Introduction aux bases de donnée relationnelles

TP2 – Création des bases de données, Contraintes d'intégrités

Nom : Djebien
Prenom : Tarik
Groupe : 2
Date : 2/ 11/ 2010

A) Reconstruire la base de données INFOTOUR

A-1/
Voir fichier MON_INFOTOUR.sql

A-2/

non l'ordre des DROPS n'est pas important dans ce cas précis.

A-3/

On observe des valeurs qui ne respectent pas le type correspondant à la propriété associée.
Exemple pour un VARCHAR(50) : valeur trop grande pour la colonne
"MUSEES"."DESCRIPTION" (réelle : 117, maximum : 50)

On observe aussi des violation de contrainte d'intégrité car les clé parents sont introuvables.
Exemple : insert into hotels values(12,'du Midi',3,'Bd du schmilblick','Essaye de donner l''illusion d''y être',2,0)
le directeur 0 n'existe pas en tant que numper 0 dans la table personnes.

A-4 et A-5/

violation de contrainte d'intégrité (DJEBIEN.FK_HOTELS_DIRECTEUR) - clé parent introuvable
sql > ALTER TABLE hotels DROP CONSTRAINT FK_HOTELS_DIRECTEUR;

violation de contrainte d'intégrité (DJEBIEN.FK_CATCHAMBRES_NUMHOTEL) - clé parent introuvable
sql > ALTER TABLE catchambres DROP CONSTRAINT FK_CATCHAMBRES_NUMHOTEL;

violation de contrainte d'intégrité (DJEBIEN.FK_MONUMENTS_DIRECTEUR) - clé parent introuvable
sql > ALTER TABLE monuments DROP CONSTRAINT FK_MONUMENTS_DIRECTEUR;

violation de contrainte d'intégrité (DJEBIEN.FK_UTILISER_NUMTRSP) - clé parent introuvable
sql > ALTER TABLE utiliser DROP CONSTRAINT FK_UTILISER_NUMTRSP;

violation de contrainte d'intégrité (DJEBIEN.FK_UTILISER_NUMUTOUR) - clé parent

introuvable
sql > ALTER TABLE utiliser DROP CONSTRAINT FK_UTILISER_NUMUTOUR;

violation de contrainte d'intégrité (DJEBIEN.FK_ACHETER_NUMPER) - clé parent introuvable
sql > ALTER TABLE acheter DROP CONSTRAINT FK_ACHETER_NUMPER;

violation de contrainte d'intégrité (DJEBIEN.FK_ACHETER_NUMTOUR) - clé parent introuvable

sql > ALTER TABLE acheter DROP CONSTRAINT FK_ACHETER_NUMTOUR;

On vide les enregistrements :

sql > delete * from hotels
sql > delete * from catchambres
sql > delete * from monuments
sql > delete * from utiliser
sql > delete * from acheter

On recommence l'insertion des valeurs :
On constate alors que sans les contraintes d'intégrité référentiels précédentes, les insertions se font sans aucun contrôle et donc sans aucune erreur.

A-6/

ALTER TABLE hotels ADD CONSTRAINT FK_hotels_directeur FOREIGN KEY (directeur) REFERENCES personnes(numper);
ALTER TABLE catchambres ADD CONSTRAINT FK_catchambres_numhotel FOREIGN KEY (numhotel) REFERENCES hotels(numhotel);
ALTER TABLE monuments ADD CONSTRAINT FK_monuments_directeur FOREIGN KEY (directeur) REFERENCES personnes(numper);
ALTER TABLE utiliser ADD CONSTRAINT FK_utiliser_numtrsp FOREIGN KEY (numtrsp) REFERENCES transports(numtrsp);
ALTER TABLE utiliser ADD CONSTRAINT FK_utiliser_numutour FOREIGN KEY (numutour) REFERENCES tours(numtour);
ALTER TABLE acheter ADD CONSTRAINT FK_acheter_numper FOREIGN KEY (numper) REFERENCES personnes(numper);
ALTER TABLE acheter ADD CONSTRAINT FK_acheter_numtour FOREIGN KEY (numtour) REFERENCES tours(numtour);

On constate que de nouveau le SGBD Oracle nous informe des erreurs vu précédemment lors de l'ajout des contraintes d'intégrités référentielle par ALTER TABLE, on peut donc conclure que le SGBD vérifie la cohérence de la base complète à chaque ajout de contraintes d'intégrités.

