TP4

1) Contraintes:

Nous revenons maintenant de manière plus approfondie sur les contraintes portant sur le contenu des tables. Il est très important de spécifier les contraintes dans le schéma, afin que le SGBD puisse les prendre en charge. Cela évite d'effectuer des contrôles de manière répétitive dans les applications qui accèdent à la base.

Les contraintes essentielles portant sur les clés simples, primaires et étrangères ont déjà été vues. Les contraintes décrites ici sont moins spécifiques et portent sur les valeurs des attributs ou plus globalement sur le contenu d'une ou plusieurs relations. La commande **CHECK (condition)**

- **On considère le shéma relationnel suivant :**

Cinéma (**nomCinéma**, numéro, rue, ville)
Salle (**nomCinéma, no**, capacité, climatisée)
Film (**idFilm**, titre, année, genre, résumé, idMES)
Séance (**idFilm**, **nomCinéma**, **noSalle**, tarif)

- Indiquez comment exprimer les contraintes suivantes sur le schéma relationnel avec la syntaxe SQL2. (Lors de l'insertion des tuples à chaque table donner un contre example et vérifier si le check fonctionne bien)

1- Un Cinéma doit faire partie de la liste des villes {Alger, Oran, Constantine, Adrar}

ville CHAR(20) CHECK (ville IN (‘*Alger’, ‘Oran’, ‘Constantine’, ‘Adrar’***)),**

2- On restreint les valeurs possibles des attributs **capacité <300** et **climatisée** (soit oui ou non) dans la table *Salle*.

```
CREATE TABLE Salle
(nomCinema VARCHAR (30) NOT NULLL,
no INTEGER,
capacite INTEGER CHECK (capacite < 300),
climatisee CHAR(1) CHECK (climatisee IN ('O','N')),
PRIMARY KEY (nomCinema, no),
FOREIGN KEY nomCinema REFERENCES Cinema);
```

Il s'agit de contraintes portant sur des valeurs d'attributs. A chaque insertion d'un tuple, ou mise-à-jour de ce tuple affectant un des attributs contraints, le contrôle sera effectué.

3- *L'année de parution d'un film est toujours connue*

année INTEGER NOT NULLL

4- *Le tarif d'une séance est supérieur à 10 euro.*

tarif INTEGER CHECK (tarif > 10)

2) Chaines de caractères:

- **Donner trois fois les tarifs des films dont le nom contient « l »**

```
SELECT tarif * 3 FROM Film
WHERE titre LIKE '%l%'
```

- **Donner les Cinéma dont leurs villes commencent par 'A'**

```
SELECT nomCinéma FROM Cinéma
WHERE ville LIKE 'A%'
```

- **Donner les films qui ne contient pas le mot « bataille » ((la bataille d'Alger))**

```
SELECT * FROM Film
WHERE titre NOT LIKE '%bataille%'
```

- **Donner les films dont leurs genres commencent par un caractère suivi par 'or' ((horreur))**

```
SELECT titre, genre FROM Film
WHERE genre LIKE '_or' ;
```

3) Opérations Algébriques:

- **Donner les cinémas qui se trouvent à Alger ou qui ont une salle de capacité supérieure à 200 places (union)**

```
SELECT nomCinéma FROM Cinéma WHERE ville = 'Alger'
UNION
SELECT nomCinéma FROM Salle  WHERE capacité > 200;
```

- **Donner les cinémas sans salles (différence)**

```
SELECT nomCinéma FROM Cinéma
EXCEPT
SELECT nomCinéma FROM Salle  ;
```

Equivalent:

```
SELECT nomCinéma FROM Cinéma
WHERE nomCinéma  MINUS  ( SELECT nomCinéma FROM Salle)  ;
```

Equivalent:

```
SELECT nomCinéma FROM Cinéma
WHERE nomCinéma  NOT IN ( SELECT nomCinéma FROM Salle)  ;
```

Equivalent:

```
SELECT nomCinéma FROM Cinéma
WHERE nomCinéma  NOT EXISTS ( SELECT nomCinéma FROM Salle)  ;
```

- **Donner les cinémas qui se trouvent à Constantine et possèdent des salles climatisée (Intersection)**

```
SELECT nomCinéma FROM Cinéma WHERE ville = 'Constantine'
INTERSECT
SELECT nomCinéma FROM Salle  WHERE climatisé= 'O';
```

- **Donner les films qui se passent au cinéma d'Oran et dans des salles climatisées (Intersection + Jointure)**

```
SELECT titre FROM Film F, Cinéma C , Séance S   WHERE ville = ' Oran' AND F.
idFilm = S. idFilm AND S.nomCinéma = C.nomCinéma
INTERSECT
SELECT titre FROM Film F, Salle L, Séance S   WHERE climatisé = ' O' AND F. idFilm =
S. idFilm AND S.noSalle = L.no;
```

- **Donner les films qui ont un tarif au moins supérieur à 70**

```
SELECT * FROM Film  F
WHERE  EXISTS  ( SELECT *  FROM Séance S
                                  WHERE S.idFilm = F.idFilm AND S.tarif > 70)  ;
```

- **Donner les numéros des salles qui ne contiennent aucun film**

```
SELECT no  FROM Salle  L
WHERE  NOT EXISTS  ( SELECT *  FROM Séance S, Film F
                                  WHERE S.idFilm = F.idFilm AND S.noSalle= L.no)  ;
```

