Systèmes d'Information - Bases de Données Relationnelles[1]

Cheikh Ba

[1] Inspiré de *Cours de bases de données*, Philippe Rigaux, 2001

Plan

Le modèle Entité/Association

Ce chapitre présente le modèle E/A, utilisé à peu près universellement pour la conception de Bases de Données (relationnelles principalement). La conception d'un schéma correct est essentielle pour le développement d'une application viable. Dans la mesure où la base de données est le fondement de tout le système, une erreur pendant sa conception est difficilement récupérable par la suite. Le modèle E/A est simple et suffisamment puissant pour représenter des structures relationnelles. Surtout, il repose sur une représentation graphique qui facilite sa compréhension.

Principes:

- Distinguer les *entités* qui constituent la BD.
- Distinguer les *associations* entre ces *entités*.

Principes généraux

Motivations

- Table *FilmSimple*: films avec informations sur les metteurs en scène.

Titre	**Année**	**NomMES**	**PrénomMES**	**AnnéeNaiss**
Alien	1979	Scott	Ridley	1943
Vertigo	1958	Hitchcock	Alfred	1899
Psychose	1960	Hitchcock	Alfred	1899
Kagemusha	1980	Kurosawa	Akira	1910
Volte-face	1997	Woo	John	1946
Pulp Fiction	1995	Tarantino	Quentin	1963
Titanic	1997	Cameron	James	1954
Sacrifice	1986	Tarkovski	Andrei	1932

- Problème: **redondance** de l'information $\Rightarrow$ Anomalies ...

Principes généraux

Motivations

Titre	Année	NomMES	PrénomMES	AnnéeNaiss
Alien	1979	Scott	Ridley	1943
Vertigo	1958	Hitchcock	Alfred	1899
Psychose	1960	Hitchcock	Alfred	1899
Kagemusha	1980	Kurosawa	Akira	1910
Volte-face	1997	Woo	John	1946
Pulp Fiction	1995	Tarantino	Quentin	1963
Titanic	1997	Cameron	James	1954
Sacrifice	1986	Tarkovski	Andrei	1932

Anomalies:

- Lors d'une insertion:
  - Rien n'empêche de représenter plusieurs fois le même film.
  - Qu'est-ce qui différencie un film d'un autre ?
- Lors d'une modification:
  - Mise à jour partielle (**Hitchcock** par ex.) $\Rightarrow$ informations incohérentes.
- Lors d'une destruction:
  - Suppression de **Titanic** $\Rightarrow$ Suppression de **James Cameron** !

Principes généraux

La bonne méthode

Étapes:

1. Représenter individuellement les films et les réalisateurs, de manière à ce qu'une action sur l'un n'entraîne pas systématiquement une action sur l'autre ;
2. Définir une méthode d'identification d'un film ou d'un réalisateur, qui permette d'assurer que la même information est représentée une seule fois ;
3. Préserver le lien entre les films et les réalisateurs, mais sans introduire de redondance.

Principes généraux

La bonne méthode

- 2 premières étapes: représentation individuelle et identification
  - Décision: 2 films ne peuvent pas avoir le même titre. Deux réalisateurs peuvent avoir le même nom.

Titre	Année
Alien	1979
Vertigo	1958
Psychose	1960
Kagemusha	1980
Volte-face	1997
Pulp Fiction	1995
Titanic	1997
Sacrifice	1986

id	NomMES	PrénomMES	AnnéeNaiss
1	Scott	Ridley	1943
2	Hitchcock	Alfred	1899
3	Kurosawa	Akira	1910
4	Woo	John	1946
5	Tarantino	Quentin	1963
6	Cameron	James	1954
7	Tarkovski	Andrei	1932

- Clé **id** pour identifier les réalisateurs.
- Premier progrès : il n'y a maintenant plus de redondance dans la BD.
  - **Hitchcock** représenté qu'une seule fois.

Principes généraux

La bonne méthode

- Étape 3: liens entre films et metteurs en scène
  - Associer l'identifiant du metteur en scène au film
  - Déplacement de la clé **id** de **MES** vers la **Films**: clé étrangère.

Titre	Année	IdMES
Alien	1979	1
Vertigo	1958	2
Psychose	1960	2
Kagemusha	1980	3
Volte-face	1997	4
Pulp Fiction	1995	5
Titanic	1997	6
Sacrifice	1986	7

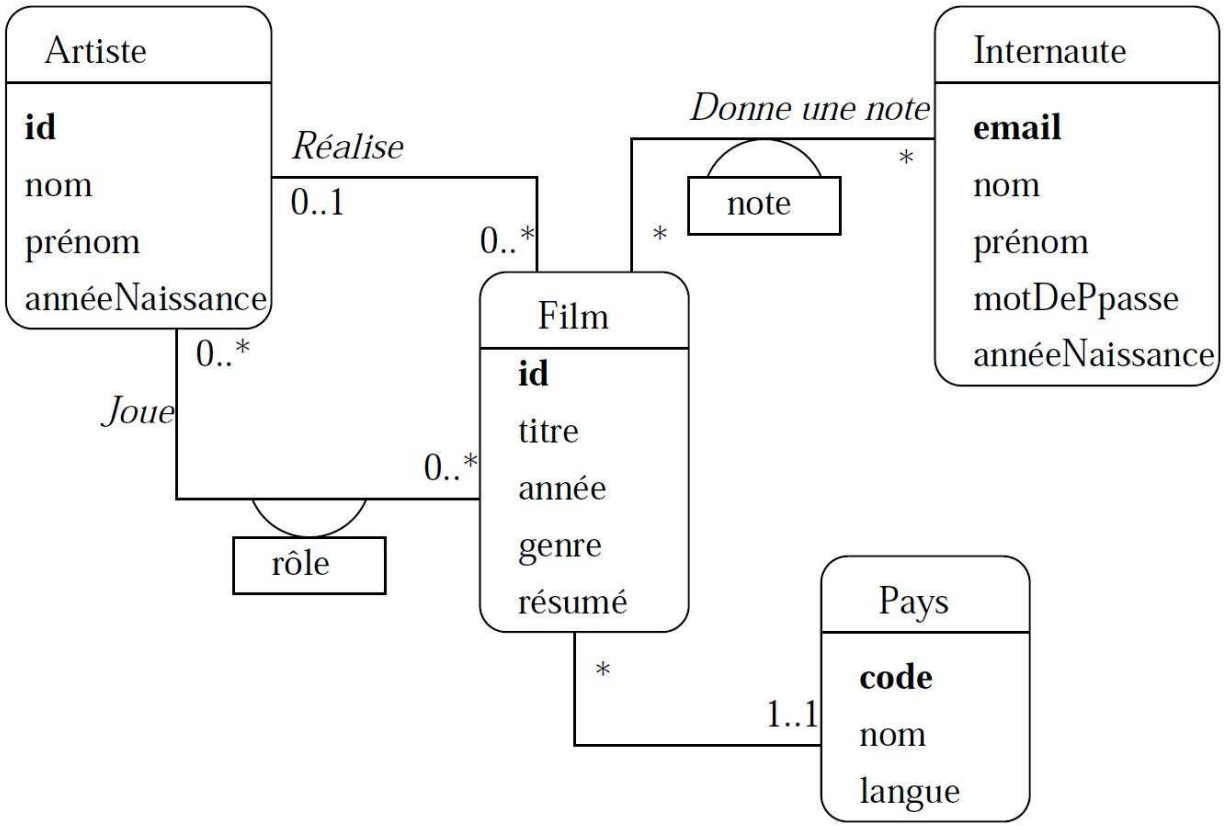
id	NomMES	PrénomMES	AnnéeNaiss
1	Scott	Ridley	1943
2	Hitchcock	Alfred	1899
3	Kurosawa	Akira	1910
4	Woo	John	1946
5	Tarantino	Quentin	1963
6	Cameron	James	1954
7	Tarkovski	Andrei	1932

- Les anomalies ont-elles disparu ? Y-a-t-il eu perte d'information ?

- La modélisation E/A offre une méthode pour arriver au m̂ résultat.

Le modèle E/A: Présentation informelle

- Abstraction d'un domaine d'étude
  - Choix, en fonction des besoins, de certains aspects de la réalité perçue.
  - **Entités**, caractérisées par des *attributs*, dont une ou plusieurs clés.
  - **Associations**, caractérisées par des *cardinalités*[2] (*min* .. *max*).



[2]Choix essentiel, parfois discutable. Aspect plus délicat de la modélisation

Le modèle E/A: Présentation informelle

Remarques

- La modélisation est totalement indépendante de tout choix d'implémentation
  - Partie la plus stable de l'application
  - On se concentre sur l'essentiel. Que veut-on faire ? et pas comment ?
- Le modèle E/A a été conçu en 1976 et est à la base de la plupart des méthodes de conception.
- La syntaxe utilisée ici est celle d'UML
  - Beaucoup d'autres notations: OMT, MERISE, ...
  - Notations globalement équivalentes, et reposent tous sur deux concepts complémentaires: *entité* et *association*

Le modèle E/A

Entités, attributs et identifiants

Entité: Tout objet identifiable et pertinent pour l'application.

Exemple: le film *Alien* et l'acteur *James Cameron* sont des entités.

- Notion d'*identité* primordiale
  - Distinguer les entités les unes des autres (gestion des redondances).
  - Moyen pour effectuer cette distinction: *identifiant* ou *clé*.

- Il faut regrouper les entités en ensembles.
  - En général, on ne s'intéresse pas à l'individu mais au groupe.
  - Exemple: les films et les acteurs sont des ensembles distincts d'entités.

  - Qu'en est il de l'ensemble des réalisateurs et de celui des acteurs ? Doit-on les distinguer ou les assembler ?[3]

[3] Il est préférable des les assembler puisque des acteurs peuvent aussi être réalisateurs

Le modèle E/A

Entités, attributs et identifiants

Attributs: propriétés qui caractérisent les entités

Exemple: le titre (du film), le nom (de l'acteur), sa date de naissance, etc.

- Désigné par un *nom* et prend ses valeurs dans un domaine énumérable comme les entiers, les chaînes de caractères, les dates, etc.
- Un nom d'attribut A peut être considéré comme une fonction $A : E \mapsto D$, E est un ensemble d'entités, D est le domaine de valeurs.
- Exemple:
  - Ensemble de films $E = \{f_1, f_2, \cdots, f_n\}$. Attributs *titre* et *année*.
  - Si f_1 est le film *Titanic*, tourné par *James Cameron* en 1997, alors:

$$titre(f_1) = Titanic; \qquad annee(f_1) = 1997$$

- Définition ne prenant pas en compte des *attributs multivalués* (ensembles) et *composés* (structures)

Le modèle E/A

Entités, attributs et identifiants

Types d'entité: Composé des éléments suivants:

1. Nom
2. Liste des attributs avec (optionnellement) les domaines de valeurs.
3. L'identification des attributs permettant d'identifier l'entité: la CLÉ.

- On dit qu'une entité e est une *instance* de son type T.
- Un ensemble d'entités $\{e_1, e_2, \cdots e_n\}$, instances d'un même type T est une *extension* de T.

Le modèle E/A

Entités, attributs et identifiants

Clé: 1/3

Soit T un type d'entité et A l'ensemble des attributs de T. Un clé de T est un sous-ensemble minimal de A permettant d'identifier de manière unique une entité parmi n'importe quelle extension de T.

Exemple du type **Internaute**. Attributs: email, nom, prénom, région

- L'email constitue une clé naturelle. Pourquoi ?
- L'identification par le nom paraît impossible. Pourquoi ?
- On peut penser au couple (nom, prénom).
  - Utiliser avec modération les id composés de plusieurs attributs.
  - Peut poser des problèmes de performance et complique les manipulations par SQL.

Le modèle E/A

Entités, attributs et identifiants

Clé: 2/3

Soit T un type d'entité et A l'ensemble des attributs de T. Un clé de T est un sous-ensemble minimal de A permettant d'identifier de manière unique une entité parmi n'importe quelle extension de T.

- Il est possible d'avoir **plusieurs clés** (candidates) pour une m̂ entité
- Une comme *clé primaire*. Les autres comme *clés secondaires*.
- Choix de la *clé primaire* déterminant pour la qualité du schéma.

- Caractéristiques d'une bonne clé primaire:
  - Sa valeur est connue pour toute entité
  - On ne doit jamais avoir besoin de la modifier.
  - Taille de stockage la plus petite possible (performance).

Le modèle E/A

Entités, attributs et identifiants

Clé: 3/3

Soit T un type d'entité et A l'ensemble des attributs de T. Un clé de T est un sous-ensemble minimal de A permettant d'identifier de manière unique une entité parmi n'importe quelle extension de T.