B) Étude de la cohérence des enregistrements dans la base INFOTOUR

B-1/

Pour la clé étrangère numpays dans la table villes, on doit vérifier qu'il n'existe pas de ville ayant un numpays n'appartenant pas au numpays de la table pays :

SELECT v.numpays FROM villes AS v WHERE v.numpays NOT IN (SELECT numpays from pays);

idem pour personnes :
SELECT p.numpays FROM personnes AS p WHERE p.numpays NOT IN (SELECT numpays from pays);

Meme principe pour musees :
SELECT numville FROM musees WHERE numville NOT IN (SELECT numville from villes);
SELECT directeur FROM musees WHERE directeur NOT IN (SELECT numper from personnes);

Idem pour hotels :
SELECT numville FROM hotels WHERE numville NOT IN (SELECT numville from villes);
SELECT directeur FROM hotels WHERE directeur NOT IN (SELECT numper from personnes);

Idem pour catchambres :
SELECT numhotel FROM catchambres WHERE numhotel NOT IN (SELECT numhotel from hotels);

Idem pour monuments :
SELECT numville FROM monuments WHERE numville NOT IN (SELECT numville from villes);
SELECT directeur FROM monuments WHERE directeur NOT IN (SELECT numper from personnes);

Idem pour tours :
SELECT numpays FROM tours WHERE numpays NOT IN (SELECT numpays from pays);

Idem pour transports :
SELECT depart FROM transports WHERE depart NOT IN (SELECT numville from villes);
SELECT arrivee FROM transports WHERE arrivee NOT IN (SELECT numville from villes);

Finalement pour utiliser et acheter :
SELECT numutour FROM utiliser WHERE numutour NOT IN (SELECT numtour from tours);
SELECT numutrsp FROM utiliser WHERE numtrsp NOT IN (SELECT numtrsp from transports);

SELECT numper FROM acheter WHERE numper NOT IN (SELECT numper from personnes);
SELECT numtour FROM acheter WHERE numtour NOT IN (SELECT numtour from tours);

B-2/

Concernant toutes les erreurs du type :
violation de contrainte d'intégrité (USERNAME.FK_TABLENAME_FOREIGNKEYNAME) - clé parent introuvable

On a une clé étrangère reliant un attribut d'une table X, a une table Y.
La clé primaire de Y est une clé étrangère pour X.

On a 2 possibilités :

- Nous pouvons y remédier en effectuant une mise à jour directement dans la table Y en ajoutant une ligne faisant référence à la clé parente manquante dans la table Y à partir des valeurs obtenues par le :
  sql > INSERT TO X VALUES(val1 , val2 , val3 , ... , valCléParentIntrouvable , ... , valN);
  en faisant préalablement :
  sql > INSERT TO Y VALUES(valCléParentIntrouvable , val2 , val3 , ... , valN);

Ceci règlerai provisoirement le problème au niveau de l'Analyse Syntaxique de SQL par le SGBD mais sur long terme créerai une incohérence sur le point de vue Sémantique de la Base de données INFOTOUR.
En effet, il serait anormale de pouvoir ajouter par exemple un monuments dont le directeur n'est pas référencé dans la table "personnes" étant donné qu'un directeur est une instance de personnes par définition, cela reviendrait à attribuer la gestion d'un monuments par un directeur inexistant ce qui est totalement absurde sur le plan pratique en condition réelle.

