# Introduction aux Systemes d'Information et aux Bases de Données

## L2 Informatique

Serenella Cerrito

Département d'Informatique – Évry

2014-2015

# Plan du cours

1. Introduction, Motivations et Objectifs
2. Le modele dit *relationnel* : c'est quoi ? Notions de base
3. Comment concevoir un SGBD relationnel ? Méthode EA
4. SQL illustré a l'aide des opérateurs de l'*algebre rélationnelle*
5. Quelques notions sur le stockage physique des données
6. Ouverture : que reste-t-il a savoir sur les SGBD ?

# Informations Utiles

Supports de cours en ligne sur ma page web (laboratoire Ibisc)

Mon adresse e-mail : Serena.Cerrito@ibisc.univ-evry.fr

# Objectifs du cours

- Connaître les principales caractéristiques d'un systeme d'information
- Pouvoir participer à la modélisation d'une base de données
- Comprendre la structure d'une base de données
- Apprendre les bases du langage SQL

# Exemple de système d'information

Vente de produits (fruits, légumes) par une coopérative :
Produits achetés à un producteur et revendus à des clients

**Différentes activités** : Achat de lot à un producteur, vente de produit à un client, suivi de l'évolution des stocks, etc.

**Informations à gerer** : Identification des produits avec prix de vente, d'achat et quantités, coordonnées des producteurs et des acheteurs, etc.

# Généralités sur les ystèmes d'information, 1

système d'information = une représentation opérationnelle de la réalité

Données (informations associées aux activités de l'organisation) +
Outils (manipulation des données, aide à la décision)

# Généralités sur les ystèmes d'information, 1

## Système d'information, vue informatique

Un système informatique ;

- Un serveur : Une base de données + une application fournissant les outils
- Des postes "clients" de travail (utilisation du web)

# Généralités sur les ystèmes d'information, 3

Une démarche de projet de réalisation d'un système d'information informatique :

1. Analyse des besoins, cahier des charges, étude de faisabilité
2. Spécifications, conception et validation du système
3. Mise en production et exploitation du système

# Rôle primordial des données

Base de données = Ensemble structuré d'informations cohérentes et pérennes, correspondant ici aux activités d'une organisation.

Modèlisation de données : Analyse des informations manipulées dans l'organisation avec représentation formelle de leur nature et leur structuration. Plusieurs sortes de modèles existent. Ici : modèle dit *relationnel*.

Implémentation des données : Utilisation d'un système de gestion de base de données (SGBD), et d'un langage de manipulation des données (ici : SQL)

# Spécificités d'un SGBD

SGBD= **S**ystème de **G**estion d'une **B**ase de **D**onnées

- Très grande quantité de données à gérer, qui doivent être stockées dans plusieurs fichiers, voir plusieurs sites.
- Besoin d'interroger et/ou mettre à jour souvent, rapidement et facilement ces données.
- Besoin d'accès concurrents.
- Besoin de sécurité.
- Besoin important de gérer des pannes éventuelles.

# Indépendance niveau “logique” /physique (implémentation)

1. Utilisateur d’une BD (base de données) : pas forcément un pro de l’implémentation.
2. L’implémentation peut changer, sans que le “schéma” (la “forme conceptuelle”) de la BD change.
3. Modèle logique clair $\Rightarrow$
   - 3.1 possibilité d’un *langage de requêtes* facile pour l’utilisateur
   - 3.2 si l’implémentation change, pas besoin d’écrire un nouveau programme pour poser la même question à la base !
4. Idem pour le *langage de mise à jour*.

# Quel modéle logique ?

## Historique

- Avant 1970 : BD=fichiers d'enregistrements, "modèles" *réseaux* et *hiérarchique*; pas de vraie indépendance logique/physique.
- En 1970 : modèle *relationnel* (Codd) : vraie indépendance logique/physique.
- Années 80 et 90 : nouveaux modèles :
  modèle à objets
  modèle à base de règles (Datalog)
- Depuis la in années 90 : données dites *semi-structurées* (XML).

Ce cours : modèle relationnel, et SQL comme langage de manipulation des données.

# Notions essentielles des BD relationnelles, 1

Une BD relationnelle = un ensemble de tables

Exemple : table Cinema (à Paris) :

| Nom_cinema | Arrondissement | Adresse |
|---|---|---|
| Rex | 2 | 22 bvd Poissoniere |
| Kino | 13 | 3 bvd Raspail |
| Halles | 1 | Forum des Halles |

Nom_cinema, Arrondissement, Adresse = *Attributs*

Une ligne = un $n$-uplet (ici $n$=3). Que disent les triplets de cette table ?

# Notions essentielles des BD relationnelles, 2

Comment on déclare la forme d'une table en SQL ?

