



# **Administration des Bases de Données**

**L3 2018-2019**

**Mr H.MATALLAH**

# Plan de la matière

1. Notions fondamentales
2. SQL Avancé
3. Gestion d'intégrité et de cohérence
4. Vues et Index
5. Optimisation des requêtes
6. Gestion des transactions : Gestion des accès concurrents

## CHAPITRE 3

### GESTION D'INTÉGRITÉ ET DE COHERENCE

# Plan Chapitre 3

- 1. Définitions**
- 2. Manières de définition des contraintes d'intégrité**
  - Contrainte de colonne
  - Contrainte de table
  - Ajout de la contrainte ultérieurement
- 3. Niveaux de définition des contraintes d'intégrité**
  - Contrainte Intra-Tables Verticale
  - Contrainte Intra-Tables Horizontale
  - Contraintes Inter-Tables
- 4. Contrainte d'intégrité référentielle**
- 5. Suppression, activation/désactivation, affichage de contraintes**

# Gestion de Cohérence

## ■ Définitions

- **Cohérence ou Consistence** : Il s'agit de prendre les données dans un état cohérent et les rendre dans un état cohérent
- **B.D Cohérente** : BD où toutes les contraintes d'intégrité définies sont vérifiées
- **Contraintes d'intégrité** : Ensemble de règles (Contraintes de clé, Contraintes référentielles, Contraintes de domaines, Dépendances fonctionnelles, etc..), et permettant d'assurer une certaine cohérence de la BD
- **Violation de contrainte** : Une mise à jour qui provoque la violation d'une CI est refusée

# Gestion de Cohérence

## ■ Définitions

- **Les CI** : Complètent le schéma de la base en ajoutant une sémantique au modèle relationnel
- **Les CI** : Moyen offert pour permettre à l'administrateur de décrire les règles logiques que doivent respecter les données pour que la base soit cohérente
- **Les CI** : Sont traitées au niveau des données et non pas placées au niveau des traitements, ce qui est plus lourd à gérer
- Le SGBD veille à ce que toutes les contraintes soient vérifiées à chaque insertion, suppression, ou modification d'une donnée

# Gestion de Cohérence

## ■ *Définitions*

- On doit définir la syntaxe et la sémantique des contraintes :
  - **Syntaxe** : c'est le langage autorisé pour définir une contrainte (Forme de la contrainte)
  - **Sémantique** : Ensemble de conditions devant être remplies par les données (Sens de la contrainte)

# Gestion de Cohérence

## ■ *Types de contraintes d'intégrité*

- **Contrainte d'intégrité statique**

- ✗ Porte sur l'état de la base à chaque instant
- ✗ Respectée pour chacun des états de la BD

- **Contrainte d'intégrité dynamique (Triggers ou Déclencheurs)**

- ✗ Décrit le lien logique qui existe entre les états de la base aux instants  $t$  et  $t+1$
- ✗ Contrôle le passage d'un état à un autre
- ✗ Les lignes du nouvel état dépendant de ceux de l'ancien



# Gestion de Cohérence

## ■ *Définition des contraintes d'intégrité*

Les contraintes peuvent être déclarées de 3 manières :

- **Contrainte de colonne** (Inline constraints)

- ✗ Dans la ligne de définition de l'attribut : En même temps que la colonne (valable pour les contraintes monocolones)

- **Contrainte de table** (Out-of-line constraints)

- ✗ Après la déclaration des colonnes : Une fois la colonne déclarée, ces contraintes ne sont pas limitées à une colonne et peuvent être personnalisées par un nom

- **Ajout de la contrainte ultérieurement**

- ✗ Reporter la déclaration de la contrainte après la définition de la table

# Gestion de Cohérence

## ■ *Définition des contraintes d'intégrité*

### ● **Inline constraints**

- ✗ **NOT NULL** : Force la saisie d'une colonne
- ✗ **DEFAULT** : Précise une valeur par défaut
- ✗ **UNIQUE** : Unicité de l'attribut
- ✗ **PRIMARY KEY** : Définition d'une clé primaire simple
- ✗ **FOREIGN KEY** : Définition d'une clé étrangère
- ✗ **CHECK** : Contrainte de domaine (Spécifie les valeurs acceptables pour une colonne)
- ✗ **Type** et **Taille** d'une colonne sont des contraintes de **domaine** définies en ligne

