

## Introduction aux Bases de Données et au langage SQL

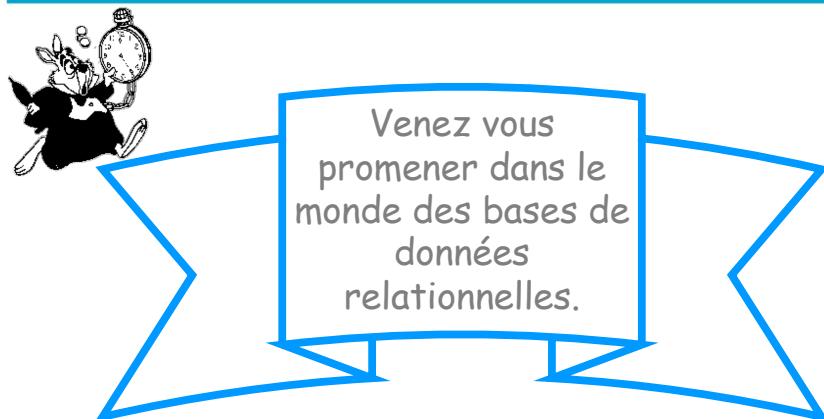
3IRC  
2018/2019

F. PERRIN

membre de UNIVERSITÉ DE LYON



## I have a dream today (2)



## I have a dream today (1)



Ne serait-ce pas merveilleux s'il existait un monde dans lequel je saurais à qui j'ai prêté mon DVD préféré, à quelle heure part le train demain, ..., sans effectuer une fouille archéologique dans tous les post-it collés sur mon bureau ? Mais ce n'est probablement qu'un rêve...

## Qu'est-ce qu'une Base de Données (1) ?



Chaque fois que vous faites une recherche sur Internet, recevez une amende pour excès de vitesse, etc., une **base de données** reçoit une demande d'informations, que l'on appelle une **requête**.



## Qu'est-ce qu'une Base de Données (2) ?

- Une **base de données** est un récipient qui contient des **tables** et d'autres structures SQL liées à ces tables.
- Bases de données est une collection de données :
  - **Non redondante**.
  - Ayant souvent des **liens** entre elles.
  - **Cohérentes** (respectent des contraintes d'intégrité).
  - **Partagées** par plusieurs utilisateurs simultanément.

## I have a dream today (1)



Ne serait-ce pas merveilleux s'il existait un moyen d'organiser les données et les liens qu'elles tissent entre elles de manière à les retrouver rapidement ?  
Mais ce n'est probablement qu'un rêve...

## Comment est gérée une BD ?

### ▪ SYSTÈME DE GESTION DE BASES DE DONNÉES

- Permet description, création de la base, l'utilisation et la mise à jour des données.
- Contrôle cohérence, non redondance, sécurité.



## I have a dream today (2)



Découvrez les secrets des tables et de leurs lignes et colonnes.



## Exemple : BD Cinemas

- Soit une BD qui gère la programmation de film dans des salles de cinéma ainsi que les caractéristiques des films, acteurs et réalisateurs.
- Les utilisateurs peuvent faire des requêtes pour connaître les films programmés à 20h au 'Pathé', mais aussi la liste des films 'fantastique', réalisés par 'Spielberg', etc.
- Elle est composée des tables FILM, PERSONNE, GENRE, DISTRIBUTION, CINEMA , SALLE , PROGRAMMATION dont la description et les liens sont modélisés page suivante.

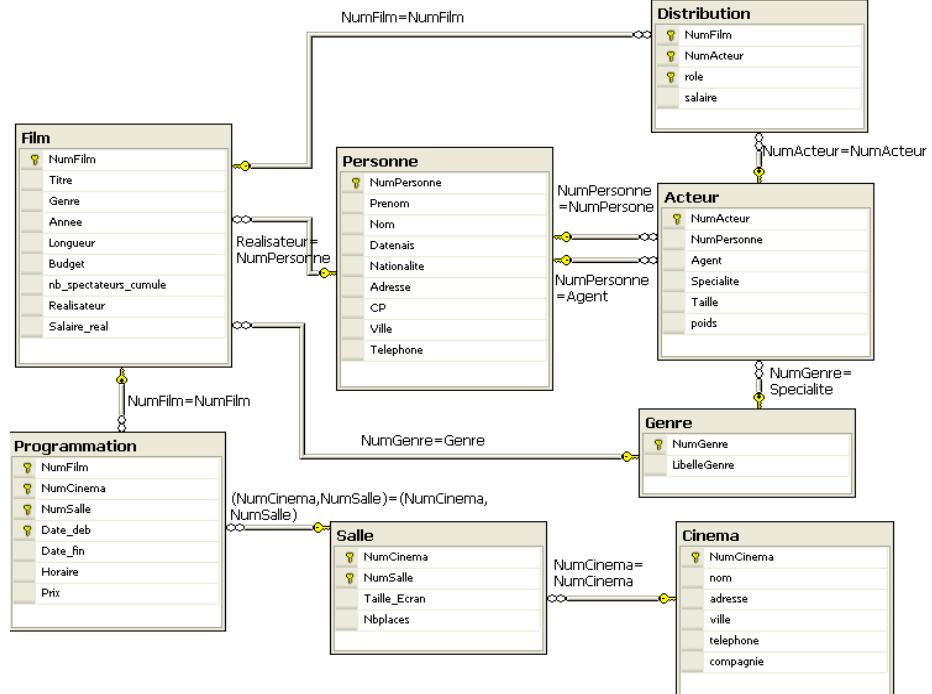


## Qu'est-ce qu'une table ?

- Une **table** est une structure qui contient des données, organisées en **colonnes** et en **lignes** (PERSONNE, etc.).
- Une **colonne** est une donnée conservée par votre table (Nom, Nationalité, etc.).
- Une **ligne** est un ensemble unique de colonnes qui décrit les attributs d'un objet particulier.



