hotesse)
   REFERENCES EMPLOYEE(numemp),
);

CREATE TABLE PAYEMENT (
        nump NUMBER(5) NOT NULL ,
        datep DATE NOT NULL,
        montant NUMBER NOT NULL,
        numc NUMBER(5) NOT NULL,

        CONSTRAINT PK_payement PRIMARY KEY (nump),

   CONSTRAINT FK_payement FOREIGN KEY (numc)
   REFERENCES CLIENTS(numclient)
);

CREATE TABLE HIST_EX (
        nomex VARCHAR(20) NOT NULL,
        annee NUMBER(4) NOT NULL,
        nbvoyage NUMBER NULL,
        nbreservation NUMBER NULL,
        revenu NUMBER NULL,

        CONSTRAINT PK_hist_ex PRIMARY KEY (nomex, annee)
);



CREATE TRIGGER UptCLIENTS
AFTER
INSERT ON RESERVATION
FOR EACH ROW

BEGIN

UPDATE CLIENT
SET CLIENT.nbreservation = CLIENT.nbreservation + 1;
WHERE RESERVATION.numclient = CLIENTS.nomclient;


END;
```

HIST_EX (nomex, annee, nbvoyages, nbreservations, revenu)

Fonctions

1. Trouver le nombre de voyages effectués par un type de bus donné depuis le début de l'année.

```
create or replace NONEDITIONABLE FUNCTION nbvoyage0(
        type IN varchar2
        )
RETURN number
IS
  total number := 0;

BEGIN
   SELECT COUNT(idv) INTO total
                FROM VOYAGE JOIN BUS
                ON VOYAGE.bus = BUS.numb
                        AND BUS.typebus = type
                        AND annee = EXTRACT(YEAR FROM CURRENT_DATE);

  RETURN total;
END;
```

2. Trouver pour une excursion donnée, le nombre de voyages faits à ce jour

```
create or replace NONEDITIONABLE FUNCTION nbvoyage1(
        nomex0 IN varchar
        )
RETURN number
IS
  total number := 0;

BEGIN
   SELECT COUNT(idv) INTO total
                FROM VOYAGE
                WHERE TO_CHAR(VOYAGE.datev, 'dd-mm-yyyy') = TO_CHAR(CURRENT_DATE,
'dd-mm-yyy')
                        AND nomex0 = VOYAGE.nomex;

  RETURN total;
END;
```

3. Trouver pour un client donné le nombre de réservations déjà faites

```
create or replace NONEDITIONABLE FUNCTION nbreservation(
        numclient IN number
        )
RETURN  number
IS
  total number := 0;

BEGIN
  SELECT COUNT(idr)  INTO total
                        FROM RESERVATION JOIN CLIENTS
                        ON CLIENTS.numclient = RESERVATION.numclient;

  RETURN total;
END;
```

Procédures

1. Créer pour chaque excursion existante un tuple dans HIST_EX, lire les voyages et les réservations déjà existants et mettre à jour la table HIST_EX et pour chaque client le nombre de ses réservations.

```
create or replace NONEDITIONABLE PROCEDURE HistEx_Fill
IS

CURSOR Histc IS SELECT nomex FROM EXCURSION;

 year varchar2(4) ;
 cur_var varchar2(100);
var1 Number  ;
var2 number;
var3 DATE;
BEGIN

SELECT COUNT(*) into var1 FROM EXCURSION;
SELECT COUNT(DISTINCT datev) into var2 FROM VOYAGE;
SELECT MIN(datev)INTO var3 FROM VOYAGE;
FOR b IN 1 .. var1
LOOP

OPEN Histc;

year := EXTRACT(YEAR FROM var3 ) - 1;
FETCH  Histc INTO cur_var;

        FOR a IN 1 .. var2
                LOOP
                INSERT INTO HIST_EX (annee, nomex)
                VALUES (year+1, cur_var);



        END LOOP;

CLOSE Histc;

END LOOP;

UPDATE HIST_EX
SET nbvoyage = (SELECT COUNT(*)
                                FROM VOYAGE  JOIN HIST_EX
```

```
                        ON EXTRACT(YEAR FROM VOYAGE.datev) = HIST_EX.annee
                                AND VOYAGE.nomex = HIST_EX.nomex);


UPDATE HIST_EX
SET nbreservation = (SELECT COUNT(*)
                        FROM RESERVATION JOIN VOYAGE
                        ON RESERVATION.idv = VOYAGE.idv
                                AND VOYAGE.nomex = HIST_EX.nomex
                                AND EXTRACT(YEAR FROM VOYAGE.datev) = HIST_EX.annee);


UPDATE HIST_EX
SET revenu =   ((SELECT COUNT(*)
                        FROM RESERVATION JOIN VOYAGE
                        ON RESERVATION.idv = VOYAGE.idv
                                AND VOYAGE.nomex = HIST_EX.nomex
                                AND EXTRACT (YEAR FROM VOYAGE.datev) = HIST_EX.annee) *
(SELECT EXCURSION.prix

        FROM EXCURSION INNER JOIN HIST_EX
                                 ON EXCURSION.nomex = HIST_EX.nomex));


END;


BEGIN
DBMS_SCHEDULER.create_job(
        job_name                => 'HistExUpt',
        job_type                => 'PLSQL_BLOCK',
        job_action              => 'DECLARE CURSOR nyUpt IS SELECT nomex FROM EXCURSION;
                                        DECLARE cur_ex varchar();

                                        OPEN nyUpt;

                                        FOR index 1 .. (SELECT COUNT(*) FROM EXCURSION)
                                                FETCH FROM EXCURSION INTO cur_ex;

                                                INSERT INTO HIST_EX (nomex, annee)
                                                VALUES (cur_ex, YEAR(GETDATE()));
                                        END LOOP;',

        start_date              => SYSTIMESTAMP,
        repeat_interval => 'freq=YEARLY; interval = 1',
        end_date                => NULL,
        enabled                 => TRUE,
        comments                => '');
```

```
        END;

END;
```

2. Augmenter d'un pourcentage donné le prix des excursions qui ont connu le plus grand nombre de réservations depuis le début de l'année.