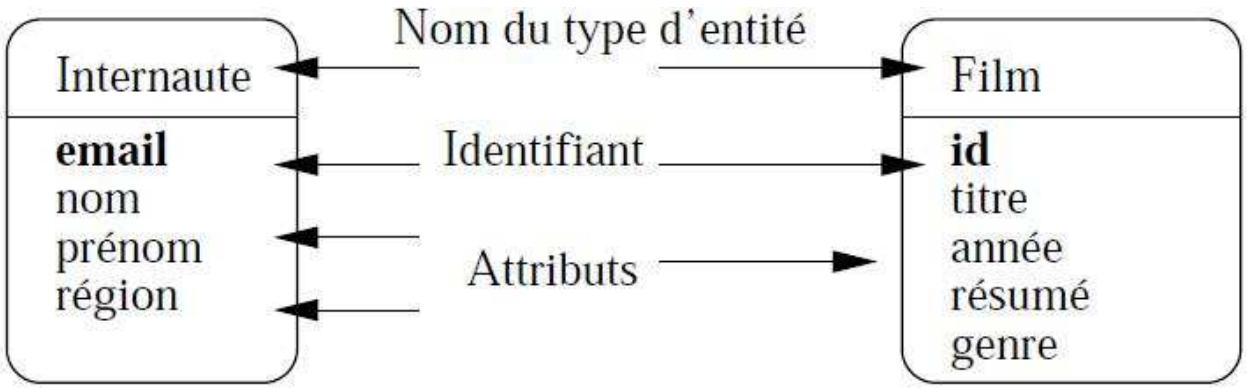
- Pas toujours évident de trouver un tel ensemble d'attributs.
- Considérons l'exemple des films.
  - Choix du titre comme id ? Incorrect (*Planète des singes*)
  - Combinaison du titre et l'année ? Difficile de garantir l'unicité.

- Dans ce genre de situation (fréquente):
  - Création d'un id **abstrait**, indépendant de tout autre attribut.
  - Exemple: ajouter dans le type d'entité Film un attribut **id**.
    - ★ Souvent un numéro séquentiel qui sera incrémenté à chaque insertion.

Le modèle E/A

Entités, attributs et identifiants

Représentation graphique d'un type d'entité



- L'attribut (ou les attributs) formant la **clé** sont en gras.

- Il faut bien distinguer *types d'entités* et *entités*
  - *types d'entités* $\neq$ *entités*
  - *schéma de la base de données* $\neq$ *Base de données* (SGBD)
  - *type* $\neq$ *valeur* (langage de programmation)

Le modèle E/A

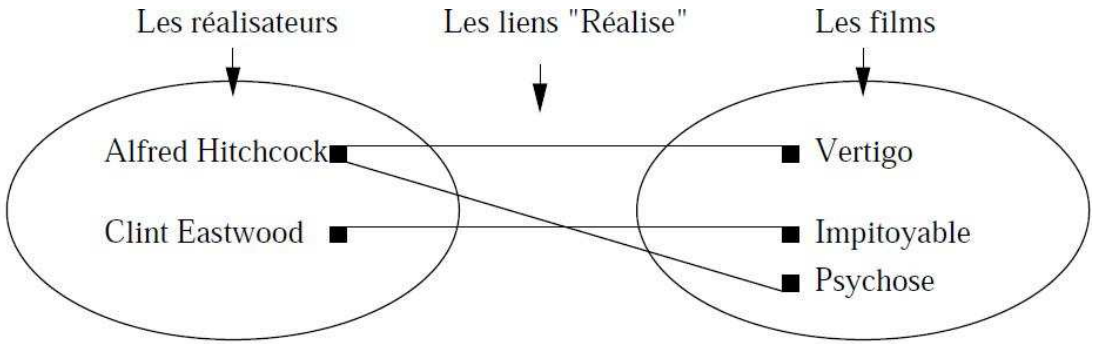
Associations binaires

- Représentation et stockage d'entités indépendantes: Très peu d'utilité.

Association binaire

Une association binaire entre les ensembles d'entités E_1 et E_2 est un ensemble de couples (e_1, e_2), avec $e_1 \in E_1$ et $e_2 \in E_2$.

- Notion classique de relation en théorie des ensembles.
- Terme "Association": éviter une confusion avec le Modèle Relationnel



- Sensibilisation au bon choix des cardinalités.
  - Situations les plus générales possible.

Le modèle E/A

Associations binaires

Cardinalité

Soit une association (E_1, E_2) entre deux types d'entités. La cardinalité de l'association pour $E_i, i \in \{1, 2\}$ est une paire [**min**, **max**] telle que:

- **max**: Cardinalité maximale - Nombre maximal de fois où une entité e_i de E_i peut intervenir dans l'association. En général ce nombre est:
  - **1**: Au plus une fois. Ou
  - **n**: Plusieurs fois, nombre indéterminé, noté par le symbole $*$
- **min**: Cardinalité minimale - Nombre minimal de fois où une entité e_i de E_i peut intervenir dans l'association. En général ce nombre est:
  - **1**: Au moins une fois. Ou
  - **0**: ?

Le modèle E/A

Associations binaires

Cardinalité[a]

[a]Les cardinalités n'expriment pas une vérité absolue, mais des choix de conception.

- Cardinalités **max** plus importantes:
  - Beaucoup plus difficile à remettre en cause après constitution de la BD.
- Cardinalités **min** parfois désignées par "contraintes de participation":
  - 0: une entité peut ne pas participer. 1: elle doit y participer.

- Notation abrégée (omission des cardinalités minimales en UML):
  - $* \Leftrightarrow \mathbf{0..*}$
  - $\mathbf{1} \Leftrightarrow \mathbf{1..1}$

  - Caractérisation d'une association en donnant les **max** aux 2 extrémités
  - $\mathbf{1 : n} \Leftrightarrow$ Association de un à plusieurs
  - $\mathbf{n : n} \Leftrightarrow$ Association de plusieurs à plusieurs

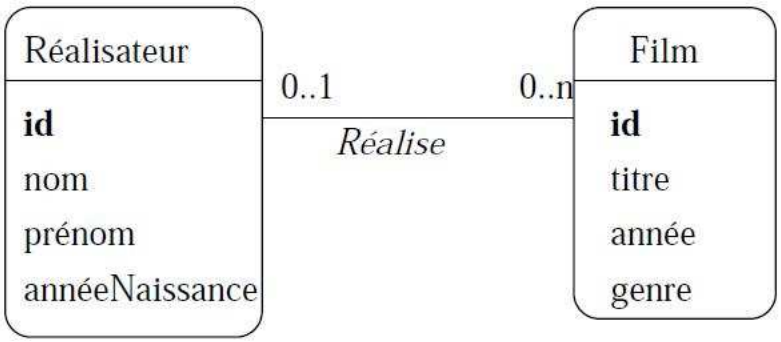
Le modèle E/A

Associations binaires

Cardinalité

- Comment lire les cardinalités (UML) ?

- Cardinalités aux extrémités d'un lien d'association entre T_A et T_B
  - Cardinalités pour T_A placées à l'extrémité du lien $T_A \rightarrow T_B$
  - Cardinalités pour T_B placées à l'extrémité du lien $T_B \rightarrow T_A$



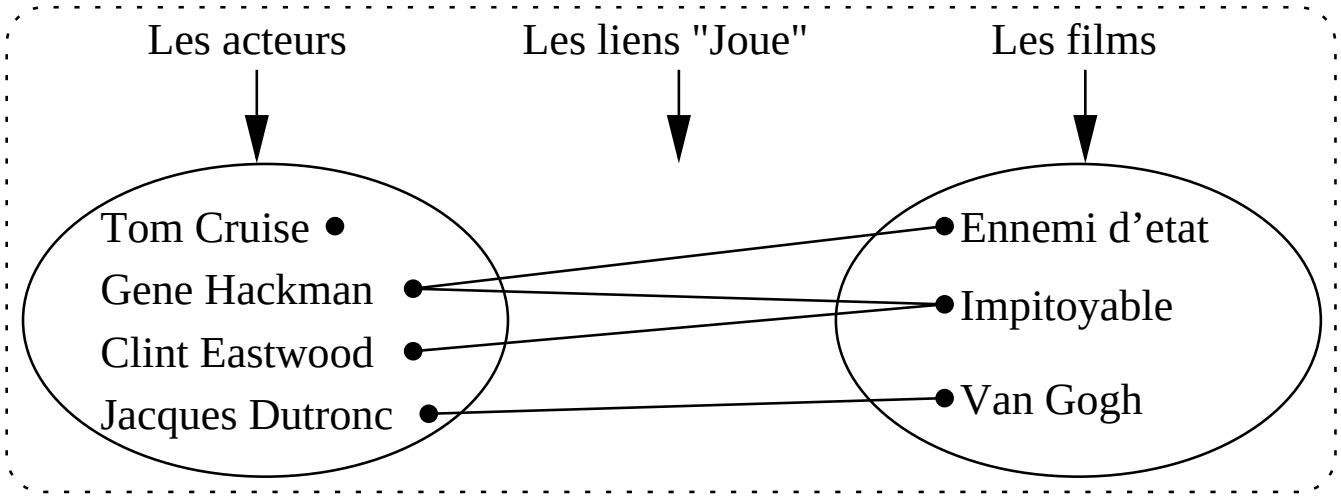
  - *Un réalisateur réalise zéro, un ou plusieurs films*
  - Possibilité d'utiliser la forme passive: intitulé *Est réalisé par*
  - *Un film est réalisé par au plus un réalisateur*

Le modèle E/A

Associations binaires

Liens multiples entre deux types d'entités: 1/3

- Exemple: Relation *Acteur* $\leftrightarrow$ *Film*
  - Graphe basé sur quelques exemples.

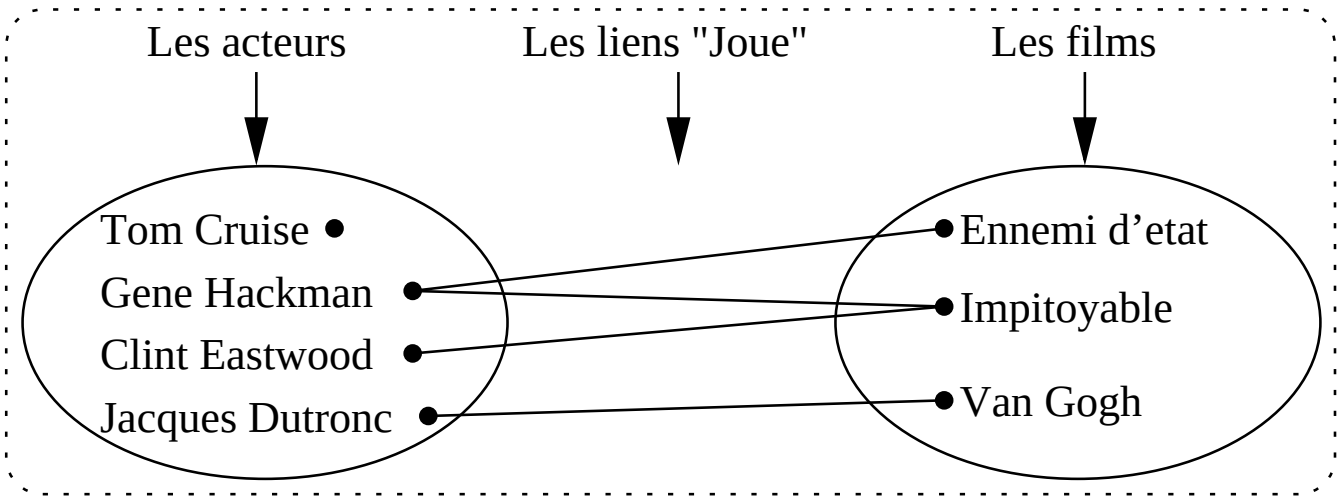


  - Un acteur peut jouer dans plusieurs films
  - Dans un film on trouve plusieurs acteurs

Le modèle E/A

Associations binaires

Liens multiples entre deux types d'entités: 2/3



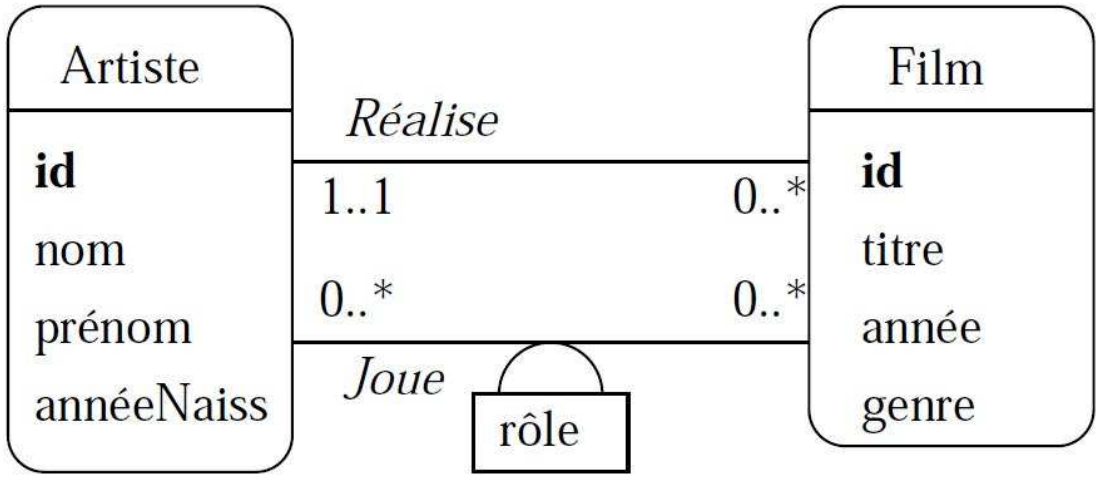
- Mieux: **Clint Eastwood**, metteur en scène, est maintenant acteur, et dans le même film.
  - Regrouper les acteurs et les réalisateurs dans un m̂ ensemble: *Artiste*
  - "OU" non exclusif: **C. Eastwood** joue dans *Impitoyable* qu'il a réalisé.

Le modèle E/A

Associations binaires

Liens multiples entre deux types d'entités: 3/3

- Associations entre *Artiste* et *Film*.