	NumPersonne	Prenom	Nom	Datenaïs	Nationalité	Adresse	CP	Ville	Telephone
1	1	Steven	Spielberg	1947	Americain	5é Avenue	NULL	New York	555111



## Certaines colonnes ont une vocation particulière

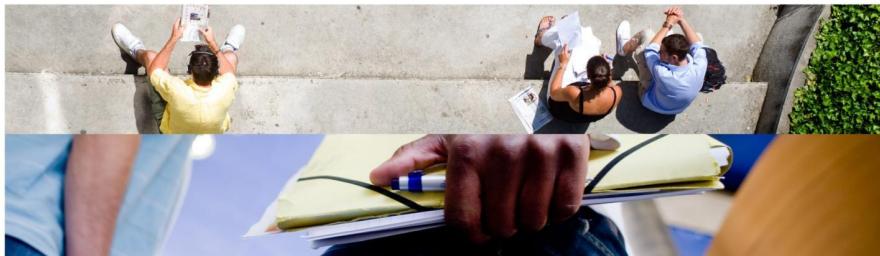
- Une **clé primaire** :
  - Est une colonne de votre table qui rend chaque enregistrement **unique** (NumPersonne mais aussi NumFilm+NumActeur+Role, etc.).
  - Sa valeur permet **d'identifier**, de manière unique, les autres attributs d'une ligne (t-uplet) de la table.
- Une **clé étrangère** :
  - Est une colonne de la table qui référence la **CLÉ PRIMAIRE** d'une autre table.
  - Elle permet de faire le lien entre les 2 tables.

## A quoi ressemble une table pleine de liens ?

NumActeur	NumPersonne	Agent	Specialite	Taille	poids
1	5 [->]	6 [->]	1 [->]	153	52
2	3 [->]	6 [->]	1 [->]	147	48
3	10 [->]	11 [->]	2 [->]	165	63
4	12 [->]	19 [->]	2 [->]	157	68
5	13 [->]	19 [->]	2 [->]	185	78
6	14 [->]	11 [->]	1 [->]	175	72
7	15 [->]	11 [->]	1 [->]	165	50
8	16 [->]	11 [->]	6 [->]	188	78
9	17 [->]	11 [->]	1 [->]	180	75
10	18 [->]	11 [->]	2 [->]	172	60

12

© CPE Lyon - Françoise PERRIN - 2018-2019



## SQL Langage de Manipulation de Données

3IRC  
2018/2019

F. PERRIN

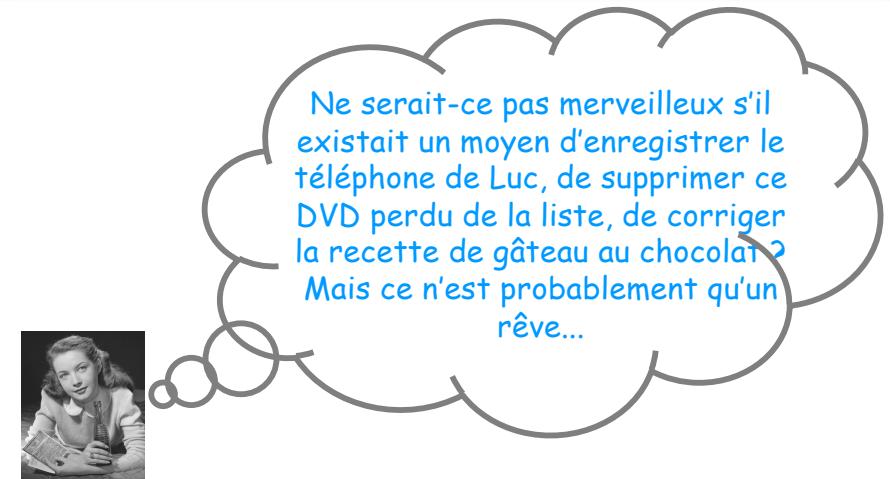
## Schéma d'une BD

- Une description des données (les colonnes et les tables) de votre base de données, ainsi que de tout autre objet en dépendant et de la façon dont ils sont tous liés entre eux s'appelle un **schéma**.

13

© CPE Lyon - Françoise PERRIN - 2018-2019

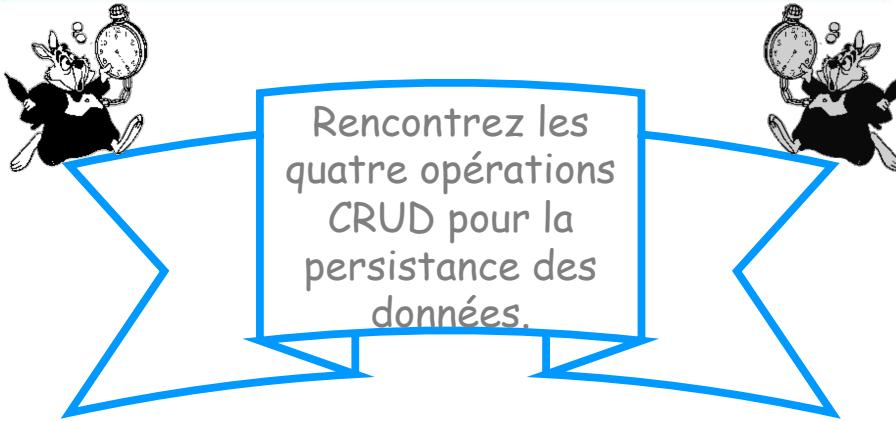
## I have a dream today (1)



© CPE Lyon - Françoise PERRIN - 2018-2019

15

## I have a dream today (2)



## Pour ajouter des lignes dans 1 table, utilisez l'instruction INSERT

- Syntaxe type :
  - **INSERT INTO** votre\_table (nom\_col1, nom\_col2,... )  
**VALUES** ( 'valeur1', 'valeur2',... );
- Exemple :
  - **INSERT INTO** Cinema **VALUES** (3,'Pathé','rue Machin','Bordeaux','0545327733','Gaumont');
  - Si vous ne les précisez pas, les valeurs doivent être dans le même ordre que les noms de colonnes.
- Les valeurs insérées peuvent être codées en dur, saisies dans le formulaire d'une page Web (Cf. cours PhP), venir d'une autre table/fichier, etc.