**CREATE TABLE** Cinema (
Nom_Cinema varchar(10) **NOT NULL**
Arrondissement decimal(2,0)
Adresse Varchar(30)
**PRIMARY KEY** (Nom_cinema)

Que vient on de déclarér?
Notions de : schéma d'une table, domaine d'un attribut, clé (primaire) d'une table, contrainte d'intégrité.

# Notions essentielles des BD relationnelles, 3

- schéma d'une table = nom de la table, et l'ensemble de ses attributs. Ici : Cinema et $\{Nom, Arrondissement, Adresse\}$.
- domaine d'un attribut $A$ = l'ensemble de valeurs que $A$ peut prendre
- clé (primaire) d'une table = ensemble d'attributs suffisant à détérminer les valeurs de tous les autres = identifiant d'une ligne. Dans l'exemple, la clé a un seul élément : Nom_Cinema
- contrainte d'intégrité = une contrainte sur les valeurs possibles. Ici : Nom_Cinema ne peut pas prendre la valeur spéciale **NULL**

# Notions essentielles des BD relationnelles, 4

Avec **CREATE TABLE** on défini juste le schéma d'une table, on n'insére pas de données.

Pour remplir une table avec SQL :

**INSERT INTO** Cinema **VALUES** ('Rex', '2', '22 bvd Poissoniere')

etc. (*Requêtes d'Insertion*)

# Notions essentielles des BD relationnelles, 5

Table = Relation. Pourquoi ?
Du point de vue mathématique, une table R ayant attributs $A_1, ..., A_n$, où $Dom(A_i)$ est le domaine de l'attribut $A_i$, est une relation finie sur $Dom(A_1) \times \cdots \times Dom(A_n)$, c'est à dire un ensemble de $n$-plets de la forme $\langle v_1, \cdots, v_n \rangle$ où $v_i \in Dom(A_i)$, pour $i \in [1, ..., n]$.

Une base de données relationnelle $\mathcal{B}$ est un ensemble fini de tables, donc un ensemble fini de relations finies.

On donne le schéma de la base de données relationnelle $\mathcal{B}$ en donnant le schéma de toutes ses tables.

# Notions essentielles des BD relationnelles, 6

Soient $X$ et $Y$ deux ensembles d'attributs d'une table $R$. Une dépendance fonctionnelle pour $R$ s'écrit :

$$X \rightarrow Y$$

et se lit :
$X$ détermine (ou donne) $Y$
ou encore : $Y$ dépend fonctionnellement de $X$.

Signification ? : La table $R$ satisfait $X \rightarrow Y$ ssi :
qques soient les $n$-uplets $t, t'$ de $r$, si $t$ et $t'$ sont egaux sur les attributs de $X$, alors le sont aussi pour ceux de $Y$.

Convention : si $X = \{A_1, \cdots, A_n\}$ et $Y = \{B_1, \cdots, B_m\}$ on peut écrire, si pas de confusion possible :
$A_1 \cdots A_n \rightarrow B_1 \cdots B_m$ (pas de $\{,\}$, pas de virgule).

Exemple de dépendances fonctionnelle pour une table $R$ :

| A | B | C | D | F |
|---|---|---|---|---|
| a1 | b1 | c1 | d1 | f1 |
| a1 | b1 | c1 | d2 | f1 |
| a1 | b2 | c2 | d2 | f1 |

Les quelles, parmi les dépendances fonctionnelles suivantes ($DF$), sont satisfaites par $R$ ?
$A \rightarrow B$, $AB \rightarrow C$, $AB \rightarrow D$, $ABD \rightarrow F$

Qui pourrait être une clé primaire, ici ?

# Notions essentielles des BD relationnelles, 7

- Associer un ensemble de *DF* à un schéma d'une base sert aussi à fixer en avance les *clés* (primaires, candidates) des tables.
- Clé d'une table $R$ ?? Intuition : identifiant de chaque $n$-uplet de $r$.
- Formellement : si $S$ est l'ensemble d'attributs d'un schéma d'une relation $R$ et $X \subseteq S$, alors :
  $X$ est une clé (primaire, candidate) pour la table $R$ ssi
  1. $R$ satisfait $X \rightarrow S$
  2. $X$ est minimal par rapport à cette propriété : $\nexists\, Z$ t.q. $Z \subset X$ et $R$ satisfait $Z \rightarrow S$
- **Exemples au tableau**

# Concevoir des bases de données relationnelles

Les étapes de la conception d'un schéma de base de données relationnelel

1. Analyse des besoins (informelle)
2. Utilisation d'un outil "graphique" pour une première modélisation : méthode *Entité Association*) (EA ou ER).
3. Passage de la représentation graphique au schéma relationnel (avec ses *dépendances fonctionnelles* (DF) : presque mécanique.
4. Analyses ulterieures, pour raffiner : pas étudiées ici (cours SGBD L3).

