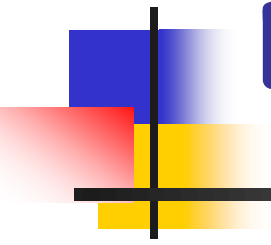
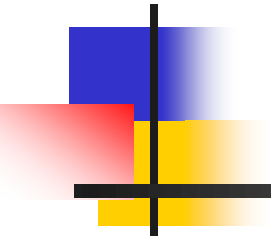


# Bases de données

## SQL - manipulation des données



# Sommaire

- 1. Rappels, introduction
- 2. Instructions de modification des données
- 3. Requêtes INSERT
- 4. Requêtes UPDATE
- 5. Requêtes DELETE
- 6. Notion de transaction

# 1.1 Rappel : les 5 langages de SQL

- **SQL = ensemble de langages**
  - DDL Data Definition Language
    - Création de tables, index, contraintes, …
    - **`CREATE, ALTER, DROP, …`**
  - DML Data Manipulation Language
    - Traitement des données : ajout, suppression, **interrogation**
    - **`INSERT, UPDATE, DELETE, SELECT`**
  - DCL Data Control Language
    - Gestion des droits d'accès aux tables
    - **`GRANT, REVOKE`**

# 1.1 Rappel : les 5 langages de SQL

- TCL Transaction Control Language
  - Contrôle de l'exécution des transactions
  - **`SET TRANSACTION, COMMIT, ROLLBACK`**

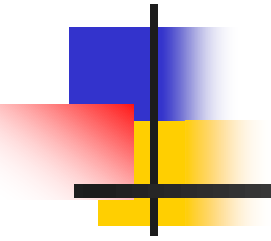
- SQL intégré **(embeddeb SQL)**
  - Permet l'intégration de SQL dans un langage procédural
    - PHP, JAVA, C…

# 1.2 Rappel : manipulation d'une BD

- **Commandes interactives**
  - requêtes envoyées à partir d'une « console »
- **À partir d'un programme d'application**
  - langage de programmation (ex : PHP, Java, C, …)
  - langage spécifique au SGBD (ex : PL/SQL d'Oracle, …)
- **Au travers d'une IHM**
  - ex : PHPMyAdmin pour MySQL

# 1.3 Rappel : objectifs du cours

- **Manipulation d'une BD dans ce cours**
  - Création de la BD et de sa structure
    - par le biais d'une interface (PHPMyAdmin)
    - TD 1 et 2
  - Interrogation d'une BD existante
    - par des requêtes : DML (`SELECT`)
    - TD 3 à 11
  - Modification des données d'une BD
    - par des requêtes : DML (`INSERT`, `UPDATE`, `DELETE`)
    - TD 12



# Sommaire

# 2.1 Instructions de modification

- **4 instructions de modification des données du DML**
  - **INSERT**
    - Ajout d'une ou plusieurs lignes dans une table
  - **UPDATE**
    - Modification d'une ou plusieurs valeurs d'attribut sur une ou plusieurs lignes dans une table
  - **DELETE**
    - Suppression d'une ou plusieurs lignes dans une table
  - **MERGE**
    - Réalise UPDATE si la ligne existe, INSERT sinon

# 2.2 Effet des instructions

- **Instruction `SELECT`**
  - ne modifie pas la BD
    - ni structure, ni données
  - produit une table à afficher

- **Instructions `INSERT`, `UPDATE`, `DELETE`**
  - sont ***susceptibles*** de modifier **les données**
    - mais peuvent les laisser inchangées
  - ne modifient pas la structure
  - ne produisent pas de table à afficher !