- L'idéal serait de proposer à l'utilisateur rencontrant ce type d'erreur de choisir une clé étrangère présente dans la table Y en tant que clé primaire, ceci permettrait alors par exemple, en reprenant l'exemple de "monuments" et "personnes" de choisir un directeur existant dans "personnes" qui serait alors disponible pour s'occuper de la gestion du monuments en question.
Sur un plan pratique en condition réelle, cette solution est meilleur car on aura alors la certitude que chacun des monuments dispose d'un directeur réelle et non imaginaire comme présenté dans la première solution.

On peut trivialement adapté le même raisonnement entre "musees" et "personnes" ou encore "catchambres" et "hotels".

Concernant toutes les erreurs du type :

valeur trop grande pour la colonne "NOMTABLE"."NOMPROPRIETE" (réelle : y, maximum : x) avec y > x

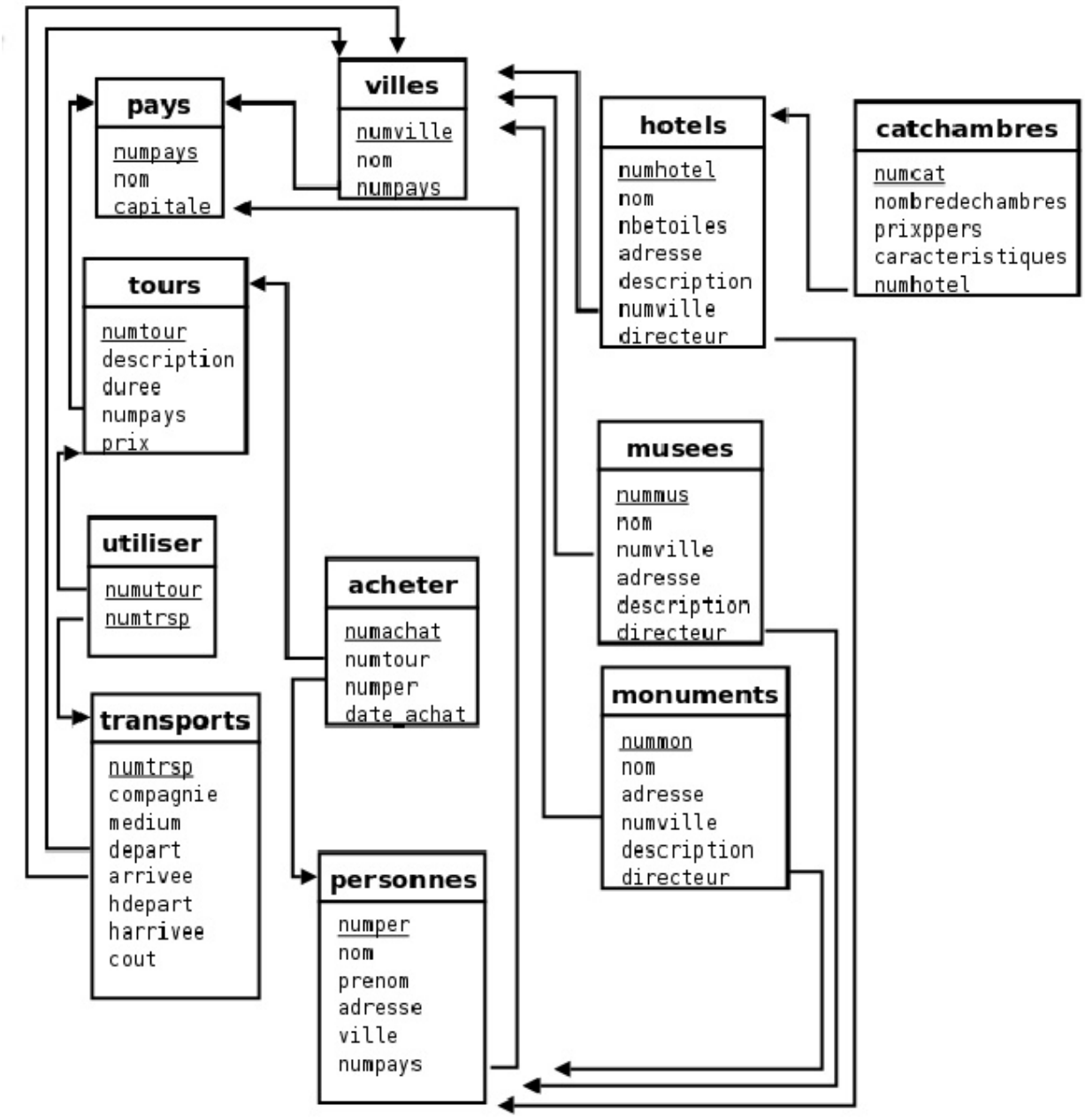
On a 3 possibilités :