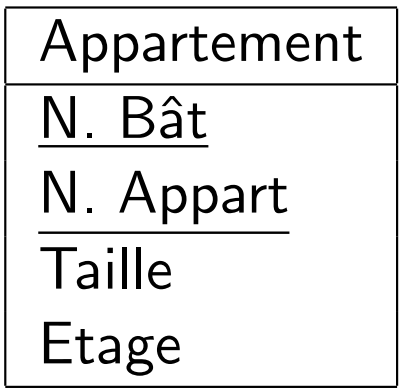
# Méthode Entité-Association (dite aussi "Modèle EA"), 1

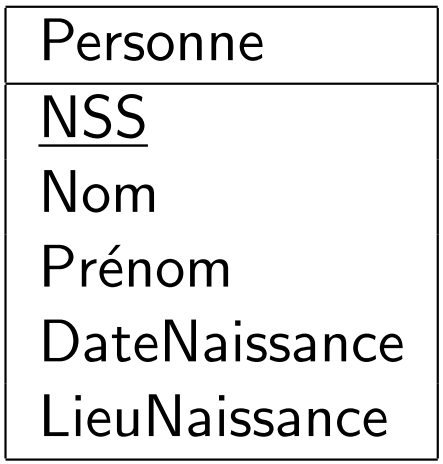
(Merci à M. M. Rahoual pour ses sources pédagogiques sur le sujet)

- Entité= un nom qui désigne un ensemble d'objets (instances).
- Par ex. : Personne, Film, Aéroport, Appartement, sont des entités. "Orly" et "Roissy - Charles de Gaulle" sont 2 instances de Aéroport.
- Attributs d'une entité : propriétés des objets de l'entité. Par ex. attributs de Appartement : N.Bât, N.Appart, Taille, Etage.
- Ensemble d'attributs permettant d'identifier les instances d'une entité : *clé*. Par ex., clé de Appartement : N.Bât, N.Appart.

# Méthode EA, 2

Représentation graphique d'une entité : un rectangle, clé soulignée.



# Méthode EA, 3

- Association : Un lien A qui relie au moins deux entités (éventuellement E avec E).
- Idée : Si E1 et E2 sont reliées par A, A indiques un ensembles de couples $\langle e_1, e_2 \rangle$ tels que $e_1$ est une instance de E1 et $e_2$ est une instance de E2.
- Par ex., l'association (binaire) Commande entre Client et Produit regroupe des couples $\langle c, p \rangle$ tq $c$ est un client et $p$ est un produit.
- Représentation graphique : les entités sont les sommets d'un graphe dont les arcs sont les associations. Un arc est étiquetée par un losange avec le nom de l'association.

# Méthode EA, 4

**Exemples de représentations graphiques des associations :**

**1** : commande, binaire, entre client et produit,

**2** : commande, ternaire, entre client, produit et fournisseur,

**3** : dirige, binaire mais réflexive, entre employé et employé.

Figure 1 au tableau

# Méthode EA, 5

- Une association A a comme attributs les clés des entités reliées. Ces attributs forment la clé de l'association A.
- A peut avoir aussi des *attributs spécifiques*, mais, attention : ils dépendent de toutes les entités reliées.

Par ex. : association Catalogue entre l'entité Fournisseur, dont la clé est NFournisseur, et l'entité Produit, dont la clé est code. Attributs de Catalogue : NFournisseur, code + Prix.

Prix est spécifique. *Il dépend à la fois du fournisseur et du produit !*

Figure 2 au tableau

Pour $1 \leq i \leq n$ :
$a_i$ = nombre minimal de $n-1$-uplets de
$E_1 \times E_2 \times \cdots \times E_{i-1} \times E_{i+1} \times E_n$ auxquels une instance qque de $E_i$ peut être reliée par A. Valeurs possibles : 1 et 0.
$b_i$ = nombre maximal de $n-1$-uplets de
$E_1 \times E_2 \times \cdots \times E_{i-1} \times E_{i+1} \times E_n$ auxquels une instance qque de $E_i$ peut être reliée par A. Valeurs possibles : 1 et N (= tout $m > 1$).

**Exemple 1**

Un département peut ne pas avoir de président et il en a au max. 1. Un président dirige exactement un département (pas +, pas -).

Figure 3 au tableau

**Exemple 2**

Un client peut ne passer aucune commande de produit à un fournisseur, mais peut aussi en passer autant qu'il veut.
Un produit peut être associé à un nombre quelconque de couples (client, fournisseur), éventuellement à aucun.
Un fournisseur doit fournir au moins un produit à au moins un client, éventuellement il peut recevoir plusieurs commandes.

Figure 4 au tableau

# Règles à suivre pour avoir un diagramme EA OK

*Règle 1 : Existence d'un identifiant pour chaque entité.*
(PB. : Modéliser une "situation" par une entité ou par une association ?)