## CRUD, vous avez dit CRUD ?

Opération	SQL	HTTP
Create	<b>INSERT</b>	<b>POST</b> (en)
Read (Retrieve)	<b>SELECT</b>	<b>GET</b> (en)
Update (Modify)	<b>UPDATE</b> (en)	<b>PUT</b> (en) / <b>PATCH</b> (en)
Delete (Destroy)	<b>DELETE</b> (en)	<b>DELETE</b> (en)

## Comment vérifier que votre ligne est bien ajoutée ?

- **SELECT** est utilisé pour obtenir des enregistrements venant d'une ou plusieurs tables.
- **SELECT \***  
**FROM** Cinema ;
- **SELECT Budget**  
**FROM** Film  
**WHERE** Titre = 'Jurassic Parc' ;

Résultat attendu : \* est un joker qui représente toutes les colonnes.

Tables utilisées pour évaluer la requête.

Restriction sur 1 seule ligne.

## Pour supprimer lignes dans 1 table, utilisez l'instruction DELETE

- Syntaxe type :
  - `DELETE FROM votre_table WHERE condition ;`
- Exemple :
  - `DELETE FROM Genre WHERE LibelleGenre = 'Fantastique' ;`
  - `DELETE FROM Cinema WHERE compagnie IS NULL ;` Supprime plusieurs lignes à la fois.
- Si la table devient vide après l'action de DELETE, elle n'est pas supprimée.

Attention, pas toujours possible si ce genre est référencé dans une autre table (e.g. Film) !

## I have a dream today (1)



Ne serait-ce pas merveilleux s'il existait un moyen de trouver à quelle heure aller voir 'Le majordome', en VO, dans un ciné à côté d'une pizzeria, dans une salle pas trop grande et climatisée, entre 20h et 22h ?  
Mais ce n'est probablement qu'un rêve...

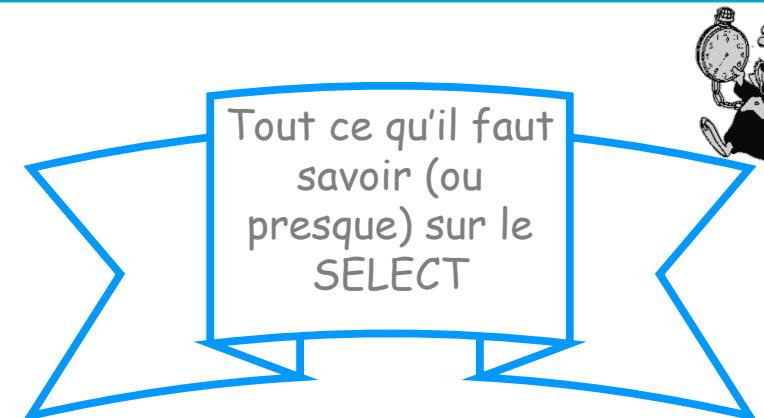
## Comment modifier les valeurs d'une ligne ?

- 1<sup>ère</sup> solution : INSERT avec les nouvelles valeur + DELETE de l'ancienne ligne. Bof ☺
- 2<sup>ème</sup> solution : utilisez l'instruction UPDATE.
  - `UPDATE Film SET nb_spectateurs_cumule = 5 000 000 WHERE Titre = 'Manhattan' ;`
  - `UPDATE Programmation SET Prix = 10.5 WHERE Prix = 10 ;` Colonne modifiée. Modifie plusieurs lignes à la fois.

## I have a dream today (2)



Tout ce qu'il faut savoir (ou presque) sur le SELECT





Extrait de SQL -Tête la Première, Lynn Beighley, 2007, O'Reilly

24

## Les 2 mots clés utiles pour améliorer la présentation



- **ORDER** : permet l'ordonnancement du résultat

▪ `SELECT Titre, Budget  
FROM Film  
ORDER BY Budget DESC, Titre ASC ;`



- **DISTINCT** : permet d'éliminer les doublons

▪ `SELECT DISTINCT compagnie FROM Cinema;`

## Comment se présente le résultat d'une requête SELECT ?



- Présentation résultat sous forme tabulaire :
  - En-tête colonne est le nom de l'attribut.
  - Largeur colonne est celle définie lors création table.
- Projection complète de toutes les colonnes, de toutes les lignes de votre table :
  - `SELECT *  
FROM votreTable ;`
- La requête sera plus rapide si vous sélectionnez seulement les champs nécessaires.
  - `SELECT col1, col2  
FROM votreTable ;`

## Comment sélectionnez les bonnes lignes selon 1ou plusieurs critères ?



- Opérateurs de sélection :
  - = != ou <> > >= < <=
- Opérateurs logiques dans prédictats composés :
  - and, or, not
- `SELECT titre, budget,  
'Film à petit budget' AS 'Type'  
FROM Film  
WHERE budget<=1000000 AND Annee = 1992`

Alias de colonne

## Et encore des mots clés pour indiquer les critères de restriction



- **BETWEEN** : teste l'appartenance d'une valeur à un intervalle :  
SELECT \* FROM Film  
WHERE Budget **BETWEEN** 1000000 AND 1500000
- **IN** : teste l'appartenance d'une valeur à liste de valeurs :  
SELECT \* FROM Genre WHERE LibelleGenre **IN** ('Drame', 'Comedie')
- **LIKE** : permet une recherche approximative  
SELECT \* FROM Personne WHERE CP **LIKE** '69%'  
SELECT \* FROM Personne WHERE Nom **LIKE** '\_\_\_\_\_'
- **IS NULL** : permet de tester si un champ a été affecté  
SELECT \* FROM Personne WHERE CP **IS NULL**