# 2.2 Effet des instructions

- **INSERT, UPDATE, DELETE**
  - ne peuvent agir que sur 1 table à la fois
- **INSERT, DELETE**
  - agissent sur 1 ou plusieurs lignes entières
- **UPDATE**
  - agit sur 1 ou plusieurs colonnes d'une ou plusieurs lignes
- **Nombre de lignes/cellules impactées**
  - impossible à déterminer a priori de façon simple
  - parfois répercussions dans d'autres tables (cascades)

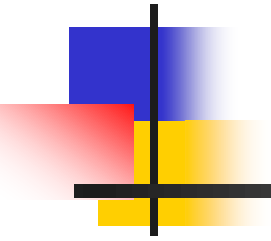
# 2.3 Précautions d'usage

- **Impossible de « revenir sur » un ordre de modif.**
  - pas de « `CTRL + Z` » pour « annuler »
  - parfois possible de « corriger à la main »
    - ex : rajouter des données supprimées
  - souvent difficile
    - ex : données supprimées en cascade – lesquelles ?
- **Précautions**
  - Vérifier les requêtes (les essayer sur une base de test ?)
  - Conserver des sauvegardes de la BD
  - Modifications dans une transaction (permet un ROLLBACK)

# 2.4 Cas d'erreur

- **Instruction `SELECT`**
  - fonctionne toujours (si syntaxe, typage, droits… corrects)
    - peut renvoyer un résultat vide (ce n'est pas une erreur !)

- **Instructions `INSERT`, `UPDATE`, `DELETE`**
  - cas d'erreur supplémentaires (malgré requête bien écrite)
    - non respect des contraintes d'intégrité référentielles
    - non respect des contraintes d'entité (unicité des clés primaires)
    - …



# Sommaire

# 3.1 Syntaxe d'une requête INSERT

**Notation** $[\ldots]_{0,n}$
$[a]_{0,n}$ signifie que a est facultatif, présent 0 à n fois

- **Syntaxe**

```
INSERT INTO <table> [<liste d'attributs>]0,1 VALUES
<liste de valeurs> [, <liste de valeurs>]0,n
```

- **Rôle**
  - Ajout d'une ou plusieurs lignes dans une table

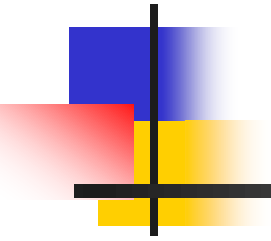
- **<table>**
  - une table existante de la BD

# 3.2 Liste d'attributs

- **Format de liste usuel**
  - attributs séparés par des virgules
  - le tout entre parenthèses
- **Attributs**
  - appartenant à la table
  - ceux pour lesquels des valeurs sont données
- **si liste absente (elle est facultative)**
  - des valeurs doivent être données pour tous les attributs
  - ordre des attributs de la machine (= ordre de création)

# 3.3 Liste de valeurs

- **Format de liste usuel**
  - valeurs séparées par des virgules
  - le tout entre parenthèses
- **Correspondance attributs/valeurs**
  - même nombre
  - types compatibles
  - dans l'ordre des listes
- **Si plusieurs listes**
  - ajout d'une ligne dans la table pour chaque liste



# 3.4 Valeurs

- **Chaque valeur peut être**
  - une constante
    - cas le plus fréquent
  - `DEFAULT`
    - utilisation de la valeur par défaut (doit avoir été définie)
    - (aussi le cas pour les attributs manquants dans la liste)
  - `NULL`
    - cette valeur doit avoir été autorisée pour l'attribut

# 3.5 Exemples

- **Insertion d'une ligne - attributs par défaut**

```
INSERT INTO artiste VALUES
('A001', 'EASTWOOD', 'Clint', 'USA', 1930, NULL, 'H')
```

- **Insertion de plusieurs lignes - attributs par défaut**

```
INSERT INTO artiste VALUES
('A002', 'KUBRICK', 'Stanley', 'USA', 1928, 1999, 'H'),
('A003', 'HITCHCOCK', 'Alfred', 'UK', 1899, 1980, 'H'),
('A005', 'LEONE', 'Sergio', 'ITALIE', 1921, 1989, 'H')
```

Noter les virgules de séparation

# 3.5 Exemples

- **Insertion d'une ligne - attributs choisis**
  - tous, ordre « normal »

```
INSERT INTO membre (MembreID, Mnom, Mprenom, Mnaissance, Msexe) VALUES
('M001', 'SABLON', 'Sabine', 1973, 'F')
```