**Situation : des clients commandent des produits** On suppose que le concept de commande inclue un num. de commande unique pour chaque commande. Modéliser ce concept par une entité :
Figure 5 au tableau

On ne fait pas cette hypothèse :
on peut modéliser par une association.
Clé de l'association : clé de Client, clé de Produit.
Figure 6 au tableau

# Règles à suivre pour avoir un diagramme EA OK

*Règle 2A : Pour chaque entité $E$, tout attribut de $E$ qui n'est pas dans la clé $C$ de $E$ doit être déterminé par $C$, mais un sous-ensemble propre $C'$ de $C$ ne doit pas suffire à le déterminer.*

**Exemple**. Hyp : toute série (TV) a un seul producteur. Violation de la règle 2A, puis solution alternative, qui introduit une nouvelle entité.

Voir les figures 7 et 8 au tableau

# Règles à suivre pour avoir un diagramme EA OK

Motivation intuitive de la règle 2A ? On veut éviter des "dépendances partielles" dans le schéma relationnel final, car sinon on risque des incohérences de données.
On dira (cours SGBD, L3) : On veut forcer le respect de la 2nde Forme Normale.

# Règles à suivre pour avoir un diagramme EA OK

*Règle 2B : Pour chaque entité ayant clé $C$, si un attribut $A$ est déterminé par un ensemble d'attributs $E$, on doit avoir $E \subseteq C$.*
(Dépendance "directe" de $A$ à partir de $C$).

**Exemple**. Hyp : tout modèle de voiture détermine la marque de la voiture, mais l'entité Voiture a comme clé Immatriculation.
Violation de la règle 2B, puis solution alternative.
Figure 9 au tableau : violation de la règle 2B
Puisque : a) Immatriculation (clé de voiture), *détermine* Modèle et b) Modèle *détermine* Marque (mais Modèle ne fait pas partie de la clé), pour résoudre le problème il suffit de:
enlever Marque et Modèle de Voiture et créer la nouvelle entité TypeVoiture, dont la clé est Modéle et l'autre attribut est Marque.
Figure 10 au tableau

# Règles à suivre pour avoir un diagramme EA OK

Motivation intuitive de la règle 2B ?
On veut éviter, dans le schéma relationnel résultat, des "dépendances transitives".
On dira (cours SGBD, L3) : On veut forcer le respect de la 3ème Forme Normale.

# Règles à suivre pour avoir un diagramme EA OK

*Règle 3 : Tous les attributs d'une association A doivent dépendre complètement de la clé de cette association (= ensemble des clés des entités reliées).*

**Exemple de cas de violation de la Règle 3** :
Figure 11 au tableau
Le prix dépend seulement du fournisseur et du produit, pas du client.
**Solution** ? Figure 12 au tableau
On a effacé l'attribut Prix de l'association Achat. Il est devenu une association dont les seuls attributs sont sa clé : N. Fournis (clé de Fournisseur), code (clé de Produit).

# Règles à suivre pour avoir un diagramme EA OK

Motivation intuitive de la règle 3 ?

A nouveau, in veut éviter, dans le schéma relationnel résultat, des "dépendances partielles".
(SGBD L3 : On veut forcer le respect de la 2nde Forme Normale).

# Transformer un diagramme EA en un schéma relationnel

Les entités et les associations vont devenir des (schémas de) tables.

Principes concernant les entités

1. Entité $E \rightsquigarrow$ Schéma $S_E$ d'une table $E$.
2. Attribut de l'entité $E \rightsquigarrow$ Attribut de $S_E$.
3. Clé de l'entité $E \rightsquigarrow$ Clé de la table $E$. Si $At = \{A_1, \cdots, A_n\}$ est l'ensemble des attributs de l'entité $E$ et $C \subseteq At$ est la clé de cette entité, alors on impose la dép. fonct. $C \rightarrow A_1 \ \cdots \ A_n$.

| Client |
|---|
| N. Client |
| Nom |
| Prénom |

$\rightsquigarrow$ :

| | **Client** | |
|---|---|---|
| N. Client | Nom | Prénom |
| | | |

# Transformer un diagramme EA en un schéma relationnel

## Principes concernant les associations

**A1 : Association binaire $A$ ayant N en cardinalité max des 2 cotés** $\rightsquigarrow$

Schéma $S_A$ d'une table $A$. Attributs de $S_A$ : ceux de $A$, clé $C$ pour la table $A$ : clé $C$ de l'association. On dit que chaque élément de $C$ est une clé étrangère de la table crée.

Figure 13 au tableau

Schéma relationnel correspondant :
Client(N.client, Nom, Prénom), Produit(Code),
Comm(N.client, Code, date).
(*Deux clés étrangères de* Comm : 1) N.client et 2) Code)
*DF* : N.client $\rightarrow$ Nom Prénom et N.client Code $\rightarrow$ date

# Transformer un diagramme EA en un schéma relationnel

Principes concernant les associations

**A2 : Association binaire $A$ ayant 0,1 en cardinalité sur au moins un des 2 cotés :**
Si SGBD tolère NULL $\rightsquigarrow$ pas de table $A$. Les attributs de l'association $\rightsquigarrow$ des attributs de la table associée à l'entité ayant 1 en cardinalité max.
Sinon $\rightsquigarrow$ Schéma $S_A$ d'une table $A$. Attributs de $S_A$ : ceux de $A$; clé pour la table $A$ = la clé de l'association.