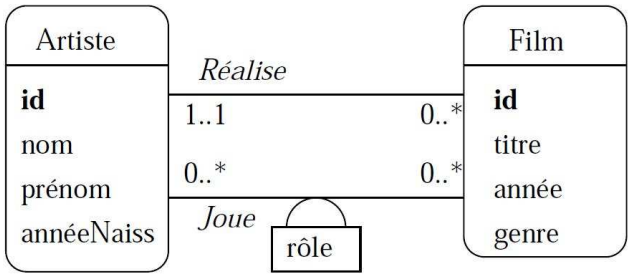
Le modèle E/A

Associations binaires

Attribut d'une association

- Attribut[a] *rôle*: rôle tenu par l'acteur dans le film.

[a]Rappel: un attribut ne peut prendre qu'une et une seule valeur



- L'attribut *rôle* peut-il être affecté à *Acteur* (*Artiste*)?
  - Non. Aurait autant de valeurs que de films dans lesquels l'acteur a joué
- L'attribut *rôle* peut-il être affecté à *Film* ? Non. Pourquoi ?
- On fait donc porter l'attribut *rôle* à l'association[4] *Joue*.

[4]Seules les associations ?..∗ ↔ ?..∗ peuvent porter des attributs

Le modèle E/A

Associations binaires

Clé d'une association[a] 1/2

[a] Rappel: association = ensemble de couples ⇒ pas de doublon

La clé d'une association (binaire) entre un type d'entité E_1 et un type d'entité E_2 est le couple constitué de la clé c_1 de E_1 et de la clé c_2 de E_2.

Définition souvent trop contraignante dans les cas où on souhaite autoriser deux entités à être liées plus d'une fois dans une association

- **Exemple:** Un internaute voulant noter plusieurs fois un même film et conserver l'historique des ces notations successives.

- Solution: ajout d'une entité discriminante - Entité *Date* par exemple.
- **Plus simple:** ajout d'un attribut discriminant à l'association *Noter*
  - Attribut *date*. Fera partie de la clé de l'association (en plus de c_1 et c_2).

Le modèle E/A

Associations binaires

Clé d'une association[a] 1/2

[a] Rappel: association = ensemble de couples ⇒ pas de doublon

La clé d'une association (binaire) entre un type d'entité E_1 et un type d'entité E_2 est le couple constitué de la clé c_1 de E_1 et de la clé c_2 de E_2.

Définition souvent trop contraignante dans les cas où on souhaite autoriser deux entités à être liées plus d'une fois dans une association

- **Exemple:** Un internaute voulant noter plusieurs fois un même film et conserver l'historique des ces notations successives.

- Solution: ajout d'une entité discriminante - Entité *Date* par exemple.
- **Plus simple:** ajout d'un attribut discriminant à l'association *Noter*
  - Attribut *date*. Fera partie de la clé de l'association (en plus de c_1 et c_2).

Le modèle E/A

Associations binaires

Clé d'une association[a] 1/2

[a] Rappel: association = ensemble de couples ⇒ pas de doublon

La clé d'une association (binaire) entre un type d'entité E_1 et un type d'entité E_2 est le couple constitué de la clé c_1 de E_1 et de la clé c_2 de E_2.

Définition souvent trop contraignante dans les cas où on souhaite autoriser deux entités à être liées plus d'une fois dans une association

- **Exemple:** Un internaute voulant noter plusieurs fois un même film et conserver l'historique des ces notations successives.

- Solution: ajout d'une entité discriminante - Entité *Date* par exemple.
- **Plus simple:** ajout d'un attribut discriminant à l'association *Noter*
  - Attribut *date*. Fera partie de la clé de l'association (en plus de c_1 et c_2).

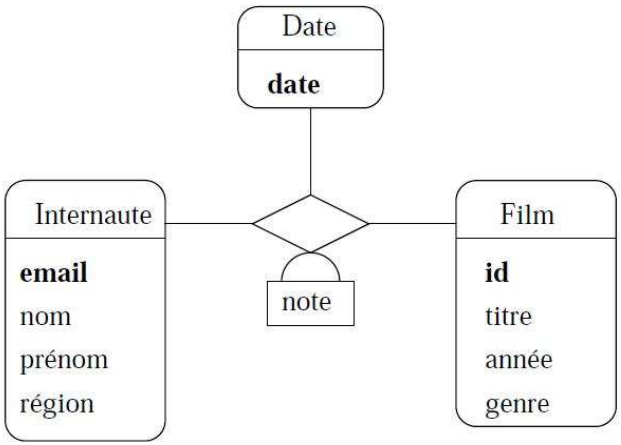
Le modèle E/A

Associations binaires

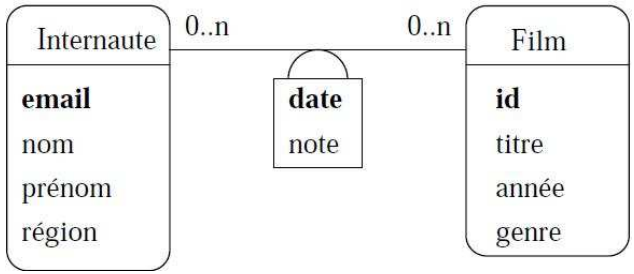
Clé d'une association 2/2

- Sol. 1: ajout d'une entité discriminante - Entité *Date* par exemple.

Sol. 1



Sol. 2



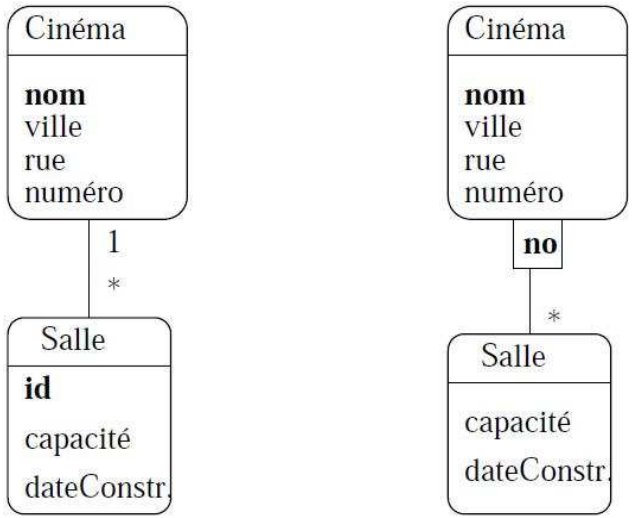
- Sol. 2 (Mieux): ajout d'un attribut discriminant à l'association *Noter*

Le modèle E/A

Entités faibles (associations $\mathbf{1} \leftrightarrow *$)

Cas où une entité ne peut exister qu'en étroite association avec une autre.

- Exemple: Un cinéma et ses salles
  - Difficile de représenter une salle sans la rattacher à son cinéma.



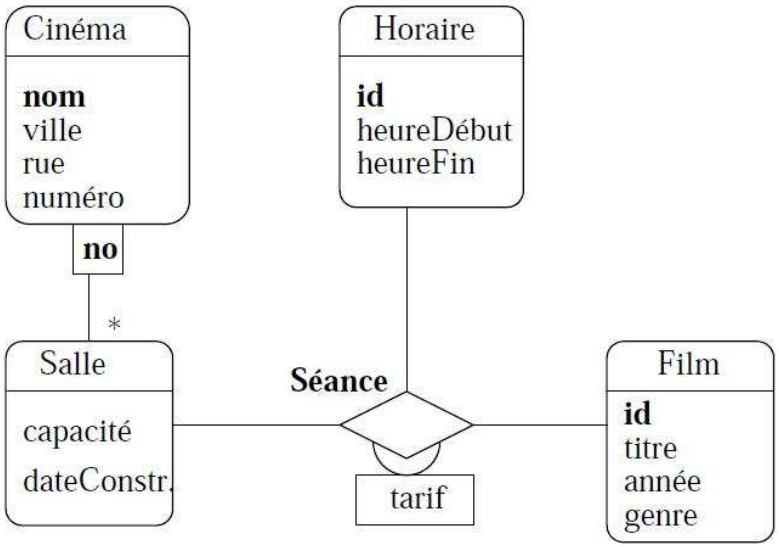
a) Modélisation sous la forme d'une association classique: clé(Salle) = id.
b) Modélisation sous la forme d'une entité faible: clé(Salle) = (nom, no). Meilleure représentation ?

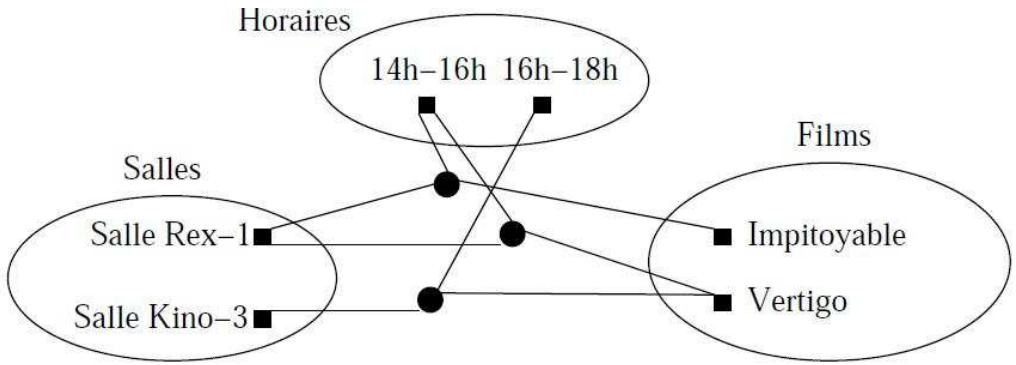
Le modèle E/A

Associations généralisées - (entre **n** entités)

Association *n*-aire

Une association *n*-aire entre *n* types d'entités $E_1, E_2, \cdots, E_n$ est un ensemble de *n*-uplets $(e_1, e_2, \cdots, e_n)$, avec $e_i \in E_i$.





- Principal problème: les cardinalités sont implicitement **0**..*
  - **!** Salle *Rex-1*: Deux films en même temps !!

Le modèle E/A

Associations généralisées - (entre **n** entités)

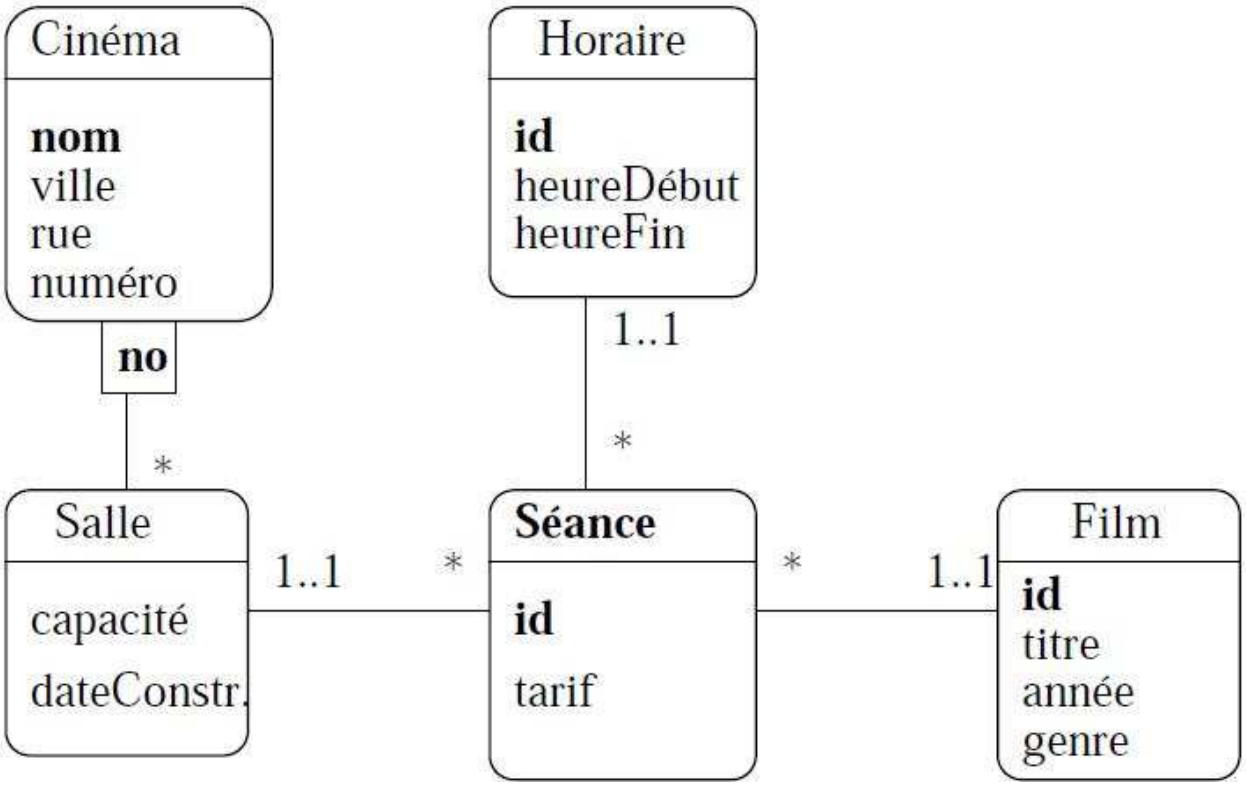
- Association de degré supérieur à 2 difficiles à manipuler et interpréter.
- Toujours possible de la remplacer par un type d'entité:

Soit A une association entre les types d'entités $\{E_1, E_2, \cdots, E_n\}$. La transformation de A en type d'entité s'effectue en trois étapes:

- On attribue un identifiant autonome à A.
- On crée une association A_i de type **1** : **n** entre A et chacun des E_i.
  - La contrainte minimal du coté de A est toujours à **1**.