  - tous, ordre personnalisé
    - correspondance dans l'ordre d'écriture !

```
INSERT INTO membre (Mnom, Mprenom , Msexe, MembreID, Mnaissance) VALUES
('WILSON', 'Yvon', 'H', 'M002', 1963)
```

# 3.6 Causes d'erreur possibles

- **Requête correcte…**
  - syntaxe correcte
  - valeurs cohérentes par rapport aux attributs

```
INSERT INTO film VALUES
('F001', 'Million dollar baby', 'A001', 'Drame', 'USA', 132, 2004, NULL)
```

- **… peut s'exécuter sans erreur**
- **… peut produire une erreur (= aucun ajout dans BD)**
  - clé primaire déjà utilisée
    - `'F001'` déjà dans la table film
  - valeur `NULL` sur attribut non NULL (le dernier ici)
  - clé étrangère qui n'existe pas en tant que clé primaire
    - `'A001'` absent de la table artiste
  - …

# 3.7 Exemple avec valeurs par défaut

- **Valeurs par défaut de la table artiste**
  - **Amort : NULL**
  - **Asexe : H**
  - **Anat : USA**

```
INSERT INTO artiste (ArtisteID, Anom, Aprenom, Anat, Anaissance) VALUES
('A230', 'Travolta', 'John', DEFAULT, 1954)
```

- Anat présent, avec valeur DEFAULT
  - utilisation de la valeur par défaut USA
- Amort et Asexe absents
  - utilisation des valeurs par défaut NULL et H

| ArtisteID | Anom | Aprenom | Anat | Anaissance | Amort | Asexe |
|---|---|---|---|---|---|---|
| A230 | Travolta | John | USA | 1954 | | H |

**rappel** : case vide dans l'interface = valeur **NULL**

# 3.8 Auto-incréments

- **Remarque : identifiants de cinéphile préfixés par l'initiale de la table A001, F023, M004…**
  - objectif pédagogique pour des débutants
    - permet de « voir » la table concernée
    - évite les erreurs de jointure (A001 ne *matche* pas F001)
  - mauvaise idée en pratique
    - chaîne de caractère : test d'égalité plus long
    - plus difficile de trouver le « prochain ID non utilisé »

# 3.8 Auto-incréments

- **Ajouter un artiste**
  - ID choisi en dur… problème de concurrence d'accès ?
    - et défini « à la main »…

```
INSERT INTO artiste (ArtisteID, Anom, Aprenom, Anat, Anaissance) VALUES
('A230', 'Travolta', 'John', DEFAULT, 1954)
```

  - ID calculé par sous-requête
    - en cas d'ID numérique

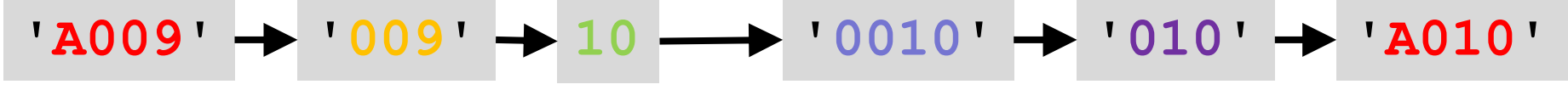
```
INSERT INTO artiste (ArtisteID, Anom, Aprenom, Anat, Anaissance) VALUES
((SELECT MAX(ArtisteID) + 1
  FROM artiste AS a),          'Travolta', 'John', DEFAULT, 1954)
```