## ... en regroupant vos lignes



- Utilisez **GROUP BY** pour appliquer les fonctions d'agrégat à des lignes reliés sémantiquement
  - SELECT compagnie, **COUNT(\*)**  
FROM Cinema  
**GROUP BY** compagnie
- Utilisez **HAVING** pour filtrer le résultat d'une fonction d'agrégat :
  - SELECT compagnie, COUNT(\*)  
FROM Cinema  
**GROUP BY** compagnie  
**HAVING** COUNT(\*) > 2



## Profitez de la puissance des fonctions d'agrégats ...



- Utilisez **COUNT()** pour compter le nombre de lignes retournées par la requête :
  - SELECT **COUNT(\*)**  
FROM Distribution  
WHERE role = 'François PIGNON'
- Utilisez **SUM(), AVG(), MIN(), MAX()** pour vos opérations statistiques :
  - SELECT **AVG(Budget)**  
FROM Film
- Toutes ces fonctions ne retournent qu'1 seule valeur.

## Bienvenue dans un monde multi-tables



- Une jointure interne (**INNER JOIN**) combine les enregistrements de deux tables en utilisant des opérateurs de comparaison dans une condition.
- Les colonnes ne sont renvoyées que là où les lignes jointes correspondent à la condition.
  - SELECT F.titre, F.annee, P.nom as 'Nom réalisateur'  
FROM Film F  
**JOIN** Personne P **ON** F.Realisateur = P.NumPersonne



## Comment nettoyer de vieilles données isolées ?

- Pour retrouver des données d'une table qui ne sont pas encore ou plus liées à d'autres tables (par une FK) utilisez **une jointure externe**.
- Une **jointure externe gauche** prend toutes les lignes de la table de gauche et vérifie s'il y a correspondance dans les lignes de la table de droite :

Table de gauche      SELECT G.LibelleGenre, F.Titre  
                        FROM Genre G      Table de droite  
                        LEFT JOIN Film F ON G.NumGenre=F.Genre

## Plus fort encore : des requêtes imbriquées

- Lorsque vous devez prendre le résultat d'une requête pour l'utiliser dans la formulation d'une autre requête, utilisez une **sous-requête**.
  - SELECT nom  
FROM Personne  
WHERE Datenaïs = (SELECT MIN(Datenaïs)  
FROM Personne  
)
- La **requête interne** est d'abord exécutée puis son résultat est récupérée pour l'exécution de la **requête externe**.

## Quel résultat produit une jointure externe ?

- Une valeur **NULL** dans les résultats d'une jointure externe gauche signifie que **la table de droite n'a aucune valeur qui corresponde à la table de gauche**.
- Respectivement à droite :  
**RIGHT [OUTER] JOIN**

## Parfois le résultat de la sous-requête est un ensemble de valeurs

- IN** : permet de tester la présence (**NOT IN** l'absence) d'une valeur particulière dans un ensemble :
  - SELECT nom  
FROM Personne  
WHERE NumPersonne IN (SELECT NumActeur  
FROM Distribution  
WHERE salaire > 15000)



## Quels comportement attendre des requêtes imbriquées ?

- **NOT EXISTS** : permet de trouver toutes les lignes de la requête externe pour lesquelles il n'existe aucune ligne dans une table liée :
  - Rechercher toutes les Personnes qui ne sont pas réalisateurs dans la table Film.
- **EXISTS** : retourne VRAI si sous-requête retourne au moins 1 ligne :
  - Rechercher toutes les Personnes qui sont réalisateurs dans la table Film.



## SQL Langage de Définition de Données

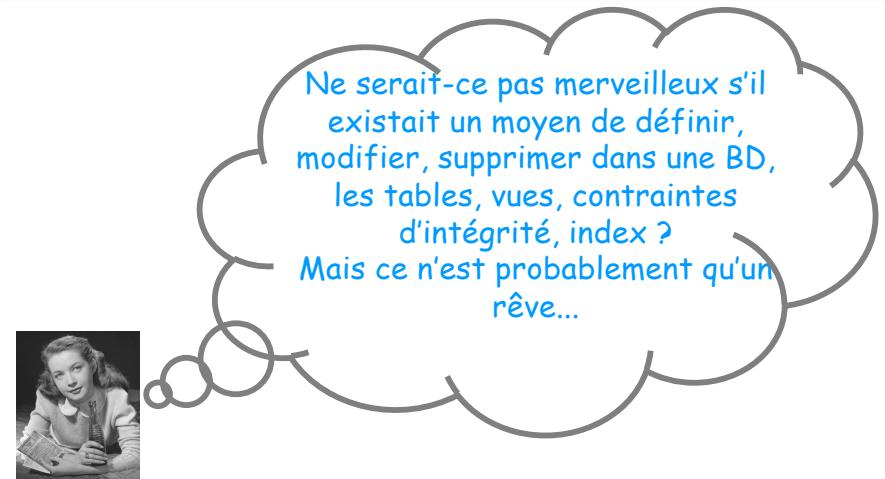
3IRC  
2018/2019

F. PERRIN

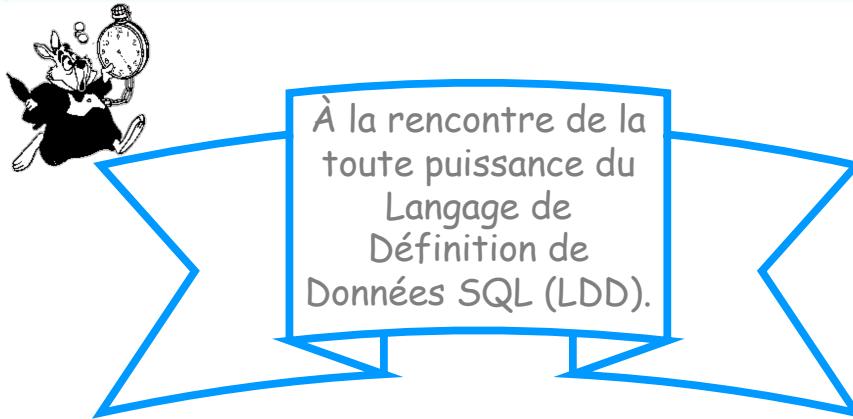
## Quels comportement attendre des requêtes imbriquées ?