# Gestion de Cohérence

## ■ *Définition des contraintes d'intégrité*

- **Inline constraints : Exemple**

```
CREATE TABLE Film (Titre VARCHAR (50) PRIMARY KEY,  
Annee INTEGER CHECK (Annee BETWEEN 1890 AND 2017) NOT NULL,  
Genre VARCHAR (10) CHECK (Genre IN ('Histoire','Western','Drame','Familial','Aventures')),  
Prix NUMBER(10) UNIQUE,  
Realisateur VARCHAR (20) REFERENCES Artiste(Code),  
CodePays INTEGER DEFAULT 00213) ;
```

# Gestion de Cohérence

## ■ Définition des contraintes d'intégrité

### ● Inline constraints : Exemple

Page d'accueil > SQL > Commandes SQL

☒ Validation automatique Afficher 10 ▼

```
CREATE TABLE HOTEL
(CODE_HOTEL VARCHAR2(4) NOT NULL,
CLASSE     NUMBER(3),
VILLE     CHAR(20) DEFAULT 'TLEMCEN',
NOM_HOTEL  CHAR(20) NOT NULL);
```

Résultats Expliquer Décrire SQL enregistré Historique

- ✗ **Rmq** : On ne peut pas accorder des noms aux contraintes définies, c'est le SGBD qui va les créer avec des noms générés automatiquement

# Gestion de Cohérence

## ■ Définition des contraintes d'intégrité

### ● Out-of-line constraints

```
CREATE TABLE (Attribut1 TYPE, Attribut2 TYPE,  
              Contrainte_intégrité1,  
              Contrainte_intégrité2, ...);
```

*Oracle recommande de déclarer les contraintes NOT NULL en ligne, les autres peuvent être définies soit en ligne, soit nommées après la déclaration des colonnes*

- ✗ **Clé primaire composée** : [CONSTRAINT nom\_contrainte] PRIMARY KEY (attr\_clé1 [,attr\_clé2, ...])
- ✗ **Clé unique composée** : [CONSTRAINT nom\_contrainte] UNIQUE(Attribut1 [,attribut2, ...])
- ✗ **Clé étrangère** référençant une clé primaire composée
- ✗ **Condition à vérifier sur plusieurs colonnes** : [CONSTRAINT nom\_contrainte] CHECK (condition)
- ✗ **Donner un nom à une contrainte** : [CONSTRAINT nom\_contrainte]

# Gestion de Cohérence

## ■ Définition des contraintes d'intégrité

### ● Out-of-line constraints : Contrainte d'intégrité référentielle

- ✗ Clé étrangère : [CONSTRAINT nom\_contrainte] FOREIGN KEY (attribut\_clé\_ét) REFERENCES tablepère (attribut)

Page d'accueil > SQL > Commandes SQL

☒ Validation automatique Afficher 10 ▼

```
CREATE TABLE FACTURE
(CODE_FACT VARCHAR2(60),
 DATE_FACT DATE NOT NULL,
 CODE_CLIENT VARCHAR(60),
 CONSTRAINT pk_FACTURE PRIMARY KEY(CODE_FACT),
 CONSTRAINT fk_CLIENT FOREIGN KEY(CODE_CLIENT)
REFERENCES CLIENT(CODE_CLIENT));
```

- ✗ Rmq : Si on ne précise pas le nom de la contrainte [CONSTRAINT nom\_contrainte], le système va attribuer un nom de contrainte automatique similaire à SYS\_C00.....

# Gestion de Cohérence

## ■ *Définition des contraintes d'intégrité*

- **Out-of-line constraints** : **Contrainte d'intégrité référentielle** (Cas particuliers)

✗ **Q1** : Est-ce qu'on peut référencier un attribut dans la table père autre que la clé primaire?

**Réponse** : **Oui**, mais à condition d'unicité de l'attribut

✗ **Q2** : Comment peut-on peut référencier une clé primaire composée dans la table père ?