- Nous pouvons y remédier en tronquant la valeur avec une taille maximale de x caractères.
  sql > UPDATE NOMTABLE SET NOMPROPRIETE = Substr(NOMPROPRIETE,1,x) WHERE LENGTH(NOMPROPRIETE) > x (avec un trigger BEFORE insertion);
  Cependant, sur le plan pratique cela reviendrait alors à enregistrer par exemple des descriptions de musées ou monuments incomplètes, ce qui nuirait donc à l'ergonomie de l'application ainsi qu'à la relation client (visiteur du musées) / entreprise (le musée) sur l'image du service que le musée propose au visiteur.

- On pourrait aussi changer le type de l'attribut NOMPROPRIETE en le passant de VARCHAR(x) à VARCHAR(y) mais ceci est à éviter car cela reviendrait à allouer plus d'espace mémoire dans toutes les tables rencontrant ce problème pour chaque utilisateur diffèrent qui saisirait leur donnée selon leur convenance sans limitation, cette méthode est inutilisable en pratique réelle.

- L'idéal serait de proposer à l'utilisateur lors de la saisie des données (par exemple au niveau d'un formulaire) de ressaisir une description qui ne dépasse pas le nombre de caractères maximales (en refusant d'envoyer le formulaire par exemple).
  Celui-ci devrait alors résumer sa description plus brièvement mais cela restera présentable par la suite au visiteur du musée.

MLD INFOTOUR



C) Établir les contraintes d'intégrité référentielle

C -1/

```
ALTER TABLE hotels ADD CONSTRAINT FK_hotels_directeur  FOREIGN KEY (directeur)
REFERENCES personnes(numper);
ALTER TABLE catchambres ADD CONSTRAINT FK_catchambres_numhotel  FOREIGN KEY
(numhotel) REFERENCES hotels(numhotel);
ALTER TABLE monuments ADD CONSTRAINT FK_monuments_directeur FOREIGN KEY
(directeur) REFERENCES personnes(numper);
ALTER TABLE utiliser ADD CONSTRAINT FK_utiliser_numtrsp  FOREIGN KEY (numtrsp)
REFERENCES transports(numtrsp);
ALTER TABLE utiliser ADD CONSTRAINT FK_utiliser_numutour FOREIGN KEY (numutour)
REFERENCES tours(numtour);
ALTER TABLE acheter ADD CONSTRAINT FK_acheter_numper   FOREIGN KEY (numper)
REFERENCES personnes(numper);
ALTER TABLE acheter ADD CONSTRAINT FK_acheter_numtour  FOREIGN KEY (numtour)
REFERENCES tours(numtour);
```

C - 2/

l'identifiant de la personne ayant dépensé le plus :

R1 :

```
SELECT IDpers FROM
                    (
                     SELECT personnes.numper AS IDpers, SUM(tours.prix) AS depense
                     FROM personnes , acheter , tours
                     WHERE personnes.numper = acheter.numper AND acheter.numtour =
                      tours.numtour
                     GROUP BY personnes.numper
                    )
WHERE depense =
          ( SELECT MAX(depense) FROM
                              (SELECT personnes.numper AS IDpers, SUM(tours.prix) AS depense
                               FROM personnes , acheter , tours
                               WHERE personnes.numper = acheter.numper AND acheter.numtour =
                               tours.numtour
                               GROUP BY personnes.numper
                              )
          )
;
```

Modifié l'identifiant de la personne satisfaisant (R1) à 100001 :