    - mais pas si simple avec A001, A002…

# 3.8 Auto-incréments

- ID suivant calculé par sous-requête

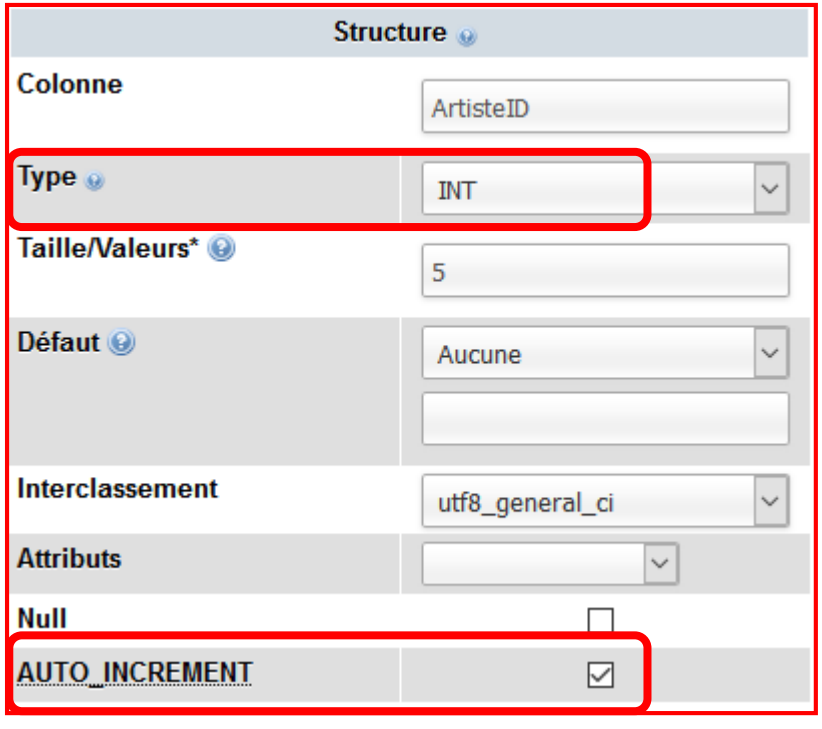
```
INSERT INTO artiste (ArtisteID, Anom, Aprenom, Anat, Anaissance) VALUES
((SELECT CONCAT('A',RIGHT(CONCAT('00', SUBSTRING(MAX(ArtisteID), 2, 3) + 1),3))
  FROM artiste),          'Travolta', 'John', DEFAULT, 1954)
```




- compliqué…
- verrou sur artiste entre la sous-requête et l'insertion ?

# 3.8 Auto-incréments

- **Meilleur identifiant : numérique, auto-incrémenté**
  - il suffit de ne pas préciser l'attribut !
  - le SGBD génère la valeur suivante




```
INSERT INTO artiste
   (Anom, Aprenom, Anat, Anaissance) VALUES
('Travolta', 'John', DEFAULT, 1954)
```

  - mais attention aux jointures !
    tous les ID deviennent compatibles…

# 3.9 Utilisation de sous-requêtes

- **SS-req pour obtenir des valeurs à ajouter**
  - Exemple : ajout du rôle de « John Travolta » dans le film « Pulp Fiction » (Vincent Vega )
    - identifiants de J. Travolta et de Pulp Fiction ?

```
INSERT INTO joue (JArtisteID, JFilmID, Jrole) VALUES
('A230', 'F320', 'Vincent Vega')
```

```
INSERT INTO joue (JArtisteID, JFilmID, Jrole) VALUES
((SELECT ArtisteID FROM artiste WHERE (Anom, Aprenom) = ('Travolta', 'John')),
 (SELECT FilmID    FROM film    WHERE Ftitre = 'Pulp Fiction'),
 'Vincent Vega')
```

# 3.9 Sous-requête sur la table à modifier

- **ne pas utiliser directement la table à modifier !**
  - Exemple : ajout du film « Dr Jekyll et Mr Hide » de G. Kikoïne (artiste `'A500'`), remake du film de 1944
    - utilisation de sous requête pour obtenir l'ID du film original

```
INSERT INTO film VALUES
('F350', 'Dr Jekyll et Mr Hide', 'A500', 'Horreur', 'UK', 87, 1989,
  (SELECT FilmID
   FROM film
   WHERE Ftitre = 'Dr Jekyll et Mr Hide'
     AND Fannee = 1944)
)
```

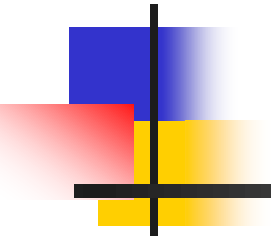
**MySQL a répondu:**

```
#1093 - You can't specify target table 'film' for update in FROM clause
```