- **ALL** : compare chacune des valeurs de l'ensemble retourné à une valeur particulière et retourne VRAI si la comparaison est évaluée pour chacun des éléments :
  - Trouver les acteurs qui ont gagné plus (en tout) que tous les réalisateurs (pour tous les films réalisés).
- **ANY** : idem ALL sauf qu'il suffit d'1 comparaison vraie :
  - Trouver les acteurs qui ont gagné plus (en tout) que la somme des salaires d'un réalisateur quelconque.

## I have a dream today (1)



## I have a dream today (2)



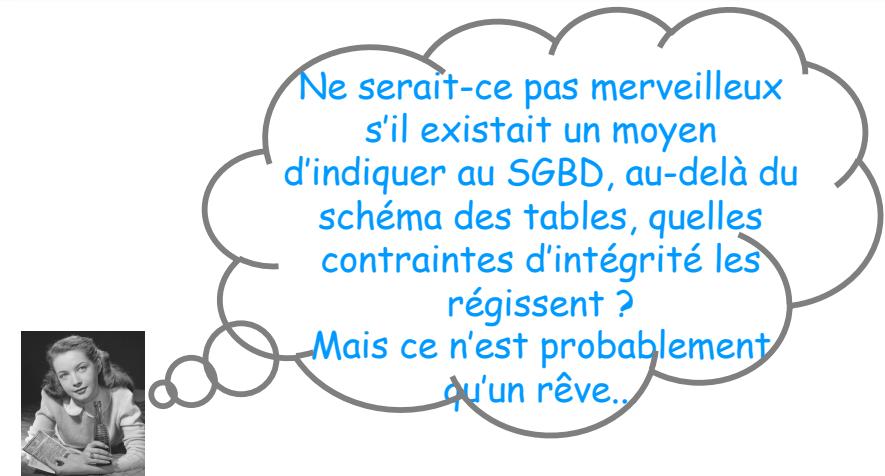
## A quoi sert le LDD ?

- Le SQL LDD vous permet de créer ou de modifier les structures de vos BD :
  - Tables et leurs contraintes d'intégrités (existence, unicité, domaine, référence).
  - Vues externes : pour limiter l'accès aux données en fonction du type d'utilisateur.
  - Index pour accélérer l'accès au données en permettant un accès direct et non séquentiel.

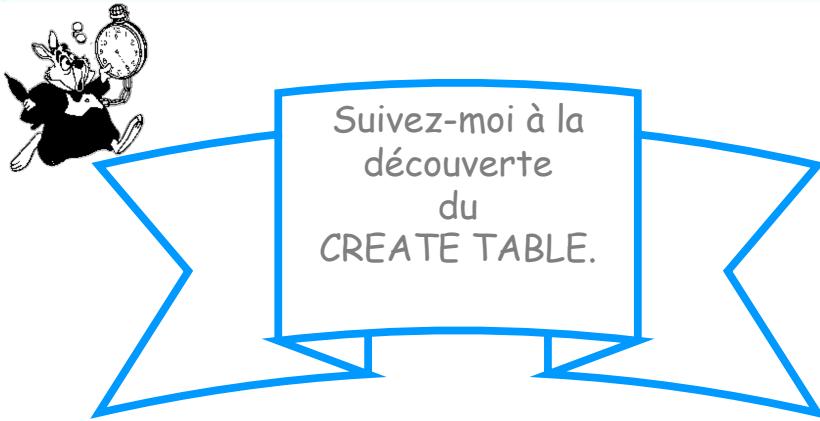
## Comment le SGBD permet-il de respecter les contraintes d'intégrité ?

- Lors de chaque accès en mise à jour (ajout, modification, suppression), le SGBD doit vérifier les contraintes d'intégrité et rejeter les requêtes si elles ne sont pas respectées.
- Elles sont en général définies lors de la création des tables, en donnant des précisions sur les attributs concernés.
- Certaines font l'objet de procédures particulières appelées triggers (déclencheur) qui sont exécutées lors de l'accès en MAJ aux données (CF. chapitre PL/SQL).

## I have a dream today (1)



## I have a dream today (2)

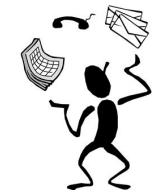


## Être NULL ou NOT NULL telle est la question...

- Certaines colonnes de votre table doivent **toujours avoir des valeurs**. 
- Vous pouvez facilement configurer votre table pour qu'elle n'accepte pas de valeurs NULL pour ses colonnes :
  - Prenom VARCHAR(30) **NOT NULL**,
- Une valeur NULL est une valeur **indéfinie**. Elle n'est pas égale à zéro ou à une valeur vide. Une colonne avec une valeur NULL **est NULL**, mais **n'est pas égale à NULL**.