```
create or replace NONEDITIONABLE PROCEDURE PFix
IS



max0 varchar(100);
 max1 varchar(100);
 max2 varchar(100);

BEGIN
SELECT  nomex INTO max0
        FROM HIST_EX
        WHERE nbreservation =  (SELECT MAX(nbreservation)
                                                FROM HIST_EX
                                                );



UPDATE EXCURSION
SET EXCURSION.PRIX = EXCURSION.PRIX*1.13
WHERE nomex = max0;



END;
```

3. Augmenter d'un pourcentage donné le salaire des employés, chauffeurs et hôtesses, qui ont été affectés au plus grand nombre de voyages depuis le début de l'année.

```
create or replace NONEDITIONABLE PROCEDURE SalFix
IS

 maxc number ;
 maxh number;

BEGIN


select cnt1.chauffeur into maxc
from (select COUNT(*) as total, chauffeur
    from VOYAGE
```

```
    group by chauffeur) cnt1,
   (select MAX(total) as maxtotal
    from (select COUNT(*) as total, chauffeur from VOYAGE group by CHAUFFEUR)) cnt2 , VOYAGE
where cnt1.total = cnt2.maxtotal
        AND EXTRACT(YEAR FROM VOYAGE.datev) = EXTRACT(YEAR FROM CURRENT_DATE);


select cnt1.hotesse into maxh
from (select COUNT(*) as total, hotesse
     from VOYAGE
     group by hotesse) cnt1,
    (select MAX(total) as maxtotal
     from (select COUNT(*) as total, hotesse from VOYAGE group by hotesse)) cnt2 , VOYAGE
where cnt1.total = cnt2.maxtotal
        AND EXTRACT(YEAR FROM datev) = EXTRACT(YEAR FROM CURRENT_DATE);


UPDATE EMPLOYEE
SET salaire = salaire * 1.12
WHERE numemp = maxc OR numemp = maxh;

END;
```

Triggers

1. Créer un nouveau tuple dans la table HIST_EX suite à l'insertion d'un nouveau tuple Excursion.

```
create or replace NONEDITIONABLE TRIGGER NewHist_EX
AFTER
INSERT ON EXCURSION
FOR EACH ROW

BEGIN

INSERT INTO HIST_EX (nomex, annee)
VALUES (:new.nomex, EXTRACT(YEAR FROM CURRENT_DATE));

END;
```

2. Mettre à jour nbvoyages dans la table HIST_EX suite à l'insertion d'un nouveau voyage.

```
create or replace NONEDITIONABLE TRIGGER UptHIST_EX0
AFTER
INSERT ON VOYAGE
FOR EACH ROW

BEGIN
```

```
UPDATE HIST_EX
SET HIST_EX.nbvoyage = HIST_EX.nbvoyage + 1
WHERE (HIST_EX.nomex = :new.nomex
        AND HIST_EX.annee = EXTRACT(YEAR FROM CURRENT_DATE));

END;
```

3. Mettre à jour nbreservations dans la table HIST_EX suite à l'insertion d'une nouvelle réservation.

```
 create or replace NONEDITIONABLE TRIGGER UptHIST_EX1
AFTER
INSERT ON RESERVATION
FOR EACH ROW

BEGIN

UPDATE HIST_EX
SET HIST_EX.nbreservation = HIST_EX.nbreservation + 1
WHERE (:new.idv = voyage.idv
   AND HIST_EX.nomex = voyage.nomex
        AND HIST_EX.annee = EXTRACT(YEAR FROM CURRENT_DATE));

END;
```

4. Interdire, pour un voyage donné, un nombre de réservations qui dépasse le nbmax de ce voyage.

```
create or replace NONEDITIONABLE TRIGGER MAX_res
AFTER
INSERT ON RESERVATION
FOR EACH ROW

DECLARE var1 NUMBER;
      var2 NUMBER;
BEGIN

SELECT COUNT(*)INTO var1
                FROM RESERVATION
                WHERE RESERVATION.idv = :new.idv ;

SELECT nbmax INTO var2
      FROM VOYAGE
                WHERE VOYAGE.idv = :new.idv;

IF (var1 > var2 )
```

```
THEN DBMS_OUTPUT.PUT_LINE( 'Le nombre de reservation pour le voyage souhaite
                a atteint son maximum ');
    DELETE FROM RESERVATION
    WHERE RESERVATION.idr = :new.idr;
END IF;
END;
```

5. Suite à l'insertion d'une réservation, mettre à jour le solde du client et le nombre de ses réservations.

```
create or replace NONEDITIONABLE TRIGGER NewRes
AFTER
INSERT ON RESERVATION
FOR EACH ROW
DECLARE var1 number;
BEGIN


SELECT prix INtO var1
FROM VOYAGE, EXCURSION, RESERVATION
WHERE voyage.nomex = excursion.nomex
AND :new.idv = voyage.idv;



UPDATE CLIENTS
SET CLIENTS.solde = CLIENTS.solde + var1;


END;
```

6. Suite à l'insertion d'un paiement, mettre à jour le solde du client.

```
create or replace NONEDITIONABLE TRIGGER NewPay
AFTER
INSERT ON PAYEMENT
FOR EACH ROW

BEGIN

UPDATE CLIENTS
SET CLIENTS.solde = CLIENTS.solde - (:new.montant)
WHERE :new.numc = CLIENTS.numclient;


END;
```