Le modèle E/A

Associations généralisées - (entre **n** entités)



- Le problème (2 films en même temps dans *Rex-1*) a-t-il disparu ?

Le modèle E/A

Avantages et inconvénients du modèle E/A

- Le modèle E/A est simple et pratique
  - Il n'y a que 3 concepts: *entités*, *associations* et *attributs*
  - Approprié à une représentation graphique intuitive.
  - Permet de modéliser rapidement des structures pas trop complexes.

- Inconvénients
  - Modèle non déterministe: pas de règle absolue pour déterminer ce qui est *entité*, *attribut* ou *relation*.
  - Pauvreté: difficile d'exprimer des contraintes d'intégrité.

Plan

Le modèle Relationnel

Rappels

Un **modèle de données** définit un mode de représentation de l'information selon 3 composantes:

1. Des **structures de données**.
2. Des **contraintes**: spécifier les règles que doit respecter une BD.
3. Des **opérations** pour manipuler les données: interrogation et M-à-J.

- Points 1 et 2 relèvent du *Langage de Définition de Données (***DDL***)*
  - Description du **schéma** de la BD: structure de données + contraintes.
- Points 3 est la base du *Langage de Manipulation de Données (***DML***)*
  - Opérations: SQL est le représentant le plus célèbre.

- Le modèle relationnel offre une **totale indépendance** entre
  - La représentation logique
  - Et l'implémentation physique.

Le modèle Relationnel

Définition d'un **schéma relationnel**

- Modèle très simple: Il n'existe qu'une seule structure: la **Relation**
  - Une relation peut se représenter par un **table**.
  - Exemple: *Film (titre: string, année: number, genre, string)*

Domaines de valeurs:

- Ensemble d'instances d'un type élémentaire.
- Ex: entiers, réels, chaînes de caractères, etc.

Attributs:

- Nomme les **colonnes** d'une relation
- Toujours associé à un domaine

Titre	Année	Genre
Alien	1979	Science-Fiction
Vertigo	1958	Suspense
Volte-face	1997	Thriller
Pulp Fiction	1995	Policier

Relation **Film**

Schéma de relation:

- Un nom suivi de la liste des attributs
- Syntaxe: $R(A_1 : D_1, A_2 : D_2, \cdots, A_n : D_n)$
- **Arité:** nombre d'attributs de la relation.

Le modèle Relationnel

Définition d'un **schéma relationnel**

- **Relation** (ou instance d'une relation) R
  - Sous-ensemble fini du produit cartésien des domaines des attributs de R
  - Produit cartésien $D_1 \times \cdots \times D_n = \{(v_1, \cdots, v_n),\ v_i \in D_i\}$
  - Une relation est donc un ensemble
    - L'ordre des lignes n'a pas d'importance
    - Pas deux lignes identiques
    - Pas de "case vide" dans la table (pas toujours dans la pratique)

Clé d'une relation :

- Plus petit sous-ensemble permettant d'identifier chaque ligne de manière unique.
- Ex: `Film (Titre, Année, Genre)`

Tuple (ou **n-uplet**) :

- Liste de n valeurs $(v_1, \cdots, v_n)$ où chaque v_i est la valeur d'un attribut A_i de domaine D_i.
- Ex: `('Volte-face', 1997, 'Thriller')`

Titre	Année	Genre
Alien	1979	Science-Fiction
Vertigo	1958	Suspense
Volte-face	1997	Thriller
Pulp Fiction	1995	Policier

Relation **Film**

Le modèle Relationnel

Définition d'un **schéma relationnel**

- **Base de données** (ou instance de base de données)
  - Ensemble fini (d'instances) de relations.

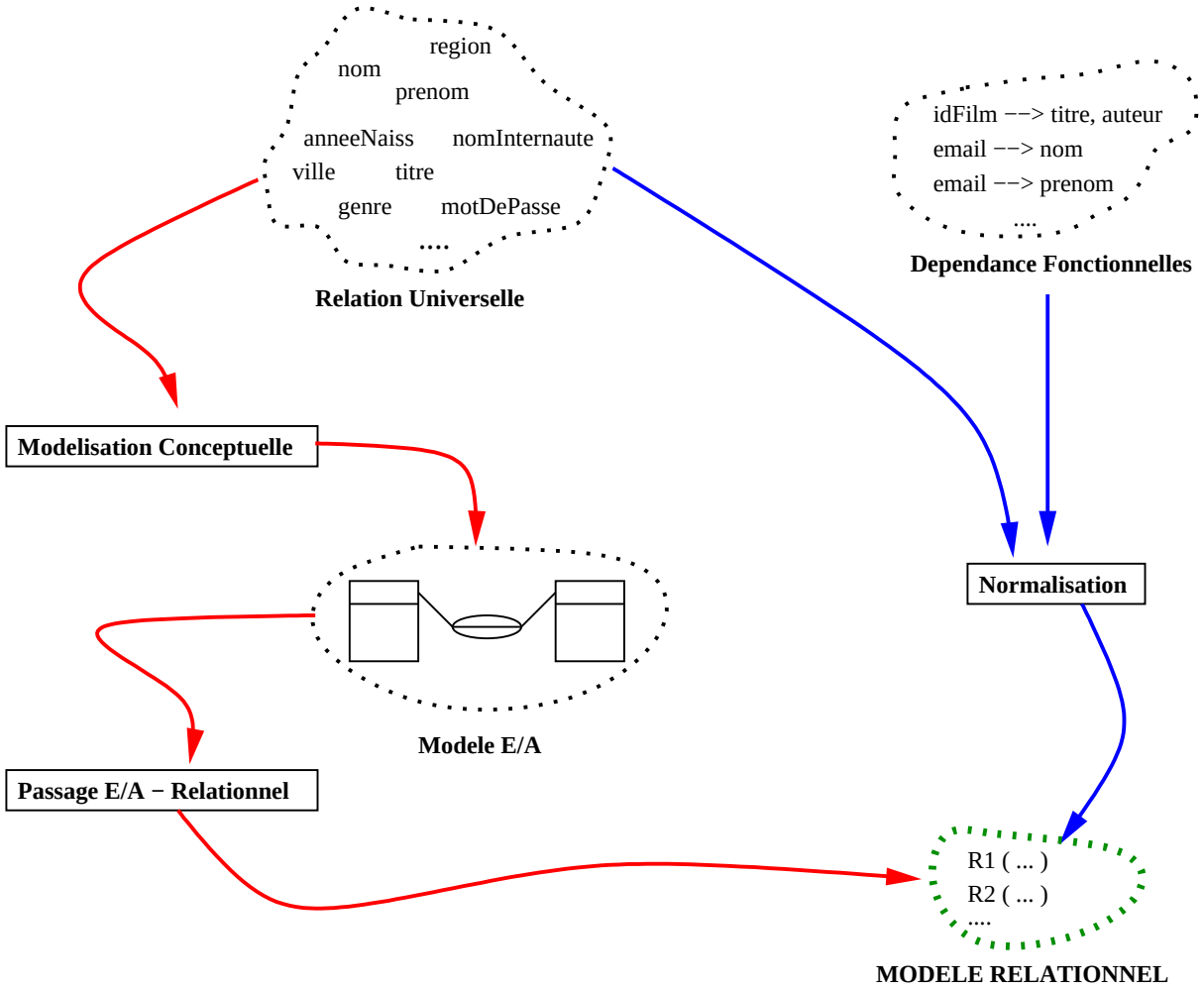
- **Schéma d'une base de données**
  - Ensemble des schémas de relations de cette base .

Le choix des relations (et schémas de relations) est essentiel

- Il détermine les qualités de la base
  - Performance, exactitude, exhaustivité, disponibilité des infos, etc.
- La théorie des bases de données relationnelles définit ce qu'est un bon schéma et propose des outils formels pour y parvenir.
  - Contraintes d'intégrité. Normalisation.
- En pratique: Manière moins rigoureuse et plus accessible:
  - Transcription du schéma conceptuel (E/A) en schéma relationnel.

Le modèle Relationnel

- Comment obtenir un schéma relationnel ?



Le modèle Relationnel

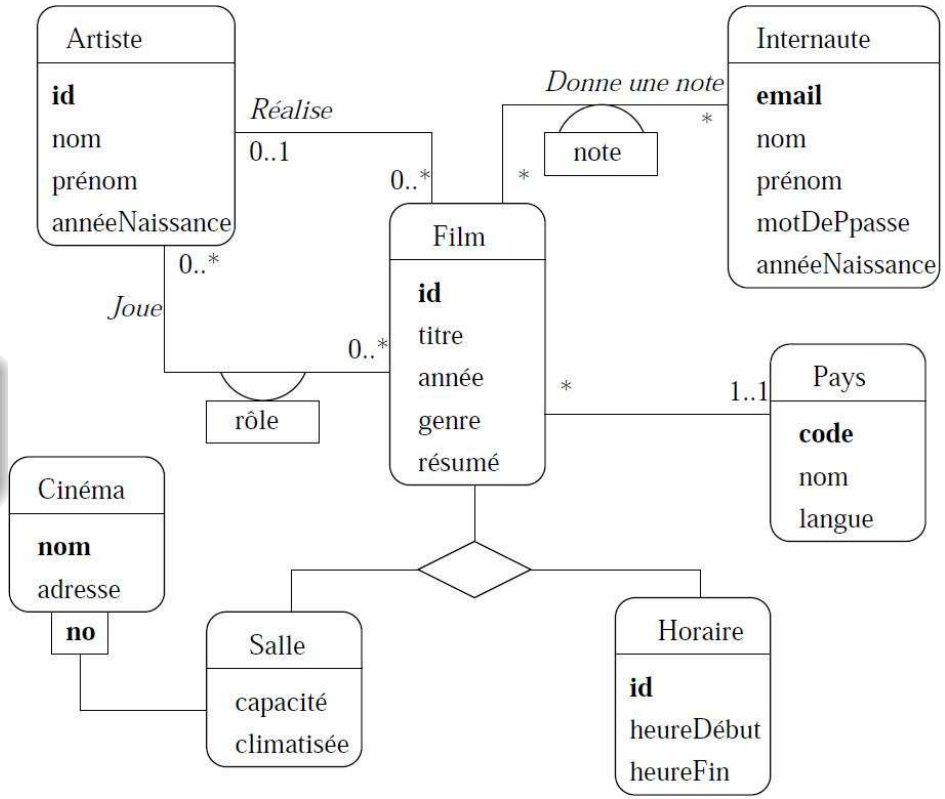
- Passage d'un schéma E/A à un schéma relationnel

Le modèle Relationnel

Passage d'un schéma E/A à un schéma relationnel

- Passage d'un modèle à deux structures (entités + associations) à un modèle à une structure (relations).
  - Nécessité de préserver les liens existant dans un schéma E/A.
  - Mécanisme de référence par valeur basé sur les clés des relations.

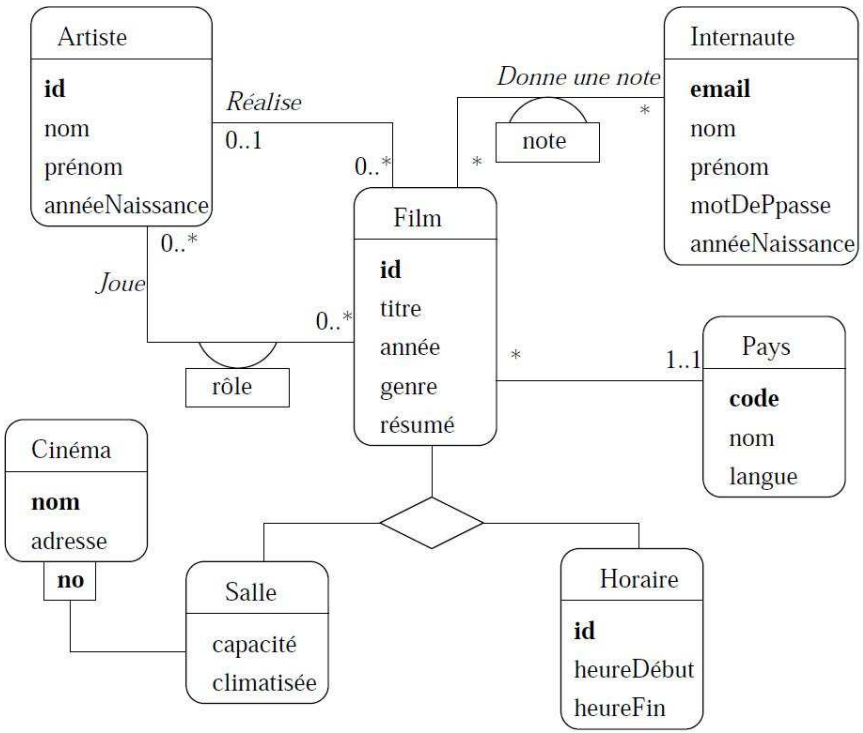
- Cas d'étude:
  - Schéma de la BD *Films*



Le modèle Relationnel

Passage d'un schéma E/A à un schéma relationnel

- **Règle 1: entités.**
  Pour chaque entité du schéma E/A
  - On crée une relation de même nom que l'entité.
  - Chaque propriété de l'entité, y compris l'identifiant, devient un attribut de la relation (colonne).
  - Les attributs de l'identifiant constituent la clé de la relation.