R2 :

```
UPDATE personnes
SET numper = 100001
WHERE numper = R1;
```

a)

```
ALTER TABLE personnes DROP CONSTRAINT PK_numper;
R2;
ALTER TABLE personnes ADD CONSTRAINT PK_numper PRIMARY KEY (numper);
```

b) Sans ON UPDATE CASCADE et sans triggers, on doit modifier prealablement toute les tables ou la clé primaire apparait comme clé etrangere puis en dernier lieu la table contenant la clé primaire :

```
UPDATE hotels
SET directeur = 100001
WHERE directeur = R1;

UPDATE musees
SET directeur = 100001
WHERE directeur = R1;

UPDATE monuments
SET directeur = 100001
WHERE directeur = R1;

UPDATE acheter
SET numper = 100001
WHERE numper = R1;

UPDATE personnes
SET numper = 100001
WHERE numper = R1;
```

C - 3 /

les tours ayant rapportés le moins d'argent à l'agence :

R3 :

```
SELECT numtour FROM
                        (SELECT  numtour , Count ( numtour) AS nbre_acheter
                         FROM acheter
                         GROUP BY numtour
                        )
WHERE nbre_acheter =
                         (SELECT MIN(nbre_acheter) FROM
                                  (SELECT  numtour , Count ( numtour) AS nbre_acheter
                                   FROM acheter
                                    GROUP BY numtour
                                  )
                         )
;
```

R4 : On supprime les tours satisfaisant (R3) dans la table « tours »:

```
DELETE numtour
FROM tours
WHERE numtour = R3;
```

a)

```
ALTER TABLE tours DROP CONSTRAINT PK_numtour;
ALTER TABLE acheter DROP CONSTRAINT FK_acheter_numtour;
R4;
ALTER TABLE tours ADD CONSTRAINT PK_numtour PRIMARY KEY (numtour);
ALTER TABLE acheter ADD CONSTRAINT FK_acheter_numtour  FOREIGN KEY (numtour)
REFERENCES tours(numtour);
```

b)

Sans ON DELETE CASCADE et sans triggers, on doit supprimer prealablement dans toute les tables ou la clé primaire apparait comme clé etrangere
les valeurs concernées puis en dernier lieu les valeurs concernées dans la table contenant la clé primaire :

```
DELETE numtour
FROM acheter
WHERE numtour = R3;

DELETE numtour
FROM tours
WHERE numtour = R3;
```

c)

ALTER TABLE tours
MODIFY CONSTRAINT PK_numtour PRIMARY KEY (numtour)
ON DELETE CASCADE;

ALTER TABLE acheter
MODIFY CONSTRAINT FK_acheter_numtour FOREIGN KEY (numtour)
REFERENCES tours(numtour)
ON DELETE CASCADE;

R4 ;

Schéma relationnel de la base

pays(**numpays** number, nom varchar(20), capitale varchar(20)) ;

villes(**numville** number, nom varchar(20), numpays number) ;

personnes(**numper** number, nom varchar(30), prenom varchar(20), adresse varchar(30), ville varchar(30), numpays number) ;

musees(**nummus** number, nom varchar(30), numville number, adresse varchar(30), description varchar(50), directeur number) ;

hotels(**numhotel** number, nom varchar(30), nbetoiles number, adresse varchar(50), description varchar(300),num_ville number, directeur number) ;

catchambres(**numcat** number, nombredechambres number, prixppers number, caracteristiques varchar(300), numhotel number) ;

monuments(**nummon** number, nom varchar(30), adresse varchar(30), numville number, description varchar(300), directeur number) ;

tours(**numtour** number, description varchar(100), duree number, numpays number, prix number(8,2)) ;

transports(**numtrsp** number, compagnie varchar(30), medium varchar(20), depart number, arrivee number, hdepart char(5), harrivee char(5), cout number(8,2)) ;

utiliser(**numutour number**, **numtrsp number**) ;

acheter(**numachat** number,numper number, numtour number, date_achat char(10)) ;

D) Étendre la base de données (modélisation → MCD / UML