## Comment créer une table ?

```
DROP TABLE IF EXISTS Personne;  
CREATE TABLE Personne(  
    NumPersonne INTEGER NOT NULL PRIMARY KEY,  
    Prenom VARCHAR(30) NOT NULL,  
    Nom VARCHAR(30) NOT NULL,  
    Datenaiss INTEGER,  
    Nationalite VARCHAR(30) NOT NULL,  
    Adresse VARCHAR(50),  
    CP CHAR(5),  
    Ville VARCHAR(30),  
    Telephone CHAR(20)  
)
```



## Qu'est-ce qu'une clé primaire ?

- Une clé primaire est une colonne de votre table qui rend chaque enregistrement unique :
  - NumPersonne INTEGER NOT NULL PRIMARY KEY.
- Une clé primaire **ne peut pas ne pas** être renseignée (NULL) : si c'était possible, elle ne pourrait pas être unique.
- L'identifiant d'une table pourrait être une clé concaténée composée de plusieurs attributs :
  - PRIMARY KEY(NumFilm, NumActeur, role),
- Cependant, une clé primaire doit être compacte : la plupart du temps un Id auto incrémenté est créé.
  - IdDistribution integer NOT NULL AUTO\_INCREMENT PRIMARY KEY

## Comment vérifier l'unicité d'une Ex clé primaire remplacée par un autoIncrément ?

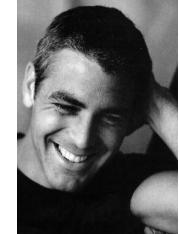
```
DROP TABLE IF EXISTS Distribution;  
CREATE TABLE Distribution(  
    IdDistribution integer  
        NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    NumFilm INTEGER ,  
    NumActeur INTEGER ,  
    role VARCHAR(50),  
    salaire INTEGER ,  
    UNIQUE (NumFilm, NumActeur, role),  
);
```

## A quoi ressemble une table pleine de liens ?

NumActeur	NumPersonne	Agent	Specialite	Taille	poids
1	5 [->]	6 [->]	1 [->]	153	52
2	3 [->]	6 [->]	1 [->]	147	48
3	10 [->]	11 [->]	2 [->]	165	63
4	12 [->]	19 [->]	2 [->]	157	68
5	13 [->]	19 [->]	2 [->]	185	78
6	14 [->]	11 [->]	1 [->]	175	72
7	15 [->]	11 [->]	1 [->]	165	50
8	16 [->]	11 [->]	6 [->]	188	78
9	17 [->]	11 [->]	1 [->]	180	75
10	18 [->]	11 [->]	2 [->]	172	60

## Comment créer 1 table dont certains attributs référencent une autre table ?

```
DROP TABLE IF EXISTS Acteur ;  
CREATE TABLE Acteur (  
    NumActeur INTEGER NOT NULL PRIMARY KEY,  
    NumPersonne INTEGER NOT NULL,  
    Agent INTEGER,  
    Specialite INTEGER,  
    Taille INTEGER,  
    poids INTEGER,  
    FOREIGN KEY (NumPersonne) REFERENCES Personne ,  
    FOREIGN KEY (Agent) REFERENCES Personne (NumPersonne),  
    FOREIGN KEY (Specialite) REFERENCES Genre (NumGenre)  
);
```



## Qu'est-ce qu'une clé étrangère (FOREIGN KEY ) ?

- La CLÉ ÉTRANGÈRE est une colonne de la table qui référence la CLÉ PRIMAIRE d'une autre table.
  - FOREIGN KEY (Agent) REFERENCES Personne (NumPersonne)
- Il est impossible de modifier une clé primaire tant qu'elle est référencée dans une autre table.
- Mais peut-on supprimer un t-uplet dont la clé primaire serait référencée dans une autre table ?
  - Ce n'est pas recommandé et c'est l'option par défaut : Dès que des notes sont rattachées à un étudiant, il ne peut plus être supprimé de la BD même s'il démissionne.

## Peut-on supprimer un t-uplet dont la clé primaire serait référencée dans une autre table ?

- Cependant, cela peut parfois être pertinent :
  - Si un vol est annulé (t-uplet supprimé avec un **DELETE**) toutes les réservations doivent être annulées.
  - Soit on autorise la destruction des t-uplets de la table **Reservation** :
    - **FOREIGN KEY** (**NumVol**) **REFERENCES** **Vol** (**IdVol**)  
**ON DELETE CASCADE**,
  - Soit on donne une autre valeur à la clé étrangère (**NumVol**) en attendant qu'une modification soit faite (par un **UPDATE**) sur un autre vol :
    - ... **ON DELETE SET NULL** ou bien
    - ... **ON DELETE SET DEFAULT**

## Que faire quand vous en avez terminé avec une table ?

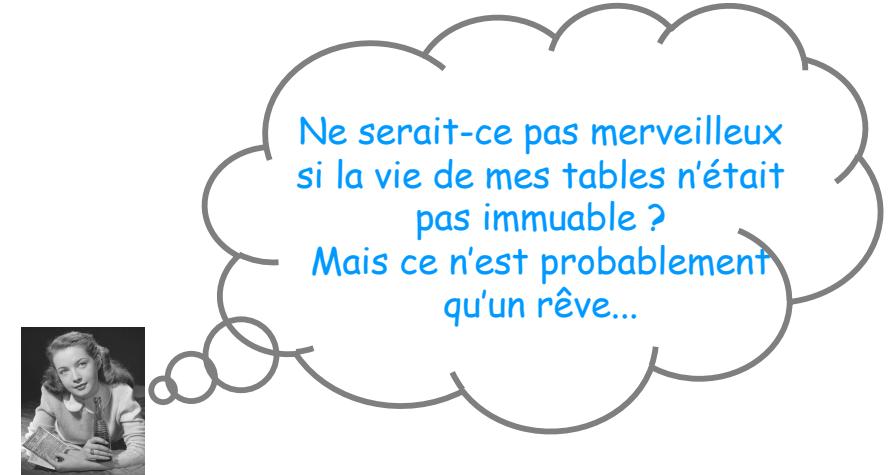
- Suppression d'une table dans un schéma de base de données :
  - **DROP TABLE** **Genre** ;
  - La table ne peut pas être supprimée s'il existe des contraintes d'intégrité référentielle avec une autre table (**Film**, **Acteur**).
- Suppression d'une table avec **suppression** des contraintes **d'intégrité référentielle** :
  - **DROP TABLE** **Genre** **CASCADE CONSTRAINTS** ;
  - SGBD autorise suppression de la table même si sa clé primaire est une clé étrangère dans d'autres tables.