- **Donner tous les films et salles (produit)**

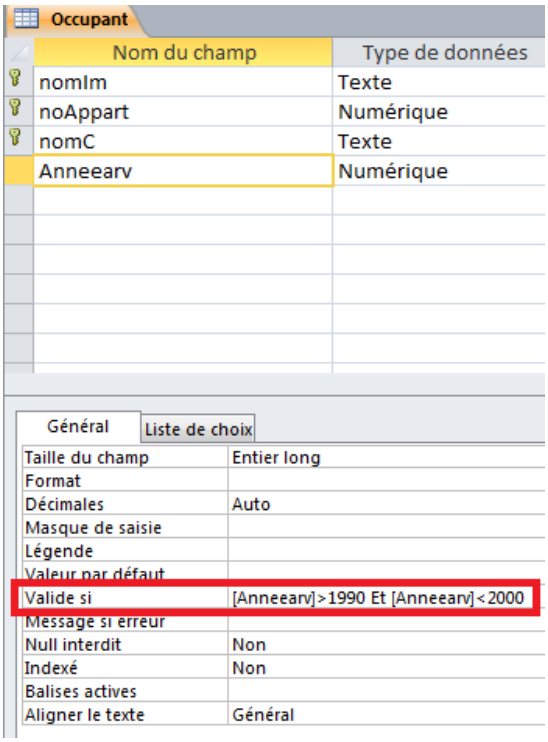
```
SELECT *  FROM Film, Salle;
```

Equivalent:

```
SELECT *  FROM Film CROSS JOIN  Salle;
```

TP5

Créer la table Occupant de l'exo 3 de la série de TD, en précisant la contrainte de l'année d'arrivée qu'elle soit dans l'intervalle [1990, 2000] (donner des contres exemples)



- Donner le code SQL des requêtes suivantes :

1) Donner l'habitant le plus ancien dans l'immeuble 1.
   SELECT nomC FROM Occupant WHERE AnneeArv =
   (SELECT MIN (AnneeArv) FROM Occupant WHERE nomIm= "Imb1");

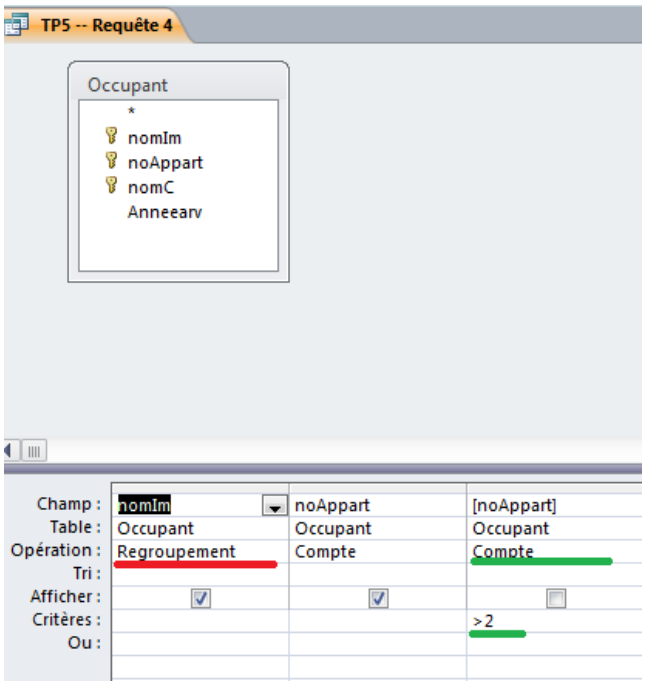
2) Donner les occupants ordonnés par leur année d'arrivé et noms
   SELECT AnneeArv, nomC FROM Occupant ORDER BY AnneeArv, nomC;

3) Donner le nombre des habitants de chaque immeuble.
   SELECT nomIm, count(*) FROM Occupant GROUP BY nomIm;

4) Donner le nombre des habitants arrivant en 1994 pour chaque immeuble
   SELECT nomIm, count(*) FROM Occupant WHERE AnneeArv= 1994
   GROUP BY nomIm;

5) Donner l'immeuble qui a plus que deux appartements occupés
   SELECT nomIm, Count (noAppart) FROM Occupant
   GROUP BY nomIm HAVING Count(noAppart) >2;

Regroupement avec Access:



6) Donner les immeubles où tout le monde a emménagé en 1994
SELECT DISTINCT nomIm FROM Occupant WHERE nomIm
NOT IN (select nomIm FROM Occupant WHERE Anneearv <> 1994);

7) Donner les Immeubles avec les noms des occupants occupés après 1993
SELECT nomIm, nomC FROM Occupant WHERE nomC **= ANY**
(SELECT nomC FROM Occupant WHERE AnneeArv > 1993);
(peut être simple)

8) Donner le dernier occupant par année de l'immeuble 1 (**ALL**)
SELECT nomC FROM Occupant WHERE Anneearv **>= ALL**
(SELECT Anneearv FROM Occupant WHERE nomIm= "Imb1");

9) Donner les immeubles où personne n'a emménagé en 1996
SELECT DISTINCT nomIm FROM Occupant WHERE nomIm NOT IN
(SELECT nomIm FROM Occupant WHERE Anneearv= 1996);

Créer la table Immeuble de l'éxo 3 de la série

10) Qui n'habite pas un appartement qu'il gère lui-même ? (requête 17 de TD)
SELECT O.nomC FROM Occupant AS O, Immeuble AS I
WHERE O.nomIm = I.nomIm AND O.nomC <> I.nomGérant;

11) Qui n'habite pas un appartement géré par Houari ? (requête 16 de TD)
SELECT Occupant.nomC FROM Immeuble INNER JOIN Occupant ON
Immeuble.nomIm = Occupant.nomIm WHERE (((Immeuble.nomGérant)<>'Houari'));

12) Qui gère l'appartement où habite Djamila ? (requête 6 de TD)
SELECT I.nomGérant FROM Immeuble AS I INNER JOIN Occupant AS O ON
I.nomIm=O.nomIm WHERE O.nomC= "Djamila";