**Exemple** : Figure 14 au tableau, qui illustre ce cas d'association

## Principes concernant les associations

**Suite de l'exemple pour la règle A2 (la figure 14)**

- NULL accepté : Client(N.Client,Nom, Prénom), Tableau(Réf, Peintre, N.Client, Prix). **NB** : N.Client est attribut de la table Tablealeau !.
  *Que se passe-t-il si le tableau X de Th. Van Gogh n'est jamais vendu, ici ?*
- NULL interdit : Client(N.Client,Nom, Prénom), Achète(N.Client, Réf, Prix), Tableau(Réf, Peintre).
  *Que se passe-t-il si le tableau X de Th. Van Gogh n'est jamais vendu, ici ?*

# Transformer un diagramme EA en un schéma relationnel

## Principes concernant les associations

**A3: Association binaire $A$, entre E1 et E2, ayant 1,1 en cardinalité sur au moins un des 2 cotés :** $\rightsquigarrow$ pas de table $A$.
Soit E2 l'entité "du coté 1,1".
Dans la table T2, on insère les attributs de l'association, dont, en particulier, la clé de E1, qui dévient clé étrangère de T2.
Figure 15 au tableau
Client(N.Client,Nom, Prénom),
Tableau(Réf, Peintre, N.Client, Prix).
Clé étrangère de Tableau : N.Client.

# Transformer un diagramme EA en un schéma relationnel

Principes concernant les associations

**A4 : Association binaire $A$ REFLEXIVE :** $\rightsquigarrow$ toujours un nouveau schéma de table (peu importent les cardinalités), comme suggéré par l'exemple.

Figure 16 au tableau

Employé(Num, Nom, Prénom), Dirige(Num_Sup, Num_Subalt)

# Transformer un diagramme EA en un schéma relationnel

## Principes concernant les associations

**A5 : Association $n$-aire $A$ avec $n > 2$** $\rightsquigarrow$ toujours un nouveau schéma de table $S(A)$.
Si toutes les cardinalités max sont N, la clé de la table $A$ est l'union des clés associées aux entités.
Sinon, prendre comme clé de la nouvelle table la clé d'une entité ayant 1 en cardinalité max.

## Figure 17 au tableau

Soit A(a1,a2,a3), B(b1,b2,b3), C(c1,c2,c3), R(a1,b1,c1,r1), soit A(a1,a2,a3), B(b1,b2,b3), C(c1,c2,c3) R(a1,b1,c1,r1).

# Interroger les données

- SQL : Langage commercial qui permet de créer des tables, d'insérer des données, et des les interroger.
- Fondé sur 2 langages formels : l'*algebre relationnellr* (AR) et le *calcul relationnel* (CR) (c'est de la logique !), définis dans les années 70 par Codd.
- Dans ce cours : explication intuitive des bases de SQL, à l'aide de notions de l'AR.

# Algèbre Relationnelle (AR) : Notations

L'Algèbre Relationnelle (AR) est un ensemple d'operateurs qui s'appliquent à des tables (= relations finies) pour produire une nouvelle table.

- Si $R$ est une table, $S(R)$ indiquera son schema, c'est à dire son ensemble d'attributs.
- Si $E$ est un ensemble d'attributs, et $n$ est un $n$-uplet, on dit que $n$ est sur $E$ si ses valeurs sont des valeurs pour $E$.
  Par ex., Si $R$ est une table et $S(R)$ est $\{TitreFilm, Cinema, HoraireProjection\}$, un triplet $n$ de $R$ pourrait être $\langle Bird, Rex, 14.00\rangle$, et on dirait qu'il est sur l'ensemble d'attributs $S(R)$.
- Si $n$ est un $n$-uplet sur $E$ et $E' \subseteq E$, alors la restriction de $n$ à $E'$ est la partie de $n$ qui utilise exclusivement les attributs de $E'$, et on la note $n(E')$ Par ex., pour le triplet précédent : si $E' = \{TitreFilm, Cinema\}$, $n(E') = \langle Bird, Rex\rangle$.

# Algèbre Relationnelle (AR)

## Les opérateurs de l'algèbre relationnelle

- Opérateurs ensemblistes : union ($\cup$), intersection ($\cap$), différence ($\setminus$), produit cartésien ($\times$)
- projection sur un ensemble d'attributs $E$ ($\pi_E$), sélection d'un ensemble de $n$-uplets selon une condition $C$ ($\sigma_C$), jointure "naturelle" ($\bowtie$), division ($\div$), renommage ($\rho$).