*Film (***idFilm**, *titre, année, genre, résumé)*

*Artiste (***idArtiste**, *nom, prénom, anneeNaissance)*

*Internaute (***email**, *nom, prénom, région)*

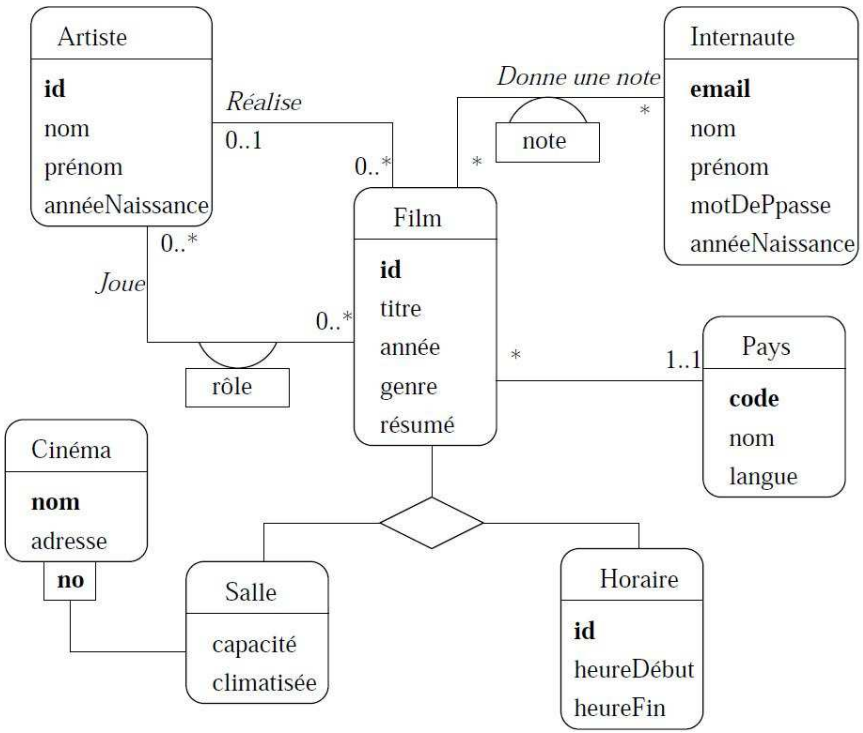
*Pays (***code**, *nom, langue)*

- **Remarque:** Pour l'instant, on a perdu tout lien entre les relations.

Le modèle Relationnel

Passage d'un schéma E/A à un schéma relationnel

- **Règle 2: association de un à plusieurs (1/2).**
  Soit une association de un à plusieurs entre A et B.
  - On crée les relations R_A et R_B correspondant aux entités A et B.
  - L'identifiant de B devient un attribut de R_A (clé étrangère).



Film (**idFilm**, *titre*, *année*, *genre*, *résumé*)

Artiste (**idArtiste**, *nom*, *prénom*, *anneéNaissance*)

Pays (**code**, *nom*, *langue*)

⇓⇓

idMES ∼ **idArtiste**

Film (**idFilm**, *titre*, *année*, *genre*, *résumé*, *idMES*, *codePays*)

Artiste (**idArtiste**, *nom*, *prénom*, *anneéNaissance*)

Pays (**code**, *nom*, *langue*)

Le modèle Relationnel

Passage d'un schéma E/A à un schéma relationnel

- **Règle 2: association de un à plusieurs (2/2).**
  - Exemple de représentation des associations *Film-Artiste* et *Film-Pays*.

idFilm	titre	année	genre	*idMES*	*codePays*
100	Alien	1979	Science Fiction	1	US
101	Vertigo	1958	Suspense	2	US
102	Psychose	1960	Suspense	2	US
103	Kagemusha	1980	Drame	3	JP
104	Volte-face	1997	Action	4	US
105	Van Gogh	1991	Drame	8	FR
106	Titanic	1997	Drame	6	US
107	Sacrifice	1986	Drame	7	FR

La table *Film*

idArtiste	nom	prénom	annéeNaiss
1	Scott	Ridley	1943
2	Hitchcock	Alfred	1899
3	Kurosawa	Akira	1910
4	Woo	John	1946
6	Cameron	James	1954
7	Tarkovski	Andrei	1932
8	Pialat	Maurice	1925

La table *Artiste*

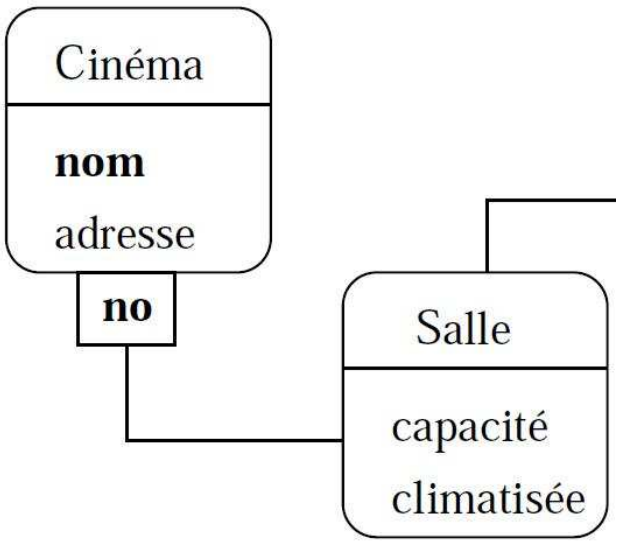
code	nom	langue
US	Etats Unis	anglais
FR	France	français
JP	Japon	japonais

La table *Pays*

Le modèle Relationnel

Passage d'un schéma E/A à un schéma relationnel

- **Règle 3: association avec type entité faible.**
  - Une entité faible est toujours identifiée par rapport à une autre entité (ex: Salle de Cinéma)
  - Il s'agit d'une association "*un à plusieurs*" $\Rightarrow$ Même règle de passage:
    - ★ Mécanisme de clé étrangère: référencer l'entité forte dans l'entité faible.
    - ★ **Nuance:** La clé étrangère fait partie de l'identifiant de l'entité faible.



Cinéma (**nomCinéma**, *numéro*, *rue*, *ville*)

Salle (**nomCinéma**, **no**, *Capacité*)

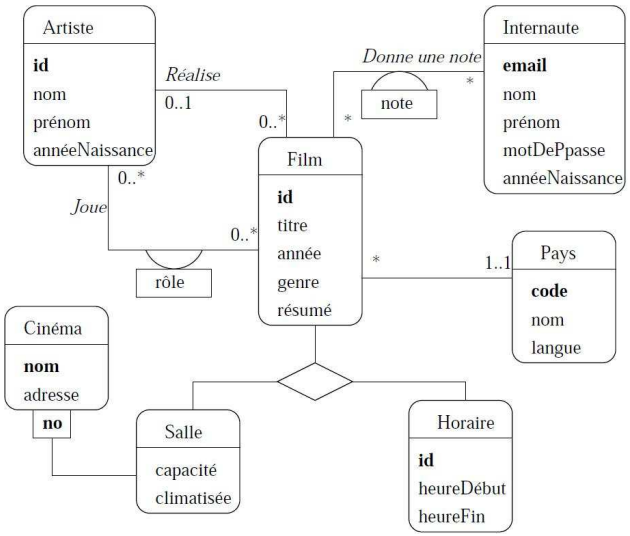
Le modèle Relationnel

Passage d'un schéma E/A à un schéma relationnel

- **Règle 4: association binaire de plusieurs à plusieurs.**

  Soit une association binaire *n-m* entre A et B.

  1. On crée les relations R_A et R_B correspondant aux entités A et B.
  2. On crée une relation R_{A-B} pour l'association.
  3. La clé de R_A et la clé de R_B deviennent des attributs de R_{A-B}.
  4. La clé de R_{A-B} est la concaténation des clés des relations R_A et R_B.
  5. Les propriétés de l'association deviennent des attributs de R_{A-B}.



Film (**idFilm**, *titre*, *année*, *genre*, *résumé*, *idMES*, *codePays*)

Artiste (**idArtiste**, *nom*, *prénom*, *anneéNaissance*)

Internaute (**email**, *nom*, *prénom*, *région*)

Role (**idFilm**, **idActeur**, *nomRole*)

Notation (**email**, **idFilm**, *note*)

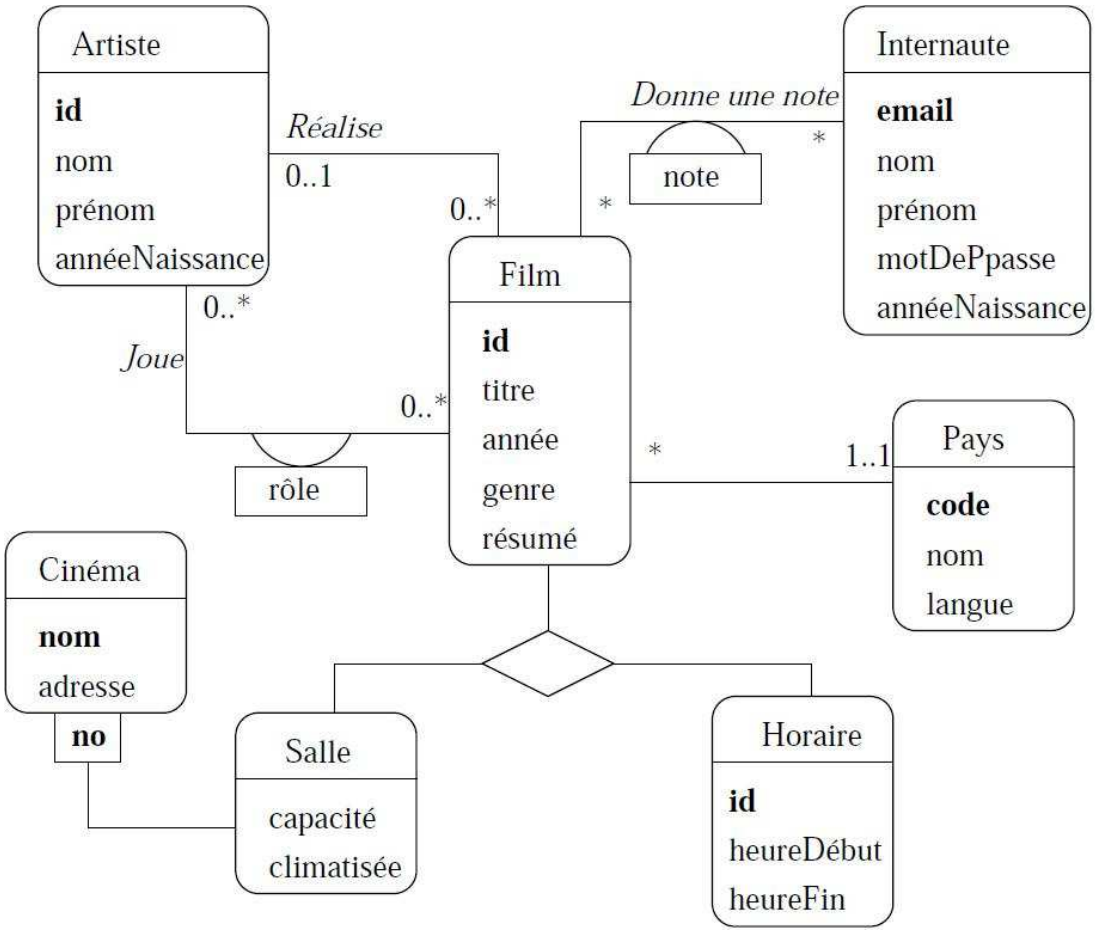
idMES $\sim$ **idArtiste**
codePays $\sim$ **idPays**
idActeur $\sim$ **idArtiste**
. . .

Le modèle Relationnel

Passage d'un schéma E/A à un schéma relationnel

- **Règle 5: associations ternaires.**

  Soit une association ternaire entre A, B et C.



?

Le modèle Relationnel

Passage d'un schéma E/A à un schéma relationnel

- **Retour sur le choix des identifiants.**
  Il est préférable en général de choisir un identifiant "neutre" qui ne soit pas une propriété de l'entité.
  - Chaque valeur de l'identifiant doit caractériser de manière unique une occurrence
    - ⋆ Ex : **Titre** pour la relation *Film* ou **nom** pour la relation *Acteur* ne sont pas de bons choix.
  - Si on utilise un ensemble de propriétés comme identifiant, la référence à une occurrence est très lourde
    - ⋆ Ex : La clé de *Cinéma* pourrait être (**nom**, **rue**, **ville**).
  - L'identifiant sert de référence externe et *ne doit donc jamais être modifiable* (il faudrait répercuter les mises à jour)
    - ⋆ Ex : **Titre** pour la relation *Film* ou **nom** pour la relation *Acteur* ne sont pas de bons choix.

Le modèle Relationnel

- Dépendances Fonctionnelles et Normalisation (formes normales)
  - *S'il nous reste du temps ...*

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- Le DDL **SQL**

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- Le LDD permet de spécifier le schéma d'une BD relationnelle.
- Ce langage correspond à une partie de la norme SQL
  - L'autre partie étant relative à la *manipulation des données* (LMD)
- La définition d'un schéma logique comprend trois parties:
  - Description des *tables* et de leur *contenu*.
    - ★ Attributs, types, etc.
  - Description des *contraintes* qui portent sur les données de la base.
    - ★ Contrôles sur l'intégrité des donnés qui s'imposent à toutes les applications accédant à cette base.
  - Description de la représentation physique.
    - ★ Les index, ...