## Comment vérifier que des valeurs appartiennent à un domaine ?

- ```
DROP TABLE IF EXISTS Genre ;  
CREATE TABLE Genre (  
    NumGenre INTEGER NOT NULL PRIMARY KEY,  
    LibelleGenre VARCHAR(20) NOT NULL,  
    CHECK (LibelleGenre in (Fantastique, Comédie, ...))  
)  
▪ Une clause CHECK peut aussi vérifier une contrainte de valeur sur une ligne (et non sur 1 seule colonne) :  
    CONSTRAINT CK_Produit_Qte *  
        CHECK (qte_cdée >= qte_livrée)
```

\* Nommer les contraintes permet de les référencer ultérieurement.

## I have a dream today (1)



## I have a dream today (2)



## Est-ce que la structure d'une table est définitive ?

- Dans la vie d'une table, certaines colonnes peuvent être ajoutées ou supprimées voire modifiées.
  - ADD COLUMN ajoute une colonne à votre table, vous choisissez le type de données.
  - ALTER COLUMN change aussi bien le nom, le type, l'obligation d'existence, la valeur par défaut, etc. d'une colonne existante \*.
  - DROP COLUMN supprime 1 colonne de votre table \*.

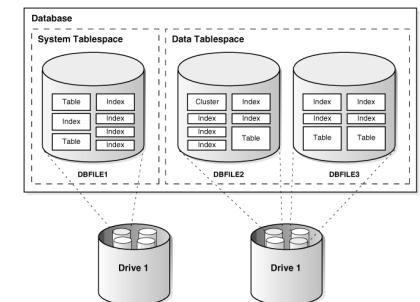
\* Risque de perte de données - pas de garantie.

## Quand est-il pertinent d'ajouter de nouvelles contraintes à une table ?

- En cas de chargement de masse, pour éviter la mise à jour des index et la perte de temps due à la vérification des contraintes, elles sont souvent ajoutées dans un 2<sup>ème</sup> temps.
  - **ALTER TABLE** Film  
ADD CONSTRAINT FK\_Film\_Genre  
FOREIGN KEY (Genre)  
REFERENCES Genre (NumGenre);

## Est-ce que l'environnement d'une table peut changer ?

- Une table peut changer de nom, de schéma, de tablespace, de propriétaire, être intégrée à un cluster, hériter d'une autre table, etc.
- Vous pouvez également lui associer (ou l'en libérer) des triggers ou des règles.



## I have a dream today (1)



Ne serait-ce pas merveilleux  
s'il était possible d'offrir aux  
utilisateurs une présentation  
des données adaptée à leurs  
besoins ?  
Mais ce n'est probablement  
qu'un rêve...

## Qu'est-ce qu'une vue ?



- Une VUE est une table qui n'existe que quand vous utilisez la vue dans une requête.
- Elle est déclarée ainsi :  

```
CREATE VIEW concepteurs_web AS
    SELECT mc.prenom, mc.nom, mc.telephone, mc.email
    FROM mes_contacts mc
    NATURAL JOIN emploi_voulu ev
    WHERE ev.profession = 'Concepteur web';
```
- Et est utilisée ainsi :  

```
SELECT * FROM concepteurs_web;
```

## I have a dream today (2)



Des VUES,  
Des VUES,  
à perte de vue...

## Que se passe-t-il en réalité ?

- Quand vous utilisez votre vue dans une requête, tout se passe en fait **comme si c'était une sous-requête** :  

```
SELECT * FROM
    (SELECT mc.prenom, mc.nom, mc.telephone, mc.email
    FROM mes_contacts mc
    NATURAL JOIN emploi_voulu ev
    WHERE ev.profession = 'Concepteur web')
    AS concepteurs_web;
```
- Elle est considérée comme une table virtuelle parce qu'elle agit comme une table et que les opérations qui peuvent être réalisées sur une table peuvent aussi l'être sur une vue.

## Pourquoi les vues sont elles si pratiques ?

- En créant des vues, vous empêchez que les **changements de structure** de votre BD ne « fassent planter » les applications qui dépendent de vos tables.
- Les vues vous simplifient la vie en transformant une requête complexe en une simple **commande stockée** une bonne fois qu'il est alors inutile de réécrire.
- Vous pouvez également créer des vues pour **cacher les informations** non nécessaires à l'utilisateur ou auxquelles il ne devraient pas accéder.

64

## Comment garantir que le SGBD vérifie que les MAJ sont conformes à la définition de la vue (1) ?

- Soit les tables et vues :
  - CREATE TABLE tirelire (  
Id INT AUTO\_INCREMENT NOT NULL PRIMARY KEY,  
piece DEC(3,2) NOT NULL, pays\_piece CHAR(20)  
)
  - CREATE VIEW 20cents AS  
SELECT \* FROM tirelire WHERE piece = 0.20;
  - CREATE VIEW 10cents AS  
SELECT \* FROM tirelire WHERE piece = 0.10  
**WITH CHECK OPTION;**

66

## Peut-on modifier (insert, update, delete) des données à travers une vue ?

- Oui mais à condition que la vue soit capable de **propager les modifications vers les tables de base**, donc dans les conditions suivantes :
  - La vue contient la clé primaire de la table de base y compris pour les vues définies sur une jointure.
  - La vue ne contient pas de sous-requête qui cite la même table.
  - La vue ne contient pas de fonction agrégatives (sum, count, etc.), ni de clause group by et distinct.
  - La vue n'est pas construite à partir d'opérateurs ensemblistes (union, intersect, etc.).