**Réponse** : FOREIGN KEY (**Attr1**, **Attr2**) REFERENCES tablepère (**Attr1**, **Attr2**)

✗ **Q3** : Par quoi commencer dans un schéma cyclique (T1 référence T2 et T2 référence T1) ?

✗ **Q4** : Dans un schéma réflexive (Les 2 tables père et fils identiques : T qui référence T ) ?  
Comment et quand peut-on définir la CIR ?

# Gestion de Cohérence

## ■ *Définition des contraintes d'intégrité*

### ● **Ajout de la contrainte ultérieurement**

- ✗ Oubli de la CI soit dans la conception, soit dans la définition des schémas
- ✗ Nouvelle règle de gestion qui a surgit, non pris en charge dans la conception
- ✗ Schéma cyclique (T1 référence T2 et T2 qui référence T1)
- ✗ Schéma réflexive (T qui référence T )
- ✗ Contrainte Inter-Relations

**Rmq :** La contrainte ajoutée peut être contrainte de table ou de colonne



# Gestion de Cohérence

## ■ Définition des contraintes d'intégrité

- Ajout de la contrainte de table ultérieurement

✗ **ALTER TABLE** nom\_table **ADD** [**CONSTRAINT** nom\_contrainte] définition\_contrainte

**Ex** : **ALTER TABLE** Etudiant **ADD CONSTRAINT** Nom\_DN **UNIQUE**(Nom, Dn)

**ALTER TABLE** Employe **ADD CONSTRAINT** Sal\_Sup\_Prime **CHECK** (Sal > Prime)

**ALTER TABLE** Employe **ADD CONSTRAINT** Dir\_Ref **FOREIGN KEY** (NoChef) **REFERENCES**  
Employe (NEmp)

**ALTER TABLE** Employe **ADD FOREIGN KEY** (NoChef) **REFERENCES** Employe (NEmp)

✗ Clause « **CONSTRAINT** » Facilite l'identification, suppression, désactivation/activation de la contrainte

# Gestion de Cohérence

## ■ Définition des contraintes d'intégrité

- Ajout de la contrainte de colonne ultérieurement

✗ **ALTER TABLE** nom\_table {**ADD/MODIFY**} **COLUMN** ([nom\_colonne type [contrainte], ...])

**Ex :** ALTER TABLE Etudiant MODIFY COLUMN Note Number(4,2) CHECK (note ≤ 20)

```
ALTER TABLE Etudiant MODIFY COLUMN Adr VARCHAR (30)
CHECK (Adr IN ('Tlemcen', 'Ain Temouchent', 'Sidi Belabes'))
```

```
ALTER TABLE Employe ADD COLUMN Age Number(2) CHECK (Age>18 AND Age<65)
```

✗ **Rmq :** Si une contrainte est ajoutée à une colonne qui comporte des valeurs ne respectant pas la contrainte, le moteur de BD par défaut, retourne une erreur et n'ajoute pas la contrainte

# Gestion de Cohérence

## ■ *Définition des contraintes d'intégrité*

Les contraintes peuvent être déclarées sur 3 niveaux :

- **Contrainte Intra-Tables Horizontale**

- ✗ Contrôler la valeur d'un attribut en fonction des valeurs apparaissant dans les autres attributs de la même ligne

- **Contrainte Intra-Tables Verticale**

- ✗ Contrôler la valeur d'un attribut d'une ligne en fonction des valeurs de cet attribut pour les autres lignes

- **Contraintes Inter-Tables**

- ✗ Vérifier une condition entre 2 attributs de 2 tables T1 et T2

# Gestion de Cohérence

## ■ *Définition des contraintes d'intégrité*

### ● **Contrainte Intra-Relations Horizontale**

Contraintes entre attributs : « *Qte\_Stock ne doit jamais descendre sous la Qte\_Securite* »

```
ALTER TABLE Produit ADD CONSTRAINT Qte_stock_min  
CHECK (Qte_Stock > Qte_Securite)
```