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Types SQL.**
  - La norme SQL ANSI propose un ensemble de types.
  - Tailles données à titre indicatif (peuvent varier selon les systèmes)

Type	Description	Taille
`INTEGER`	Type des entiers relatifs	4 octets
`SMALLINT`	Idem.	2 octets
`BIGINT`	Idem.	8 octets
`FLOAT`	Flottants simple précision	4 octets
`DOUBLE PRECISION`	Flottants double précision	8 octets
`REAL`	Synonyme de `FLOAT`	4 octets
`NUMERIC` (M, D)	Numérique avec précision fixe.	M octets
`DECIMAL` (M, D)	Idem.	M octets
`CHAR`(M)	Chaînes de longueur fixe	M octets
`VARCHAR`(M)	Chaînes de longueur variable	$L+1$ avec $L \leqslant M$
`BIT VARYING`	Chaînes d'octets	Longueur de la chaîne.
`DATE`	Date (jour, mois, an)	env. 4 octets
`TIME`	Horaire (heure, minutes, secondes)	env. 4 octets
`DATETIME`	Date et heure	8 octets
`YEAR`	Année	2 octets

  - Chaîne très longue (txt): BIT VARYING. Variantes: BLOB ou LONG.

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Création des tables.**
  - La commande principale est : `CREATE TABLE`

```
CREATE TABLE Internaute   (email VARCHAR (50) NOT NULL,
                           nom VARCHAR (20) NOT NULL,
                           prenom VARCHAR (20),
                           motDePasse VARCHAR (60) NOT NULL,
                           anneeNaiss DECIMAL (4));
```

  - `NOT NULL`: L'attribut correspondant doit **toujours** avoir une valeur.
    - ★ On ne peut pas faire d'opération incluant un `NULL`. Ni une comparaison.
    - ★ Le SGBD rejettera toute tentative d'insertion non conforme.

  - Autre manière de forcer: valeur par défaut (option `DEFAULT`)
    - ★ Valeur par défaut: constante ou variable système (ex: `SYSDATE`)

```
CREATE TABLE Cinéma   (nom VARCHAR (50) NOT NULL,
                       adresse VARCHAR (50) DEFAULT 'Inconnue');
```

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 1/12**

  Voici les règles que l'on peut demander au système de garantir :

  - Un attribut doit toujours avoir une valeur.
    - ★ Contrainte `NOT NULL` vue précédemment.
  - Un attribut (ou un ensemble d'attributs) constitue la clé de la relation.
    - ★ Option `PRIMARY KEY`
  - Un attribut dans une table est liée à la clé primaire d'une autre table.
    - ★ Clé étrangère (*intégrité référentielle*). `FOREIGN KEY ... REFERENCES`
  - La valeur d'un attribut doit être unique au sein de la relation.
    - ★ *Clé secondaire*. Option `UNIQUE`
  - Toute règle s'appliquant à la valeur d'un attribut (valeur *min* par ex).
    - ★ Contrainte de domaine. Option `CHECK`

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 2/12**

Clés d'une table

Une clé est un attribut (ou un ensemble d'attributs) qui identifie de manière unique un tuple. Il peut y avoir plusieurs clés mais l'une d'entre elles doit être choisie comme clé primaire (option `PRIMARY KEY`).

```
CREATE TABLE Internaute   (email VARCHAR (50) NOT NULL,
                           nom VARCHAR (20) NOT NULL,
                           prenom VARCHAR (20),
                           motDePasse VARCHAR (60) NOT NULL,
                           anneeNaiss DECIMAL (4),
                           PRIMARY KEY (email));
```

- Il devrait toujours y avoir une `PRIMARY KEY` dans une table pour ne pas risquer d'insérer involontairement deux lignes strictement identiques.

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 3/12**

Clés d'une table

Une clé peut être constituée de plusieurs attributs :

```
CREATE TABLE Notation   (idFilm INTEGER NOT NULL,
                         email VARCHAR (30) NOT NULL,
                         note INTEGER DEFAULT 0,
                         PRIMARY KEY (titre, email));
```

- Tous les attributs figurant dans une clé doivent être déclarés `NOT NULL`.

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 4/12**

Clés d'une table

On peut également spécifier que la valeur d'un attribut est unique pour l'ensemble de la colonne.

- Cela permet d'indiquer des *clés secondaires*

- Ex: deux artistes ne peuvent pas avoir les mêmes nom et prénom

```
CREATE TABLE Artiste   (id INTEGER NOT NULL,
                        nom VARCHAR (30) NOT NULL,
                        prenom VARCHAR (30) NOT NULL,
                        anneeNaiss INTEGER,
                        PRIMARY KEY (id),
                        UNIQUE (nom, prenom));
```

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 5/12**

Clés d'une table

On peut également spécifier que la valeur d'un attribut est unique pour l'ensemble de la colonne.

- Un autre exemple d'utilisation d'une clé secondaire
  - ⋆ On ne peut pas trouver deux cinémas à la même adresse.

```
CREATE TABLE Cinéma   (nomCinema VARCHAR (30) NOT NULL,
                       adresse VARCHAR (50) UNIQUE,
                       PRIMARY KEY (nomCinema));
```

- N'est possible que quand cette contrainte ne concerne qu'un attribut.
  - ⋆ `UNIQUE (x, y)` $\neq$ `UNIQUE(x)` $\wedge$ `UNIQUE(y)`.

- La clause `UNIQUE` ne s'applique pas aux valeurs `NULL`
  - ⋆ Il peut y avoir des cinémas d'adresse inconnue.

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 6/12**

Clés étrangères

La norme SQL ANSI permet d'indiquer quelles sont les *clés étrangères* dans une table, autrement dit, quels sont les attributs qui font référence à une ligne dans une autre table.

- Option `FOREIGN KEY ... REFERENCES`.

```
CREATE TABLE Film   (idFilm INTEGER NOT NULL,
                     titre VARCHAR (50) NOT NULL,
                     annee INTEGER NOT NULL,
                     idMES INTEGER,
                     codePays INTEGER,
                     PRIMARY KEY (idFilm),
                     FOREIGN KEY (idMES) REFERENCES Artiste,
                     FOREIGN KEY (codePays) REFERENCES Pays );
```

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 7/12**

Clés étrangères

La commande

```
FOREIGN KEY (idMES) REFERENCES Artiste
```

indique que `idMES` référence la clé primaire de la table *Artiste*.

- ⋆ Le SGBD vérifiera, pour toute modification pouvant affecter le lien entre les deux tables, que la valeur de `idMES` correspond bien à une ligne de *Artiste*. Ces modifications sont:

1. L'insertion dans *Film* d'une valeur inconnue pour `idMES` ;
2. La destruction d'un artiste ;
3. La modification de `id` dans *Artiste* ou de `idMES` dans *Film*.

- Quand un attribut est à `NULL` (ex: `idMES` de *Film*), la contrainte d'intégrité référentielle ne s'applique pas.

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 8/12**

Clés étrangères

- Que se passe-t-il quand la violation d'une contrainte d'intégrité est détectée par le système ?
  - ★ Par défaut: mise à jour rejetée.
  - ★ Il est aussi possible de demander la **répercussion** de cette mise à jour de manière à ce que la contrainte soit respectée.
- Les événements que l'on peut répercuter sont:
  - ★ La modification, désignée par `ON UPDATE`
  - ★ La destruction, désignée par `ON DELETE`
- La répercussion elle-même consiste
  - ★ Soit à mettre la clé étrangère à `NULL` (option `SET NULL`),
  - ★ Soit à appliquer la même opération aux lignes de l'entité composante (option `CASCADE`).

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 9/12**

Clés étrangères

- Que se passe-t-il quand la violation d'une contrainte d'intégrité est détectée par le système ?
  - ⋆ **Exemple 1:** Comment indiquer que la destruction d'un metteur en scène déclenche la mise à `NULL` de la clé étrangère `idMES` pour tous les films qu'il a réalisés.

```
CREATE TABLE Film   (idFilm INTEGER NOT NULL,
                     titre VARCHAR (50) NOT NULL,
                     annee INTEGER NOT NULL,
                     idMES INTEGER,
                     codePays INTEGER,
                     PRIMARY KEY (idFilm),
                     FOREIGN KEY (idMES) REFERENCES Artiste
                             ON DELETE SET NULL,
                     FOREIGN KEY (codePays) REFERENCES Pays);
```

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 10/12**

Clés étrangères

- Que se passe-t-il quand la violation d'une contrainte d'intégrité est détectée par le système ?
  - ★ Cas d'une entité faible: on décide en général de détruire le composant quand on détruit le composé.
  - ★ **Exemple 2:** Destruction d'un cinéma ⇒ Destruction de ses salles ; Modification clé d'un cinéma ⇒ Répercussion des modif. sur ses salles.

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

  - ★ Aurions-nous pu faire `ON DELETE SET NULL` pour `nomCinema` ?

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 11/12**

Contraintes de domaine: option `CHECK`

- **Ex:** Restriction des valeurs possibles pour annee et genre.

```
CREATE TABLE Film   (idFilm INTEGER NOT NULL,
                     titre VARCHAR (50) NOT NULL,
                     annee INTEGER
                         CHECK (annee BETWEEN 1890 AND 2000) NOT NULL,
                     genre VARCHAR (10)
                         CHECK (genre IN ('Histoire', 'Western')),
                     idMES INTEGER,
                     codePays INTEGER,
                     PRIMARY KEY (idFilm),
                     FOREIGN KEY (idMES) REFERENCES Artiste,
                     FOREIGN KEY (codePays) REFERENCES Pays);
```

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Contraintes d'intégrité. 12/12**

– La spécification des actions (contraintes)
  - `ON DELETE`
  - `ON UPDATE`
  - `CHECK`
  - `FOREIGN KEY`

simplifie considérablement la gestion de la base par la suite
  - On n'a plus, par ex, à se soucier de faire des mises à jour en cascade.

– Si on n'opte pas pour la vérification automatique:
  - Il faudra faire la vérification dans l'application cliente.
  - Cela est plus lourd à gérer.

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Modification du schéma. 1/2**
  La forme générale de la commande est :

–

$$\texttt{ALTER TABLE } \textit{nomTable} \texttt{ ACTION } \textit{description}$$

–

où `ACTION` peut être principalement

  - `ADD`,
  - `MODIFY`,
  - `DROP`,
  - `RENAME`

et *description* est la commande de modification associée à `ACTION`

– La modification d'une table peut poser des problèmes. Par exemple, passer un attribut à `NOT NULL` implique que cet attribut a déjà des valeurs pour toutes les lignes de la table.

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- **Modification du schéma. 2/2**

Modification des attributs

- ?

```
ALTER TABLE Internaute ADD region VARCHAR(10);
```

- ?

```
ALTER TABLE Internaute MODIFY region VARCHAR(30);
```

- ?

```
ALTER TABLE Internaute DROP region;
```

Le modèle Relationnel

Le langage de définition de données (**DDL**) SQL

- Création d'index
  - *S'il nous reste du temps ...*

Plan

L'algèbre relationnelle

Préambule 1/4

Peut être vue comme un langage de programmation très simple permettant d'exprimer des requêtes sur une base de données relationnelle :

- Ensemble d'opérateurs permettant de manipuler les *relations*
  - *Union*, *Différence*,
  - *sélectionner* une partie de la relation,
  - *Produis cartésiens*, *Projections*

5 Opérations "relationnelles" (ensemblistes) de base :

- Une opération prend en entrée une ou deux relations (ensembles de *n-uplets* ou *tuples*) de la base de données
  - ⋆ Opérateurs unaires: Sélection (σ), Projection (π).
  - ⋆ Opérateurs binaires: Union ($\cup$), Différence ($-$), Produit cartésien ($\times$).
- Autres opérations qui s'expriment en fonction des 5 opérations de base
  - ⋆ Jointure ($\bowtie$), Intersection ($\cap$) et Division ($\div$).
- Le résultat **est toujours** une relation (un ensemble)

L'algèbre relationnelle

Préambule 2/4

Exemple de référence: organisme de voyage

- Séjours (sportifs, culturels, etc) dans des stations de vacances.
- Activités (ski, voile, tourisme).
- Maintient d'une liste des clients et des séjours auxquels ils ont participé.

Schéma relationnel de la Base

*Station (***nomStation**, *capacité, lieu, région, tarif)*

*Activité (***nomStation**, **libellé**, *prix)*

*Client (***id**, *nom, prénom, ville, région, solde)*

*Séjour (***idClient**, **station**, **début**, *nbPlaces)*

- Quel est le modèle E/A correspondant ?