# 3.9 Sous-requête sur la table à modifier

- **Il faut renommer la table de la sous-requête**
  - ➔ ce n'est plus la même table pour le SGBD !

```
INSERT INTO film VALUES
('F350', 'Dr Jekyll et Mr Hide', 'A500', 'Horreur', 'UK', 87, 1989,
  (SELECT FilmID
   FROM film AS film_original
   WHERE Ftitre = 'Dr Jekyll et Mr Hide'
     AND Fannee = 1944)
)
```



# Sommaire

# 4.1 Syntaxe d'une requête UPDATE

- **Syntaxe 1 : liste attribut=valeur**

```
UPDATE <table>
SET <attr> = <val> [, <attr> = <val>]0,n
[WHERE <condition>]0,1
```

- **Syntaxe 2 : n-uplet**

```
UPDATE <table>
SET (<attr1>, <attr2>, …) = (<val1>, <val2>, …)
[WHERE <condition>]0,1
```

- **Rôle**
  - modif. un ou plusieurs attr. d'une ou plusieurs lignes d'une table

# 4.1 Syntaxe d'une requête UPDATE

```
 UPDATE <table>
 SET <attr> = <val> [, <attr> = <val>]0,n
[WHERE <condition>]0,1
```

- **<table>**
  - une table existante de la BD
- **<attr> = <val>**
  - nouvelle valeur pour l'attribut de la table
  - valeur peut être obtenue par ss-req.
- **<condition>**
  - modif. faite sur les lignes pour lesquelles la cond. est VRAIE
  - sans clause **WHERE** : toutes les lignes sont modifiées

# 4.2 Exemples

- **Modification d'une date de décès d'un artiste**
  - Eli Wallach est décédé en 2014

```
UPDATE artiste
SET Amort = 2014
WHERE (Anom, Aprenom) = ('Wallach', 'Eli')
```

- **Modifier toutes les notes**
  - diviser par 4 pour avoir une évaluation sur 5 (cf. étoiles…)

```
UPDATE note
SET Nnote = TRUNC(Nnote / 4)
```

  - pas de clause **WHERE** : toutes les lignes sont modifiées

# 4.3 Utilisation de sous-requêtes

- **Deux cas d'utilisation**
  - pour obtenir la nouvelle valeur d'un attribut
    - requêtes qui renvoient 1 colonne et 1 ligne
  - pour l'écriture de la condition du WHERE
    - cf. cours sur les sous-requêtes
- **Même précaution sur la table modifiée**
  - si table modifiée utilisée dans une sous-requête il faut la renommer dans la sous-requête