Ou

```
CREATE TABLE Produit  
-----  
CHECK (Qte_Stock > Qte_Securite)
```

# Gestion de Cohérence

## ■ Définition des contraintes d'intégrité

### ● Contrainte Intra-Relations Verticale

**Exemple :** Employé ( Nom, Salaire, Nom-Service )

« Un employé ne peut gagner plus du double de la moyenne des salaires de son service »

```
ALTER TABLE Employe E1 ADD CONSTRAINT Salaire_CI  
CHECK (Salaire ≤ 2 * (SELECT AVG(Salaire) FROM Employe E2  
WHERE E1.Nom_Service = E2.Nom_Service))
```

Ou

```
CREATE TABLE Employe E1  
-----  
CHECK (Salaire ≤ 2 * (SELECT AVG(Salaire) FROM Employe E2  
WHERE E1.Nom_Service = E2.Nom_Service))
```

# Gestion de Cohérence

## ■ *Définition des contraintes d'intégrité*

### ● **Contrainte Inter-Relations**

✗ « *Qte\_Cmd ne doit jamais dépasser la Qte\_Stock* »

```
CREATE TABLE Comande
```

```
-----
```

```
CHECK (Qte_Cmd ≤ (SELECT Qte_Stock FROM Produit WHERE NoProd = Produit.NoProd))
```

✗ « *Tout service apparaissant dans la table Employé est décrit dans la table service* »

```
ALTER TABLE Employe ADD CONSTRAINT Serv_CI
```

```
CHECK (Nom_Service IN (SELECT Nom_Service FROM Service))
```

# Gestion de Cohérence

## ■ *Contrainte d'intégrité référentielle*

### ● **Recommandations**

- ✗ Les deux attributs dans les 2 tables père et fils doivent avoir le **même type** et la **même taille** (pas obligatoirement le même nom !)
- ✗ **Ordre de création** : Les tables pères doivent être créées en premier (de même pour les insertions de données)
- ✗ **Ordre de destruction** : Il faut détruire les tables dans l'ordre inverse de création (Les tables fils d'abord), afin de ne pas violer les contraintes de FOREIGN KEY (de même pour la suppression de données)

# Gestion de Cohérence

## ■ *Contrainte d'intégrité référentielle*

- Intérêt

- ✗ Interdire les anomalies de :

1. **Suppression** ( Si on supprime le client STAR de la table **père** Client, 5 commandes de la table fils Commande deviennent incohérentes, elles doivent donc être supprimées ou non ? )
2. **Modification** ( Si on modifie le code du produit PEUG308 de la table **père** Produit que l'on transforme de 225 à 325, les 3 enregistrements de la table fils Commande le concernant doivent être modifiés ou non ? )
3. **Ajout** ( Si on tente d'ajouter une commande dans la table **fils** Commande qui référence le produit 222, l'ajout est refusé puisque ce produit n'existe pas dans la table père produit ou non ? )



# Gestion de Cohérence

## ■ *Contrainte d'intégrité référentielle*

- **Différentes configurations**

✗ **Cette vérification peut se faire de 3 manières selon les souhaits de concepteur :**

1. **Refus de l'opération et simple signalement** d'une anomalie de présence. Dans ce cas un message apparaît et la mise à jour est refusée
2. **Effacement automatique** des lignes qui référencent un objet qui n'existe plus dans la table principale
3. **Mise à jour automatique** des lignes utilisant la clé étrangère qui référence une clé primaire venant de changer de valeur

# Gestion de Cohérence

## ■ *Contrainte d'intégrité référentielle*

- **Politiques de gestion de CIR**

- ✗ **2 Clauses**

- ON DELETE

- ON UPDATE

- ✗ **4 Options**

- NO ACTION (Par défaut)

- CASCADE

- SET NULL

- SET DEFAULT

- ✗ **Tentative de mise à jour d'un client dans la table père Client**

- DELETE FROM Client WHERE NoClient = 10

# Gestion de Cohérence

## ■ *Contrainte d'intégrité référentielle*

- Rejet d'une violation de la contrainte par défaut (NO ACTION)

Table Facture		
N°Facture	DateFacture	NoClient
1	21/04/2012	10
2	03/05/2012	20
3	20/06/2012	10
4	26/07/2012	10
5	14/08/2012	30
6	08/10/2013	20
7	14/10/2013	40
8	25/11/2013	40