L'algèbre relationnelle

Préambule 3/4

nomStation	capacité	lieu	région	tarif
Venusa	350	Guadeloupe	Antilles	1200
Farniente	200	Seychelles	Océan Indien	1500
Santalba	150	Martinique	Antilles	2000
Passac	400	Alpes	Europe	1000

Table **Station**

nomStation	libellé	prix
Venusa	Voile	150
Venusa	Plongée	120
Farniente	Plongée	130
Passac	Ski	200
Passac	Piscine	20
Santalba	Kayac	50

Table **Activité**

idClient	station	début	nbPlaces
10	Passac	1998-07-01	2
30	Santalba	1996-08-14	5
20	Santalba	1998-08-03	4
30	Passac	1998-08-15	3
30	Venusa	1998-08-03	3
20	Venusa	1998-08-03	6
30	Farniente	1999-06-24	5
10	Farniente	1998-09-05	3

Table **Séjour**

L'algèbre relationnelle

Préambule 4/4

id	nom	prénom	ville	région	solde
10	Fogg	Phileas	Londres	Europe	12465
20	Pascal	Blaise	Paris	Europe	6763
30	Kerouac	Jack	New York	Amérique	9812

Table **Client**

L'algèbre relationnelle

Les opérateurs de l'algèbre relationnelle

La sélection - σ

- $\sigma_F(R)$: renvoie (de la relation R) les tuples satisfaisant F.
- Critère de sélection F:
  - Comparaison entre un attribut A de la relation et une constante a
    - ⋆ $A\ \Theta\ a, \qquad \Theta \in \{=, \neq, <, >, \leq, \geq\}$
  - Comparaison entre deux attributs A_1 et A_2 de la relation: $A_1\ \Theta\ A_1$

nomStation	capacité	lieu	région	tarif
Venusa	350	Guadeloupe	Antilles	1200
Farniente	200	Seychelles	Océan Indien	1500
Santalba	150	Martinique	Antilles	2000
Passac	400	Alpes	Europe	1000

⇓

$$\sigma_{region='Antilles'}(Station)$$

⇓

nomStation	capacité	lieu	région	tarif
Venusa	350	Guadeloupe	Antilles	1200
Santalba	150	Martinique	Antilles	2000

L'algèbre relationnelle

Les opérateurs de l'algèbre relationnelle

La projection - π

- $\pi_{A_1,A_2,\cdots,A_k}(R)$:
  - S'applique à une relation R et ne garde que les attributs $A_1, \cdots, A_k$.
  - Contrairement à σ, on ne supprime pas des lignes mais des colonnes.

nomStation	capacité	lieu	région	tarif
Venusa	350	Guadeloupe	Antilles	1200
Farniente	200	Seychelles	Océan Indien	1500
Santalba	150	Martinique	Antilles	2000
Passac	400	Alpes	Europe	1000

$\Downarrow$

$\pi_{nomStation,region}(Station)$

$\Downarrow$

nomStation	région
Venusa	Antilles
Farniente	Océan Indien
Santalba	Antilles
Passac	Europe

  - Le résultat est toujours une relation. **Question:** $\pi_{region}(Station) = ?$

L'algèbre relationnelle

Les opérateurs de l'algèbre relationnelle

Le produit cartésien - $\times$

- $R \times S$:
  - Crée une relation où chaque tuple de R est associé à chaque tuple de S.

A	B
a	b
x	y

R

C	D
c	d
u	v
x	y

S

A	B	C	D
a	b	c	d
a	b	u	v
a	b	x	y
x	y	c	d
x	y	u	v
x	y	x	y

R $\times$ S

  - $|R \times S| = |R| \times |S|$
  - Ne prend vraiment son sens qu'associé à l'opération de sélection (σ), ce qui permet d'exprimer des jointures ($\bowtie$).

L'algèbre relationnelle

Les opérateurs de l'algèbre relationnelle

Le produit cartésien - $\times$

- **Conflits** de noms d'attributs
  - Quand les deux relations ont des attributs qui ont le même nom.
  - Il faut distinguer l'origine des colonnes dans le résultat.

⋆ **Solution 1:** *Préfixation* de l'att. par le nom de la table

A	B
m	n
o	p

T

R.A	R.B	T.A	T.B
a	b	m	n
a	b	o	p
x	y	m	n
x	y	o	p

R $\times$ **T**

⋆ **Solution 2:** *Renommage* d'attributs d'une relation.

– Renommage de A en C et de B en D dans la relation T:

$$R \times \rho_{A \to C, B \to D}(T)$$

⋆ Solution assez lourde à utiliser.

L'algèbre relationnelle

Les opérateurs de l'algèbre relationnelle

L'union - $\cup$

- $R \cup S$: contiendra tous les tuples existant dans l'une ou dans l'autre.
- Condition: les relations doivent avoir le même schéma.
  - Union $R \cup S$ interdite: $schema(R) \neq schema(S)$
  - Mais ...

A	B
a	b
x	y
c	d
u	v

$\mathbf{R} \cup \rho_{\mathbf{C} \to \mathbf{A}, \mathbf{D} \to \mathbf{B}}(\mathbf{S})$

- Remarque:
  - Le résultat étant une relation, le tuple (x, y) n'y figure qu'une seule fois

L'algèbre relationnelle

Les opérateurs de l'algèbre relationnelle

La différence - $-$

- $R - S$: contiendra tous les tuples de R qui ne sont pas dans S.
- Condition: les relations doivent avoir le même schéma.

A	B
a	b

$$\mathbf{R} - \rho_{\mathbf{C}\rightarrow\mathbf{A},\mathbf{D}\rightarrow\mathbf{B}}(\mathbf{S})$$

- Seul opérateur qui permet d'exprimer des requêtes avec une négation
  - Quand on veut "rejeter" quelque chose.
  - Quand on "ne veut pas" des lignes ayant telle propriété.
  - Voir exemple de négation en fin de chapitre (requêtes avec $\cup$ et $-$).

L'algèbre relationnelle

Les opérateurs de l'algèbre relationnelle

La jointure - $\bowtie$

- Retour sur l'opération du produit cartésien ($\times$):
  - Résultat en général énorme. Ne présente pas bcp d'intérêt à lui seul.
  - Beaucoup de lignes sans intérêt: des infos sans lien sur une même ligne.

S.nomStation	cap.	lieu	région	tarif	A.nomStation	libelle	prix
Venusa	350	Guadeloupe	Antilles	1200	Venusa	Voile	150
Venusa	350	Guadeloupe	Antilles	1200	Venusa	Plongée	120
Venusa	350	Guadeloupe	Antilles	1200	Farniente	Plongée	130
Venusa	350	Guadeloupe	Antilles	1200	Passac	Ski	200
Venusa	350	Guadeloupe	Antilles	1200	Passac	Piscine	20
Venusa	350	Guadeloupe	Antilles	1200	Santalba	Kayac	50
Farniente	200	Seychelles	Océan Indien	1500	Venusa	Voile	150
Farniente	200	Seychelles	Océan Indien	1500	Venusa	Plongée	120
Farniente	200	Seychelles	Océan Indien	1500	Farniente	Plongée	130
Farniente	200	Seychelles	Océan Indien	1500	Passac	Ski	200
Farniente	200	Seychelles	Océan Indien	1500	Passac	Piscine	20
Farniente	200	Seychelles	Océan Indien	1500	Santalba	Kayac	50
Santalba	150	Martinique	Antilles	2000	Venusa	Voile	150
Santalba	150	Martinique	Antilles	2000	Venusa	Plongée	120
Santalba	150	Martinique	Antilles	2000	Farniente	Plongée	130
Santalba	150	Martinique	Antilles	2000	Passac	Ski	200
Santalba	150	Martinique	Antilles	2000	Passac	Piscine	20
Santalba	150	Martinique	Antilles	2000	Santalba	Kayac	50
Passac	400	Alpes	Europe	1000	Venusa	Voile	150
Passac	400	Alpes	Europe	1000	Venusa	Plongée	120
Passac	400	Alpes	Europe	1000	Farniente	Plongée	130
Passac	400	Alpes	Europe	1000	Passac	Ski	200
Passac	400	Alpes	Europe	1000	Passac	Piscine	20
Passac	400	Alpes	Europe	1000	Santalba	Kayac	50

Station $\times$ Activite

L'algèbre relationnelle

Les opérateurs de l'algèbre relationnelle

La jointure - $\bowtie$

- Retour sur l'opération du produit cartésien ($\times$):
  - On peut utiliser la sélection (σ) pour rapprocher les informations générales sur une station et la liste des activités de cette station.
  - L'opération **composée** est:

$$\sigma_{S.nomStation=A.nomStation}(Station \times Activite)$$

  - Résultat de la composition de $\times$ et σ:

S.nomStation	cap.	lieu	région	tarif	A.nomStation	libelle	prix
Venusa	350	Guadeloupe	Antilles	1200	Venusa	Voile	150
Venusa	350	Guadeloupe	Antilles	1200	Venusa	Plongée	120
Farniente	200	Seychelles	Océan Indien	1500	Farniente	Plongée	130
Santalba	150	Martinique	Antilles	2000	Santalba	Kayac	50
Passac	400	Alpes	Europe	1000	Passac	Ski	200
Passac	400	Alpes	Europe	1000	Passac	Piscine	20

$$\sigma_{\mathbf{S.nomStation=A.nomStation}}(\mathbf{Station} \times \mathbf{Activite})$$

L'algèbre relationnelle

Les opérateurs de l'algèbre relationnelle

La jointure - $\bowtie$

- L'opération suivante est une *jointure*.

$$\sigma_{S.nomStation=A.nomStation}(Station \times Activite)$$

- On peut directement et simplement la noter par :

$$Station \bowtie_{nomStation=nomStation} Activite$$

  - Consiste à rapprocher les lignes ayant des valeurs d'attr. identiques
  - Dans 90% des cas, ces attributs communs sont (1) la clé primaire d'une des relations et (2) la clé étrangère dans l'autre relation.

$$R \bowtie_F S \quad \equiv \quad \sigma_F(R \times S)$$

L'algèbre relationnelle

Expression de requêtes avec l'algèbre

Sélection (σ) généralisée

- *Conjonction*

$$\sigma_{capacite>200}\ (\sigma_{region='Antilles'}(Station)) \equiv \sigma_{capacite>200 \wedge region='Antilles'}(Station)$$

- *Disjonction*

$$\sigma_{capacite>200}(Station) \cup \sigma_{region='Antilles'}(Station) \equiv \sigma_{capacite>200 \vee region='Antilles'}(Station)$$

- *Différence*

$$\sigma_{capacite>200}(Station) - \sigma_{region='Antilles'}(Station) \equiv \sigma_{capacite>200 \wedge region\neq'Antilles'}(Station)$$

L'algèbre relationnelle

Expression de requêtes avec l'algèbre

Requêtes conjonctives

- Constituent l'essentiel des requêtes courantes.
  - S'expriment avec des *et* (par opposition à *ou* ou *not*)
  - Peuvent s'écrire avec seulement trois opérateurs: π, σ, $\times$ (donc $\bowtie$).

- Les plus simples sont celles où on n'utilise que π et σ.

  - Nom des stations aux Antilles :

$$\pi_{nomStation}(\sigma_{region='Antilles'}(Station))$$

  - Nom des stations où l'on pratique la voile:

$$\pi_{nomStation}(\sigma_{libelle='Voile'}(Activite))$$

  - Nom et prénom des clients européens:

$$\pi_{nom,prenom}(\sigma_{region='Europe'}(Client))$$

L'algèbre relationnelle

Expression de requêtes avec l'algèbre

Requêtes conjonctives

- Requêtes impliquant une *jointure* ($\bowtie$)
  - Légèrement plus complexes et extrêmement utiles.
  - Obligatoire quand les attributs nécessaires sont répartis dans au moins deux tables. Ces attributs peuvent être:
    - ⋆ Soit des attributs qui figurent dans le résultat ;
    - ⋆ Soit des attributs sur lesquels on exprime un critère de sélection.

- **Ex:** Nom et région des stations où l'on pratique la voile.

$$\pi_{nomStation,region}(Station \bowtie_{nomStation=nomStation} \sigma_{libelle='Voile'}(Activite))$$

- En général, la *jointure* se fait sur des attributs clé primaire dans une relation, et clé secondaire dans l'autre.
  - Il ne s'agit pas d'une règle absolue.

L'algèbre relationnelle

Expression de requêtes avec l'algèbre

Requêtes conjonctives

- Nom des clients qui sont allés à Passac :

$$\pi_{nom}(Client \bowtie_{id=idClient} \sigma_{nomStation='Passac'}(Sejour))$$

- Quelles régions a visité le client 30 :

$$\pi_{region}(\sigma_{idClient=30}(Sejour) \bowtie_{station=nomStation} (Station))$$

- Nom des clients qui ont eu l'occasion de faire de la voile :

$$\pi_{nom}(Client \bowtie_{id=idClient} (Sejour \bowtie_{station=nomStation} \sigma_{libelle='Voile'}(Activite)))$$

- Nom des clients partis en vac. dans leur région, et nom de la région.

$$\pi_{nom,client.region}(Client \bowtie_{id=idClient \wedge region=region} (Sejour \bowtie_{station=nomStation} Station))$$

L'algèbre relationnelle

Expression de requêtes avec l'algèbre

Requêtes avec $\cup$ et $-$

- Utilisation moins fréquente, mais peuvent s'avérer nécessaire.
- En particulier, la différence $(-)$ permet d'exprimer une négation.
- Quelles sont les stations qui **ne** proposent **pas** de voile ?

$$\pi_{nomStation}(Station) - \pi_{nomstation}(\sigma_{libelle='Voile'}(Activite))$$

- *"Tous les O qui ne satisfont pas p"* - démarche générale:
  - Construire une requête A qui sélectionne tous les O.
  - Construire une requête B qui sélectionne tous les O satisfaisant p.
  - Finalement, faire $A - B$.