# Algèbre Relationnelle (AR)

Union, intersection, différence. Arguments : 2 relations $r$ et $r'$ de même schéma $S$. Résultat : une nouvelle relation, encore sur $S$.
Notation : ici et après, $n$ indique un $n$-uplet.

$$r \cup r' = \{n \mid n \in r \text{ ou } n \in r'\}$$

$$r \cap r' = \{n \mid n \in r \text{ et } n \in r'\}$$

$$r \setminus r' = \{n \mid n \in r \text{ et } n \notin r'\}$$

# Algèbre Relationnelle (AR)

Projection $\pi$. Arguments : 1 relation $r$. Résultat : une nouvelle relation dont le schéma est inclus dans celui de $r$.
$S =$ schéma de $r$, $E \subseteq S$.

$$\pi_E(r) = \{n(E) \mid n \in r\}$$

Le schéma de la relation résultat, est, évidemment, $E$.

Ecriture équivalente :

$$\pi_E(r) = \{m \mid \exists n\, (n \in r \text{ et } m = n(E)\}$$

# Algèbre Relationnelle (AR)

Notation : si $A$ est un attribut, $Dom(A)$ indique le domaine de $A$ = l'ensemble de valeurs que $A$ peut prendre .

Sélection.

- **Condition de Sélection** $C$.
  *Atomes* : $A_i$ *op* $A_j$ ou $A_i$ *op* $v$ où :
  $A_i$ et $A_j$ sont des attributs, $v \in Dom(A_i)$,
  *op* $\in \{=, \neq, >, <, \geq, \leq\}$.
  $C$ est une formule booléenne construite à partir des atomes.
  (Exemples au tableau)
- **Opérateur de Sélection** $\sigma_C$. Arguments : 1 relation $r$.
  Résultat : une nouvelle relation sur le même schéma que $r$.

$$\sigma_C(r) = \{n \mid n \in r \text{ et } n \text{ rend vraie } C\}$$

# Algèbre Relationnelle (AR)

- Produit Cartésien $\times$. Arguments : 2 relations $r$ et $r'$, de schémas $S$ et $S'$, telles que $S$ et $S'$ sont disjoints. Résultat : une nouvelle relation dont le schéma est $S \cup S'$.

$$r \times r' = \{n \text{ sur } S \cup S' \mid n(S) \in r \text{ et } n(S') \in r'\}$$

- Jointure "naturelle" $\bowtie$. Arguments : 2 relations $r$ et $r'$, de schémas $S$ et $S'$. Résultat : une nouvelle relation dont le schéma est $S \cup S'$.

$$r \bowtie r' = \{n \text{ sur } S \cup S' \mid n(S) \in r \text{ et } n(S') \in r'\}$$

**N.B.** : Si $S$ et $S'$ sont disjoints, le résultat de $r \bowtie s$ est le même que celui de $r \times s$. Donc : $\times$ = cas particulier de $\bowtie$.

# Algèbre Relationnelle (AR)

Notation : si $n$ et $m$ sont 2 $n$-uplets, notons $n \times m$ l'unique élément de la relation $\{n\} \times \{m\}$.

Division.

Arguments : 2 relations $r$ et $r'$, de schémas $S$ et $S'$ tels que $S' \subset S$. Résultat : une nouvelle relation dont le schéma est $S \setminus S'$.

$$r \div r' = \{n \text{ sur } S \setminus S' \mid n \in \pi_{S \setminus S'}(r) \text{ et } \forall\, m \in r',\ n \times m \in r\}$$

**Exemple**.

*AimeLivre*

| Pers | NomLivre |
|---|---|
| p1 | l1 |
| p2 | l2 |
| p1 | l2 |

*Livre*

| NomLivre |
|---|
| l1 |
| l2 |

"Qui aime tous les livres ?" : $AimeLivre \div Livre$

# Algèbre Relationnelle (AR)

Renommage.

Arguments : une relations $r$ , de schéma $S$, un attribut $A \in S$ et un nouveau attribut $A' \notin S$. Résultat : une copie de la relation $r$ où l'attribut $A$ est renommé en $A'$.

Schéma de la copie : $(S \setminus \{A\}) \cup \{A'\}$.

Ecriture : $\rho_{A \rightsquigarrow A'}(r)$

**Exemple** : $\rho_{NomLivre \rightsquigarrow NomLivre2}(AimeLivre) =$

| Pers | NomLivre2 |
|---|---|
| p1 | l1 |
| p2 | l2 |
| p1 | l2 |

"Qui aime au moins deux livres ?" :

$$\pi_{Pers}(\sigma_{NomLivre \neq NomLivre2}(AimeLivre \bowtie \rho_{NomLivre \rightsquigarrow NomLivre2}(AimeLivre)))$$

# Introduction à SQL

On va voir comment SQL exprime les opérateurs AR.
Pour cela, on va prendre comme exemple une petite base *cinema.sql* dont le schéma est :

Artiste(Nom,Prenom,Annee_Naissance)
Cinema(Nom_cinema,Arrondissement,Adresse)
Role(Nom_Role,ID_film,Nom_Acteur)
Salle(Nom_cinema, No_salle, Climatise,Capacite)
Seance(Nom_cinema, No_salle, No_seance,Heure_debut,Heure_fin, ID_film)
Film(Nom_Realisateur, Titre, ID_film, Annee).