✗ DELETE FROM Client WHERE NoClient = 10  
{Opération rejetée}

✗ DELETE FROM Client WHERE NoClient = 70  
{Opération acceptée}

# Gestion de Cohérence

## ■ *Contrainte d'intégrité référentielle*

### ● Politique de gestion de CIR

✗ `CONSTRAINT nom_contrainte FOREIGN KEY (attribut_clé_ét) REFERENCES tablepère (attribut) [ON DELETE | ON UPDATE { CASCADE | SET NULL | SET DEFAULT }]`

✗ Les clauses **ON DELETE** et **ON UPDATE** disposent de trois options :

- **CASCADE** applique la même opération à tous les enregistrements fils rattachés à l'enregistrement père
- **SET NULL** modifie leur clé étrangère à NULL
- **SET DEFAULT** remet leur clé étrangère à la valeur par défaut déclaré lors de la définition de la table

# Gestion de Cohérence

## ■ *Contrainte d'intégrité référentielle*

- **Suppression des lignes correspondantes (CASCADE)**

Table Facture		
N°Facture	DateFacture	NoClient
1	21/04/2012	10
2	03/05/2012	20
3	20/06/2012	10
4	26/07/2012	10
5	14/08/2012	30
6	08/10/2013	20
7	14/10/2013	40
8	25/11/2013	40

✗ DELETE FROM Client WHERE noClient = 10  
**{Opération acceptée}**

✗ Suppression automatique de toutes les lignes correspondantes dans la table fils Facture : 1, 3 et 4

# Gestion de Cohérence

## ■ *Contrainte d'intégrité référentielle*

- Remise à Null de la clé étrangère des lignes correspondantes (SET NULL)

Table Facture		
N°Facture	DateFacture	NoClient
1	21/04/2012	10
2	03/05/2012	20
3	20/06/2012	10
4	26/07/2012	10
5	14/08/2012	30
6	08/10/2013	20
7	14/10/2013	40
8	25/11/2013	40

✗ DELETE FROM Client WHERE noClient = 10  
**{Opération acceptée}**

✗ Remplacement automatique de toutes les lignes correspondantes dans la table fils  
Facture : 1, 3 et 4 par NULL

# Gestion de Cohérence

## ■ *Contrainte d'intégrité référentielle*

- Remise à la valeur par défaut de la FK des lignes correspondantes (SET DEFAULT)

Table Facture		
N°Facture	DateFacture	NoClient
1	21/04/2012	10
2	03/05/2012	20
3	20/06/2012	10
4	26/07/2012	10
5	14/08/2012	30
6	08/10/2013	20
7	14/10/2013	40
8	25/11/2013	40

✗ DELETE FROM Client WHERE noClient = 10  
**{Opération acceptée}**

✗ Remplacement automatique de toutes les lignes correspondantes dans la table fils Facture : 1, 3 et 4 par la val par défaut désignée lors de la définition

# Gestion de Cohérence

## ■ *Contrainte d'intégrité référentielle*

### ● Politique de gestion de contrainte d'intégrité référentielle

- ✗ Dans le cas d'une entité faible, on décide de détruire le composant quand on détruit le composé
- ✗ **Exemple** : quand on détruit un cinéma, on veut également détruire les salles ; quand on modifie la clé d'un cinéma, on veut répercuter la modification sur ses salles

```
CREATE TABLE Salle (nomCinema VARCHAR (30) NOT NULL,  
no INTEGER NOT NULL,  
capacite INTEGER,  
PRIMAR KEY (nomCinema, no),  
FOREIGN KEY (nomCinema) REFERENCES Cinema ON DELETE CASCADE ON UPDATE CASCADE)
```

- ✗ On ne peut pas spécifier ici ON DELETE SET NULL car nomCinema fait partie de la clé et ne peut donc pas être NULL
- ✗ La spécification des actions ON DELETE et ON UPDATE simplifie considérablement la gestion de la base par la suite (on n'a plus par exemple à se soucier de détruire les salles quand on détruit un cinéma)



# Gestion de Cohérence

## ■ *Contraintes d'intégrité statiques*

- **Modification de contrainte**

✗ **ALTER TABLE** <Nom\_table> **MODIFY CONSTRAINT** <Nom\_contrainte>  
<Nouv\_définition\_contrainte> [VALIDATE | NOVALIDATE]