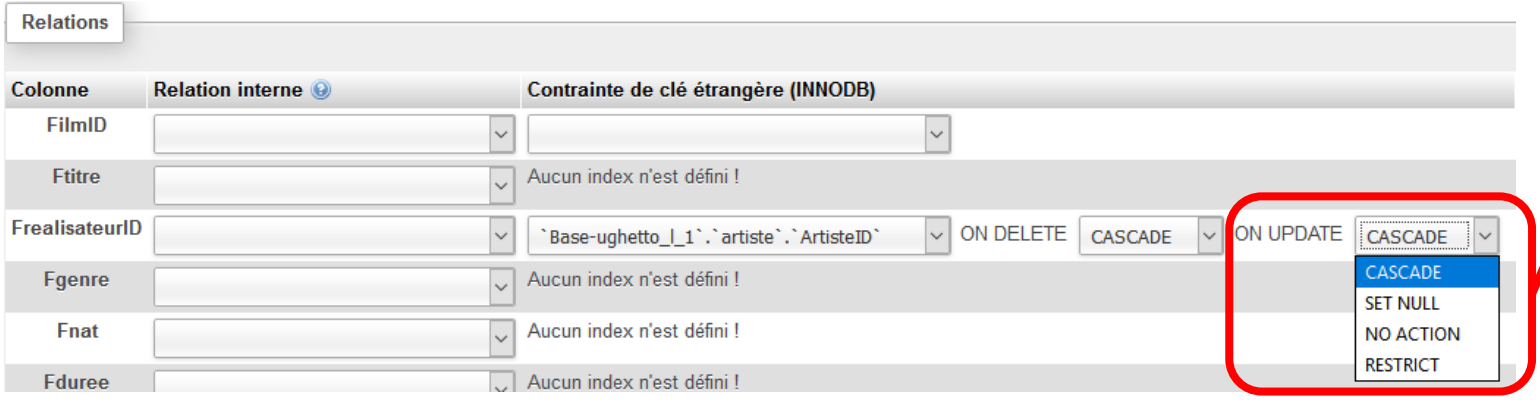
# 4.4 Modification d'une clé primaire

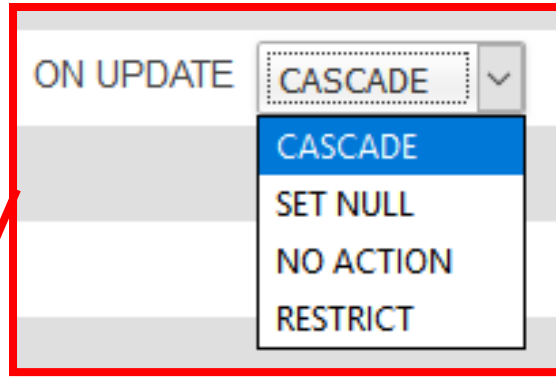
- **Modifier la valeur d'une clé primaire…**
  **… peut avoir des répercutions**
  **sur une clé étrangère qui y fait référence !**

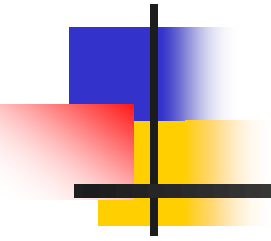
- **Comportement du SGBD**
  - défini lors de la déclaration des CIR
    - CIR = Contrainte d'Intégrité Référentielle
  - (si CIR pas déclarées, c'est à l'utilisateur de faire attention…)
  - défini par la directive **`ON UPDATE`** liée à la CIR

# 4.4 Modification d'une clé primaire

- **4 possibilités pour `ON UPDATE`**
  - `RESTRICT` : modification non autorisée
    - donc refusée dans tous les cas !
  - `NO ACTION` : SGBD ne fait rien de particulier
    - refuse la modification si viole une CIR
  - `SET NULL` : clé étrangère devient NULL
    - modification refusée si clé étrangère « non NULL »
  - `CASCADE` : modifie aussi les valeurs des clés étrangères
    - dans les tables contenant les clés étrangères





# Sommaire

# 5.1 Syntaxe d'une requête DELETE

- **Syntaxe**

```
DELETE FROM <table>
[WHERE <condition>]0,1
```

- **Rôle**
  - supprime une ou plusieurs lignes dans une table
- **<table>**
  - une table existante de la BD
- **<condition>**
  - suppression des lignes pour lesquelles la condition est VRAIE
  - sans clause **WHERE** : toutes les lignes sont supprimées !

# 5.2 Exemples

- **Vider une table**
  - supprimer toutes les notes

```
DELETE FROM note
```

  - pas de clause **WHERE** : toutes les lignes sont supprimées

- **Supprimer des lignes choisies**
  - Supprimer les membres mineurs du club

```
DELETE FROM membre
WHERE YEAR(CURRENT_DATE()) - Mnaissance < 18
```

# 5.3 Utilisation de sous-requêtes

- **Pour l'écriture de la condition du WHERE**
  - cf. cours sur les sous-requêtes
- **Même précaution sur la table modifiée**
  - si table modifiée utilisée dans une sous-requête il faut la renommer dans la sous-requête