65

## Comment garantir que le SGBD vérifie que les MAJ sont conformes à la définition de la vue (2) ?

- Et les instructions suivantes :
  - INSERT INTO 20cents (piece, pays\_piece)  
VALUES (0.20, 'Luxembourg');
    - OK conforme aux attentes
  - INSERT INTO 20cents (piece, pays\_piece)  
VALUES (0.10, 'Autriche');
    - OK contrairement aux attentes : pas de with check option
  - INSERT INTO 10cents (piece, pays\_piece)  
VALUES (0.20, 'France');
    - KO - with check option empêche l'insertion dans la table
  - DELETE FROM 20cents WHERE piece = 0.50 OR piece = 0.10;
    - KO – n'a pas d'impact sur la table puisque la vue ne regarde que les lignes où piece = .020



67

## Que faire quand vous en avez terminé avec une vue ?

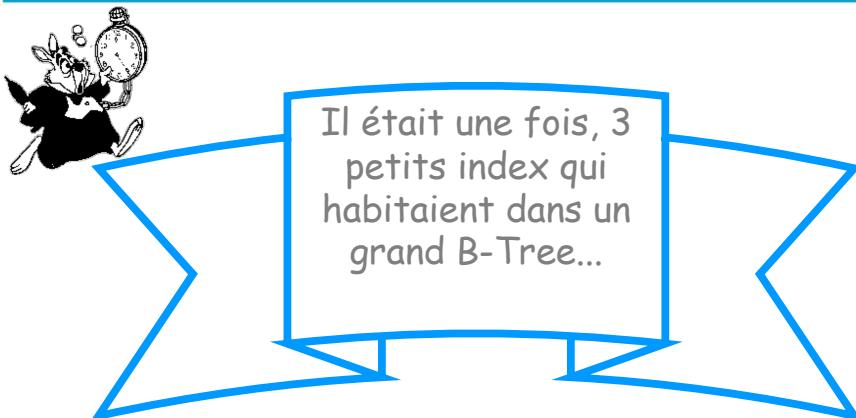
- **DROP VIEW <nom\_de\_la\_vue>** permet de supprimer une vue.
- Contrairement à la commande **DROP TABLE**, les données incluses dans votre vue **ne seront pas détruites**, ce qui est logique puisqu'une vue n'est que la table temporaire générée suite à l'exécution d'une requête.
- Seule la définition de la vue est supprimée du catalogue.

## I have a dream today (1)



Ne serait-ce pas merveilleux s'il existait un moyen d'accéder directement à une donnée précise sans parcourir en séquentiel toutes celles qui la précède dans la table ? Mais ce n'est probablement qu'un rêve...

## I have a dream today (2)



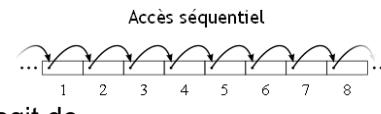
## Comment sont rangées les données dans une BD ?

- Une BD est implémentée sur 1 ou plusieurs disques.
- L'espace disque est découpé en **pages** (**blocs**) adressables. La page est **la plus petite unité d'échange** entre le disque et la RAM.
- Un fichier (ou espace de stockage) est formé d'une suite de pages.
- Les lignes (**t-uplets**) des tables sont rangées dans **les pages d'un fichier**.

## Quel mode d'accès aux données coexistent ?

### ■ Accès séquentiel :

- Parfait s'il s'agit d'exploiter toutes les données successivement,
- Temps de réponse prohibitif s'il s'agit de n'accéder qu'à quelques données dispersées.



### ■ Accès direct :

- Grâce à un **index** qui connaît le **ROWID** du T-uplet défini ainsi :
  - N° objet dans le catalogue
  - N°bloc dans le fichier
  - N°ligne dans bloc
  - N° fichier sur disque.
- Par calcul grâce à une **fonction de hachage** dans un **cluster**.

## Comment ça marche ?

Waouh,  
serait-ce  
comme  
une  
TreeMap ?



- Les index sont organisés selon une structure d'arbre équilibré (**B-Tree**) :
  - Comportent 3 types de blocs : racine, branches et feuilles.
  - Bloc **racine** et blocs **branches** pointent sur blocs feuilles qui contiennent les informations permettant un accès direct aux lignes.
  - Toutes les **feuilles** sont à la même profondeur : recherche prend même temps  $\forall$  valeur clé.
  - Les blocs feuilles sont **chaînés** entre eux pour faciliter le balayage croissant et décroissant des valeurs de clés.
- Le coût d'une recherche correspond à la hauteur de l'arbre + 1 :
  - 3 blocs lus dans l'index, (4 pour de très grosses tables),
  - + 1 bloc lu à la bonne adresse dans la table.

## Qu'est-ce qu'un index ?

Tiens,  
une  
MAP !



- Un **index** est associé à une table et permet de retrouver l'**adresse** (ROWID) d'un t-uplet connaissant une valeur de **clé** (simple ou concaténée).
- La majorité des SGBD créent automatiquement un index à la déclaration d'une **contrainte d'unicité** ou d'une **clé primaire** sur une table.
- Vous pouvez créer vos propres index pour accélérer les recherches, en particulier sur les **clés de jointure** (Foreign Key) :
  - **CREATE [UNIQUE | BITMAP \*] INDEX nomIndex ON nomTable (nomCol1 [ASC | DESC], ...)**

\* Oracle

## Peut-on indexer tous les attributs ?

### ▪ Quels attributs indexer ?

- Les clés primaires et les regroupement d'attributs uniques (création automatique de l'index).
- Les colonnes servant de critère de jointure.
- Les colonnes servant de critère de recherche.

### ▪ Quels attributs ne pas indexer ?

- Les attributs non discriminants (code postal) : l'index serait alors peu efficace (renvoie plus de 15% des lignes).
- Les attributs fréquemment modifiés.
- Les attributs pouvant être NULL.
- Les attributs toujours sollicités par l'intermédiaire d'une expression dans la clause WHERE.



# CONTACT

Domaine Scientifique de la Doua  
43, bd du 11 Novembre 1918 - Bâtiment Hubert Curien  
B.P. 2077 - 69616 Villeurbanne cedex - France

Tél. : (33) 04 72 43 17 00  
Fax : (33) 04 72 43 16 84

[www.cpe.fr](http://www.cpe.fr)

[francoise.perrin@cpe.fr](mailto:francoise.perrin@cpe.fr)

membre de  UNIVERSITÉ DE LYON