# Introduction à SQL

Le format général d'une requête SQL est :

**SELECT** `liste d'atributs`
**FROM** `liste de noms de tables`
**WHERE** `condition booléenne sur les` *n*`-plets`

La partie **WHERE** est optionnelle

SIGNIFICATION ???

Projection sur une table

**SELECT** Nom_Realisateur, Titre
**FROM** Film
exprime :
*Quels sont les couples* $\langle r, f \rangle$ *tel que* $r$ *est le nom du réalisateur du film* $f$? Ou encore : Donner les nom des réalisateurs, avec leur films

En AR : $\pi_{Nom\_Realisateur, Titre}(Film)$

Le SELECT exprime l'operateur de projection $\pi$ , et apres le FROM on a la table argument de l'opérateur

## Projection sur une table

## Un autre exemple

**SELECT** nom, prenom
**FROM** Artiste
exprime :
*Quels sont les noms et les prénom des artistes ?*

En AR : $\pi_{Nom,Prenom}(Artiste)$

Ajutons un WHERE : operateur de séléction

**SELECT** Nom, Prenom
**FROM** Artiste
**WHERE** Annee_Naissance = 1950
exprime :
*Quels sont les noms et les prénom des artistes nés en 1950 ?*
En AR : $\sigma_{Annee\_Naissance=1950}(\pi_{Nom,Prenom}(Artiste))$, ou encore :

$$\sigma_{Annee\_Naissance=1950}\ \pi_{Nom,Prenom}(Artiste)$$

WARNING : le SELECT est une projection algébrique, pas un $\sigma$ de l'AR. Attention aux faux amis. Le $\sigma$ de l'AR est exprimé par le SELECT.

## UN FROM plus complexe

**SELECT** Titre, Nom_Cinema
**FROM** Film, Salle
exprime : *Donner les couples* $\langle titre, nc \rangle$ *où titre est le titre d'un film, et nc est un nom de cinéma répértorié dans la table Salle.* On ne demande pas que le film soit projecté dans la salle en question.

En AR : $\pi_{Titre,Nom\_Cinema}(Film \times Cinema)$

Si, après le FROM, on a $r_1, ..., r_n$, la signification implicite de la virgule est le produit cartesien des tables $r_i$

UN FROM encore plus complexe

Ici, on veut savoir :

*Donner les cinémas qui commencent à projecter un film à 14.00 dans une salle climatisée.*

Il faut travailler sur Seance et Salle

mais

Seance(Nom_cinema, No_salle, No_seance,Heure_debut,Heure_fin),

et

Salle(Nom_cinema, No_salle, Climatise,Capacite)

partagent les attribut Nom_cinema et.

Comment faire avec le FROM, qui s'attend le produit cartésian des ses argument ?

UN FROM encore plus complexe, suite

Une solution

**SELECT** Seance.Nom_cinema,
**FROM** Seance, Salle
**WHERE** Seance.Nom_cinema = Salle.Nom_cinema
**AND** Seance.No_salle = Salle.No_salle **AND** Climatise = True
**AND** Heure_debut = 14.00 **AND** Salle.Nom_cinema = 'Rex';

En AR : Soit

$R_1 = \rho_{Nom\_cin \rightsquigarrow Seance.Nom\_cin}\ (\rho_{No\_sa \rightsquigarrow Seance.No\_sa}(Seance))$

$R_2 = \rho_{Nom\_cin \rightsquigarrow Salle.Nom\_cin}\ (\rho_{No\_sa \rightsquigarrow sa.No\_sa}(Salle))$

$R3 = R_1 \times R_2$

$C = Seance.Nom\_cin = Salle.Nom\_cin \wedge Seance.No\_sa = Salle.No\_sa \wedge Clim = True \wedge Heure\_de = 14.00 \wedge Salle.Nom\_cin = ' Rex'$

On a exprimé : $\pi_{Seance.Nom\_cin}\ \sigma_C(R3)$

UN FROM encore plus complexe, suite

La requête SQL précédente a aussi une écriture AR qui utilise une jointure naturelle $\bowtie$.

Laquelle ?

## UN FROM encore plus complexe, suite
## Solution 2

**SELECT** se.Nom_cinema,
**FROM** Seance se, Salle sa
**WHERE** se.Nom_cinema = sa.Nom_cinema
**AND** se.No_salle = sa.No_salle **AND** Climatise = True **AND**
Heure_debut = 14.00 **AND** sa.Nom_cinema = 'Rex';

Ici, on renomme carrement les tables : Seance devient se, Salle devient sa.