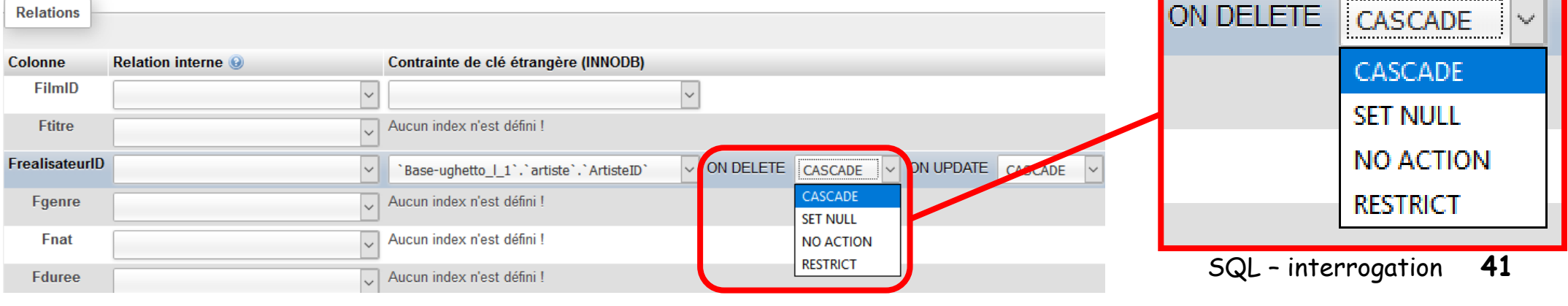
# 5.4 Suppression d'une clé primaire

- **Supprimer la valeur d'une clé primaire…**
  **… peut avoir des répercutions**
  **sur une clé étrangère qui y fait référence !**

- **Comportement du SGBD**
  - défini lors de la déclaration des CIR
    - CIR = Contrainte d'Intégrité Référentielle
  - (si CIR pas déclarées, c'est à l'utilisateur de faire attention…)
  - défini par la directive **`ON DELETE`** liée à la CIR

# 5.4 Suppression d'une clé primaire

- **4 possibilités pour** `ON DELETE`
  - `RESTRICT` : suppression non autorisée
    - donc refusée dans tous les cas !
  - `NO ACTION` : SGBD ne fait rien de particulier
    - refuse la suppression si viole une CIR
  - `SET NULL` : clé étrangère devient NULL
    - suppression refusée si clé étrangère « non NULL »
  - `CASCADE` : supprime aussi les **lignes** des clés étrangères
    - dans les tables contenant les clés étrangères, **en cascade !**

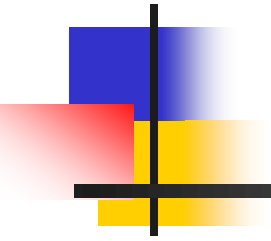


# 5.4 Suppression d'une clé primaire

- **Exemple : supprimer François Truffaut**
  - CIR : **CASCADE** sauf FremakeDe/FilmID : **SET NULL**

```
DELETE FROM artiste
WHERE (Aprenom, Anom) = ('François', 'Truffaut')
```

  - Supprime
    - F. Truffaut de `artiste`
    - les films réalisés par F. Truffaut de `film`
    - les notes mises aux films de F. Truffaut de `note`
    - les rôles joués dans les films de F. Truffaut de `joue`
    - les rôles joués *par* F. Truffaut de `joue`
  - Met `NULL` dans la colonne `FRemakeDe` (de `film`)
    - aux films de F. Truffaut dont on a fait un remake

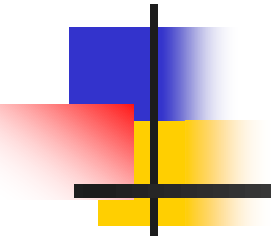


# Sommaire

# 6.1 Transaction

- **Définition : *ensemble* de modifications d'une BD**

- **Intérêt**
  - modifications simultanées / retour sur panne système
    - lorsqu'elles doivent être effectuées en « tout ou rien »
    - ex : modification de réservation, transfert bancaire…
  - assurer la cohérence de la BD
    - lorsqu'une modif. isolée serait rejetée pour incohérence
    - ex : modif. de l'ID d'un artiste (avec ON UPDATE NO ACTION)



# Questions ?