### Exemples

ALTER TABLE Etudiant MODIFY CONSTRAINT Serv\_CI UNIQUE (Nom\_Service)

ALTER TABLE Employe MODIFY CONSTRAINT Sal\_Sup\_Prime CHECK (Sal = 2\*Prime)

✗ **VALIDATE** : Vérifiée sur les données existantes déjà dans la table

✗ **NOVALIDATE** : Vérifiée seulement sur les nouvelles insertions mais pas sur les données déjà existantes

# Gestion de Cohérence

## ■ *Contraintes d'intégrité statiques*

- **Renommage de contrainte**

```
x ALTER TABLE <Nom_table> RENAME CONSTRAINT <constraint_name> TO  
    <constraint_name>
```

### Exemple

```
ALTER TABE Employe RENAME CONSTRAINT Sal_Sup_Prime TO Sal_Double_Prime
```

# Gestion de Cohérence

## ■ *Contraintes d'intégrité statiques*

- **Suppression de contrainte**

- ✗ **ALTER TABLE** nom\_table **DROP CONSTRAINT** nom\_contrainte

- Exemples** : ALTER TABLE Etudiant DROP CONSTRAINT note\_max

- ALTER TABLE Employe DROP PRIMARY KEY, DROP CONSTRAINT emp\_fkey

- ALTER TABLE Employe DROP UNIQUE (nom, salaire)

- ALTER TABLE Employe DROP FOREIGN KEY Dir\_Ref

- ALTER TABLE Employe DROP CONSTRAINT Dir\_Ref

- ✗ ALTER TABLE Etudiant MODIFY Attribut *(Modification de la table)*

- ✗ DROP TABLE Employe CASCADE CONSTRAINTS

- (Supprime la table et toutes les contraintes associées directes ou indirectes des tables fils sans rien modifier aux données qui y sont stockées )*

# Gestion de Cohérence

## ■ *Contraintes d'intégrité statiques*

- **Q** : Problème de violation de clé lors d'insertion dans la 1<sup>ère</sup> table d'un schéma cyclique  
Comment le résoudre ?
- **Désactivation/Activation d'une contrainte**
  1. **ALTER TABLE** <Nom\_table> **DISABLE CONSTRAINT** <Nom\_contrainte>  
Ex : ALTER TABLE Etudiant DISABLE CONSTRAINT note\_max
  2. **Insertion des lignes** : INSERT INTO Etudiant (Note) VALUES ('21')
  3. **ALTER TABLE** <nom\_table> **ENABLE VALIDATE** | **ENABLE NOVALIDATE CONSTRAINT** <nom\_contrainte>  
ENABLE VALIDATE : Active la contrainte si l'ensemble des lignes déjà présentes respecte la CI  
ENABLE NOVALIDATE : Active la contrainte pour les mises à jour suivantes sans vérifier les données déjà présentes  
Ex : ALTER TABLE Etudiant ENABLE CONSTRAINT note\_max (par défaut c'est Validate)

# Gestion de Cohérence

## ■ *Contraintes d'intégrité statiques*

### ● **Récapitulatif**

✗ NOT NULL

✗ PRIMARY KEY

✗ UNIQUE

✗ DEFAULT

✗ CHECK

✗ FOREIGN KEY

- ON DELETE CASCADE

- ON DELETE SET NULL

- ON DELETE DEFAULT

- ON UPDATE CASCADE

- ON UPDATE SET NULL

- ON UPDATE DEFAULT

# Gestion de Cohérence

## ■ *Informations sur les contraintes*

- Le dictionnaire de données conserve les informations sur les contraintes définies sur la base
- On peut les retrouver dans les vues d'Oracle : CONSTRAINT\_DEFS, USER\_CONSTRAINTS, USERS\_CONS\_COLUMNS, USER\_CROSS\_REFS

✗ **Exemple** : `SELECT constraint_name, constraint_type, table_name, r_constraint_name  
FROM user_constraints`

- Dans la réponse de cette requête, on trouvera les abréviations: P pour primary, U pour unique, C pour check ou NOT NULL, R pour foreign key, et V pour le type de contraintes créés par WITH CHECK OPTION pour les vues