## Travailler sur plusieurs copies de la même table

La solution 2 suggère comment on peut travailler, en SQL, avec 2 copies de la même table.
Soient : AimeLivre(Pers, NomLivre) et Livre(NomLivre).
On avait vu que la requête "Qui aime au moins deux livres ?"
se dit, en AR :

$$\pi_{Pers}(\sigma_{NomLivre \neq NomLivre2}(AimeLivre \bowtie \rho_{NomLivre \rightsquigarrow NomLivre2}(AimeLivre)))$$

En SQL :
**SELECT** a1.Pers
**FROM** AimeLivre a1, AimeLivrea2
**WHERE** a1.NomLivre tt != a2.NomLivre;

# La différence algébrique, et les sous-requêtes, 1

*Quels identifiants de film, répertoriés dans la table Film, ne sont pas repertoriés dans la table Seance ?*

Solution 1

**SELECT** ID_film
**FROM** Film
**MINUS**
**SELECT** ID_film
**FROM** Seance

OK seulement car les deux tables arguments du **MINUS** ont le même schéma

# La différence algébrique, et les sous-requêtes, 2

*Quels identifiants de film, répertoriés dans la table Film, ne sont pas repertoriés dans la table Seance ?*

Solution 2

**SELECT** ID_film
**FROM** Film
**WHERE** ID_film **NOT IN**
**SELECT** ID_film
**FROM** Seance

Construction acceptable même si les 2 tables en question n'avaient pas eu le même schéma... Voir diapositive suivante

# La différence algébrique, et les sous-requêtes, 2

*Quels noms d'artistes ne sont pas des noms d'acteurs (repértoriés dans la table Role) ?*

Solution 2

**SELECT** Nom
**FROM** Artiste
**WHERE** Nom **NOT IN**
**SELECT** Nom_Acteur
**FROM** Role

Nom et Nom_Acteur sont deux attributs différents

# L'intersection algébrique

*Quels identifiants de film, répertoriés dans la table Film, sont aussi repertoriés dans la table Seance ?*

Solution 2

**SELECT** ID_film
**FROM** Film
**WHERE** ID_film **IN**
**SELECT** ID_film
**FROM** Seance

# L'union algébrique

*Quels titres de films ont paru en 2000 ou bien en 2012 ?*

**SELECT** Titre
**FROM** Film
**WHERE** Annee = 2012
**UNION**
**SELECT** Titre
**FROM** Film
**WHERE** Annee = 2014

ou encore :
**SELECT** Titre
**FROM** Film
**WHERE** Annee = 2012 **OR** Annee = 2014

# Le quantificateur **EXISTS**, 1

**EXISTS** est très utile, mais il n'a pas de correspondance directe en AR. Il vient plutôt du Calcul Relationnel.
Si T est une table résultat d'une requête, **EXISTS**(T) donne la valeur True si T n'est pas vide, fausse sinon.

*Quels réalisateurs n'ont réalisé aucun film après 2000 ?*

```
SELECT t1.Nom_Realisateur
FROM Film t1
WHERE NOT EXISTS
    (SELECT t2.Annee
    FROM Film t2
    WHERE
    t1.Nom_Realisateur = t2.Nom_Realisateur AND t2.Annee > 2000);
```

Pour comprendre : voir t1 et t2 comme des VARIABLES qui balayent Film : au tableau.

# Le quantificateur **EXISTS**, 2

```
SELECT t1.Nom_Realisateur
FROM Film t1
WHERE NOT EXISTS
    (SELECT t2.Annee
    FROM Film t2
    WHERE
    t1.Nom_Realisateur = t2.Nom_Realisateur AND t2.Annee > 2000);
```

En AR ?
Soit $R1 = \pi_{Nom\_Realisateur}(Film)$ = tous les réalisateurs
Soit $R2 = \pi_{Nom\_Realisateur}\sigma_{Annee>2000}(Film)$ = ces réalisateurs ayant réalisé au moins un film après 2000.
Alors : $R1 \setminus R2$.

# La division algébrique

Reprennons l'exemple AR : AimeLivre(Pers, NomLivre), Livre(Nomlivre)
*Qui aime tous les livres ?* : $AimeLivre \div Livre$

```
SELECT a1.Pers
FROM
AimeLivre a1
WHERE
NOT EXISTS (
SELECT * FROM Livre li WHERE
NOT EXISTS (
SELECT * FROM AimeLivre a2 WHERE
a2.pers = a1.pers AND a2.NomLivre = li.NomLivre ) )
```

*Le personne cherchées sont ces p qui sont la valeur a1.Pers pour quelque ligne a1 de AimeLivre et pour lesquelles il n'existe pas de livre li tel il n'existe pas de ligne a2 $\langle p, li \rangle$ dans AimeLivre.*