L'algèbre relationnelle

Expression de requêtes avec l'algèbre

Requêtes avec $\cup$ et $-$

- Régions où il y a des clients, mais pas de station.

$$\pi_{region}(Client) - \pi_{region}(Station)$$

- Nom des stations qui n'ont pas reçu de client américain.

$$\pi_{nomStation}(Station) - \pi_{station}(Sejour \bowtie_{idClient=id} \sigma_{region='Amerique'}(Client))$$

- Id des clients qui ne sont pas allés aux Antilles.

$$\pi_{idClient}(Client) - \pi_{idClient}(\sigma_{region='Antilles'}(Station) \bowtie_{nomStation=station} Sejour)$$

  - Exercice: Comment faire pour obtenir le nom de ces clients ?

L'algèbre relationnelle

Expression de requêtes avec l'algèbre

Requêtes avec $\cup$ et $-$

- **Complément d'un ensemble**

  La différence peut être employée pour calculer le complément.

- On veut les *ids* des clients et les *stations* où ils ne sont pas allés.
  - En d'autres termes, parmi toutes les associations *Client/Station* possibles, on veut celles qui ne sont pas représentées dans la base.

$$(\pi_{id}(Client) \times \pi_{nomStation}(Station)) - \pi_{idClient,station}(Sejour)$$

L'algèbre relationnelle

Expression de requêtes avec l'algèbre

Requêtes avec $\cup$ et $-$

- **Quantification universelle**
  - Une propriété est vraie pour tous les éléments d'un ensemble
  - $\Leftrightarrow$ Il n'existe pas d'élément pour lequel la propriété est fausse.
- On se ramène toujours à la négation et à la quantif. existentielle.
  - Stations dont *toutes* les activités ont un prix supérieur à 100 ?
  - $\Leftrightarrow$ Stations pour lesquelles il n'existe pas d'activité avec un prix inférieur à 100 ?

$$\pi_{nomStation}(Station) - \pi_{nomStation}(\sigma_{prix<100}(Activite))$$

L'algèbre relationnelle

Expression de requêtes avec l'algèbre

Requêtes avec $\cup$ et $-$

- **Quantification universelle**
  - **La division ($\div$)**
  - Une des requêtes les plus complexes et rares (heureusement)
- **ids** des clients qui sont allés dans *toutes* les stations.
- Traduction en négation et quantification existentielle:
  - **ids** des clients tels qu'il *n'existe pas* de station où ils *ne soient pas* allés
  - Double négation

$$\pi_{id}(Client) - \pi_{id}((\pi_{id}(Client) \times \pi_{nomStation}(Station)) - \pi_{idClient,station}(Sejour))$$

L'algèbre relationnelle

Expression de requêtes avec l'algèbre

La division (÷) - exemple

NUM	NOM	PNOM	QTE
1	Jean	briques	100
2	Jean	ciment	2
3	Jean	parpaing	2
4	Paul	briques	200
5	Paul	parpaing	3
6	Vincent	parpaing	3

COMM

NOM	PNOM
Jean	briques
Jean	ciment
Jean	parpaing
Paul	briques
Paul	parpaing
Vincent	parpaing

$\mathbf{R} = \pi_{\mathbf{NOM},\mathbf{PNOM}}(\mathbf{COMM})$

PNOM
briques
ciment
parpaing

PROD

- **Requête:** Clients qui commandent **tous** les produits:

NOM
Jean

$\mathbf{R} \div \mathbf{PROD}$

L'algèbre relationnelle

Expression de requêtes avec l'algèbre

La division ($\div$) - exemple

A	B	C	D
a	b	x	m
a	b	y	n
a	b	z	o
b	c	x	o
b	d	x	m
c	e	x	m
c	e	y	n
c	e	z	o
d	a	z	p
d	a	y	m

R

C	D
x	m
y	n
z	o

S

A	B
a	b
c	e

R$\div$S

- 2 arguments (relations) $R(A_1, \cdots, A_m, X_1, \cdots, X_k)$ et $S(X_1, \cdots, X_k)$.
- $T = R \div S \quad \Rightarrow T(A_1, \cdots, A_m)$
  - ★ $T = \{(a_1, \cdots, a_m) \mid \forall\ (x_1, \cdots, x_k) \in S, (a_1, \cdots, a_m, x_1, \cdots, x_k) \in R\}$
  - ★ $T = R_1 - R_2$ où $R_1 = \pi_{A_1, \cdots, A_m}(R)$ et $R_2 = \pi_{A_1, \cdots, A_m}((R_1 \times S) - R)$

Plan

Le langage SQL

Le langage de manipulation de données (DML) SQL

- Langage d'interrogation et de manipulation de données
  - Insertion, mise-à-jour, destruction.
  - Norme SQL2, implantée +/- complètement dans la plupart des SGBDR.
- Langage *déclaratif*[a] qui permet d'interroger la BD sans se soucier de
  - La représentation interne (physique) des données
  - Leur localisation,
  - Des algorithmes.
- S'adresse à une large communauté (pas seulement informaticienne)
- Peut être utilisé:
  - De manière interactive
  - En association avec des langages de programmation.

[a] Voilà ce que je veux, débrouille-toi pour le faire.

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes simples SQL**
  - **Sélections simples** - Les stations se trouvant aux Antilles

```
SELECT   nomStation
FROM     Station
WHERE    region = 'Antilles'
```

  - `SELECT`: Liste des attributs constituant le résultat.
  - `FROM`: Indique les tables dans lesquelles ont trouve les attributs utiles
    - ⋆ Attribut dont on souhaite afficher le contenu.
    - ⋆ Attribut dont on souhaite qu'il ait une valeur particulière.
  - `WHERE`: Conditions que doivent satisfaire les n-uplets du résultat.

nomStation
Venusa
Santalba

  - Utilisation combinée de la sélection (σ) et de la projection (π).

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes simples SQL**
  - **Sélections simples**

– Fonctions applicables aux valeurs des attributs
  - ⋆ Opérateurs arithmétiques $(+, -, *, /, \cdots)$ pour les attributs numériques
  - ⋆ Manipulation de chaînes de caractères: concaténation, sous chaîne, mise en majuscule, ...

```
SELECT   libelle, prix / 6.56, 'Cours de l\'euro = ', 6.56
FROM     Activite
WHERE    nomStation = 'Santalba'
```

libelle	?column?	?column?	?column?
Kayac	7.62	Cours de l'euro	6.56

- ⋆ Sous certains systèmes, les noms des attributs du résultat sont par défaut ceux indiqués dans la clause `SELECT`.

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes simples SQL**
  - **Sélections simples**

    Renommage

```
SELECT   libelle, prix / 6.56 AS prixEnEuros,
                 'Cours de l\'euro = ' AS cde, 6.56 AS cours
FROM     Activite
WHERE    nomStation = 'Santalba'
```

libelle	prixEnEuros	cde	cours
Kayac	7.62	Cours de l'euro	6.56

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes simples SQL**
  - **Doublons**
    - ★ SQL permet l'existence de doublons ($\neq$ de l'algèbre relationnelle).
    - ★ Les clés permettent d'éviter les doublons dans les relations stockées.

```
SELECT   libelle
FROM     Activite
```

libelle
Voile
Plongee
Plongee
Ski
Piscine
Kayac

  - Élimination des doublons

```
SELECT   DISTINCT libelle   FROM Activite
```

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes simples SQL**

  - **Tri du résultat**
    - ⋆ Clause `ORDER BY` suivie de la liste des attributs servant de critère au tri

```
SELECT      *
FROM        Station
ORDER BY    tarif, nomStation
```

  – Trie en ordre ascendant les stations par leurs tarifs
  – Pour un même tarif, présente les stations selon l'ordre lexicographique.

  - Tri par **ordre descendant**: mot clé `DESC` après la liste des attributs

  - Le caractère * dans le `SELECT`: intégralité des colonnes.
    - ⋆ Équivaut à lister tous les attributs

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes simples SQL**

  - **La clause** `WHERE`
    - ★ Condition booléenne portant sur les attributs des relations du `FROM`

  - Connecteurs logiques: `AND`, `OR`, `NOT`
  - Opérateurs de comparaison: $<, <=, >, <=, <>$ $(!=)$, `BETWEEN`, `IN` ...

```
SELECT   nomStation, libelle
FROM     Activite
WHERE    nomStation = 'Santalba'
AND      (prix > 50 AND prix < 120)
```

```
SELECT   nomStation, libelle
FROM     Activite
WHERE    nomStation = 'Santalba'
AND      prix BETWEEN 50 AND 120
```

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes simples SQL**

  - **Chaînes de caractères**

  - Différence entre chaînes de longueur fixe et de longueur variable.
    - ★ Les premiers sont complétées par des blancs (' ') et pas les secondes.
  - SQL distingue les majuscules des minuscules pour les chaînes.
    - ★ `'SANTALBA'` $\neq$ `'Santalba'`
  - Clause `LIKE`: recherche par motif (*pattern matching*)
    - ★ Le caractère '_' désigne n'importe quel caractère
    - ★ Le caractère '%' désigne n'importe quelle chaîne de caractères.

```
SELECT   nomStation
FROM     Station
WHERE    nomStation LIKE '%a'
```

Stations dont le nom se termine par 'a'.

```
SELECT   nomStation
FROM     Station
WHERE    nomStation LIKE 'V____'
```

Commence par 'V' et comprend 6 caractères.

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes simples SQL**

  - **Dates**

  - Possibilité de manipulation de dates.
  - Spécification de date en SQL2:
    - ★ Mot-clé `DATE`
    - ★ Suivi d'une chaîne de caractère au format `'aaaa-mm-jj'`
  - Exemple de date: `DATE '1998-01-01'`
  - Requête: *Id des clients qui ont commencé un séjour en juillet 1998*

```
SELECT   idClient
FROM     Sejour
WHERE    debut BETWEEN DATE '1998-07-01' AND DATE '1998-07-31'
```

  - Beaucoup de fonctions proposées par les systèmes:
    - ★ Calcul d'écarts de dates, ajout de mois, d'années, etc.

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes simples SQL**

  - **Valeurs nulles** (1/3)

    - ★ SQL admet que la valeur de certains attributs soit inconnue
    - ★ On parle de *valeur nulle*, désignée par le mot clé `NULL`.

  - La *valeur nulle* n'est pas une valeur, mais une absence de valeur !
    - ★ Tout opération appliquée à `NULL` donne pour résultat `NULL`.
    - ★ Toute comparaison avec `NULL` donne un résultat qui n'est ni vrai, ni faux, mais une troisième valeur booléenne `UNKNOWN`.

  - Supposons que les définitions `TRUE` $= 1$, `FALSE` $= 0$, `UNKNOWN` $= 1/2$
    - ★ x `AND` $y = min(x, y)$
    - ★ x `OR` $y = max(x, y)$
    - ★ `NOT` $x = 1 - x$

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes simples SQL**

Valeurs nulles (2/3)

- Exemple d'une table `SEJOUR` avec des informations manquantes

SEJOUR			
idClient	station	debut	nbPlaces
10	Passac	1998-07-01	2
20	Santalba	1998-08-03	
30	Passac		3

- La présence de `NULL` peut avoir des effets surprenants. La requête:

```
SELECT   station
FROM     Sejour
WHERE    nbPlaces <= 10 OR nbPlaces >= 10
```

- Devrait ramener toutes les stations de la table.
- 'Santalba' ne figurera pas dans le résultat car `nbPlaces` est `NULL`.

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes simples SQL**

  - **Valeurs nulles** (3/3)

    - ⋆ Autre piège: `NULL` est un mot-clé, pas une constante
    - ⋆ Une comparaison comme `nbPlaces = NULL` est incorrecte

    - ⋆ Prédicat pour tester l'absence de valeur: `IS NULL`

      Ex: *Quels sont les séjours dont on connaît le nombre de places.*

```
SELECT   *
FROM     Sejour
WHERE    nbPlaces IS NOT NULL
```

    - ⋆ La présence de `NULL` est une source de problèmes: dans la mesure du possible il faut l'éviter en spécifiant la contrainte `NOT NULL` ou en donnant une valeur par défaut.

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes sur plusieurs tables**

  - ★ Manipulation simultanée de plusieurs tables
  - ★ Expression des opérations binaires de l'algèbre rel.: $\bowtie, \times, \cup, \cap, -$

- **Jointures**

  - ★ "*Nom des clients avec le nom des stations où ils ont séjourné*"
  - ★ Attributs répartis dans les tables `Client` et `Sejour`.

```
SELECT   nom, station
FROM     Client, Sejour
WHERE    id = idClient
```

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes sur plusieurs tables**

  - **Jointures**
    - ★ Problèmes d'**ambiguïté**
    - ★ Attribut partagé par plusieurs tables impliquées dans la jointure

    - ★ Sol. 1: préfixation de l'attribut par le nom de la table

```
SELECT   Station.nomStation, tarif, libelle, prix
FROM     Station, Activite
WHERE    Station.nomStation = Activite.nomStation
```

    - ★ Sol. 2: Association d'un synonyme, et utilisation en tant que préfixe.

```
SELECT   S.nomStation, tarif, libelle, prix
FROM     Station S, Activite A
WHERE    S.nomStation = A.nomStation
```

Le langage SQL

Le langage de manipulation de données (DML) SQL

- **Requêtes sur plusieurs tables**

  - **Jointures**

    - ★ Problèmes d'**ambiguïté**

      Sol. 2 (Utilisation de synonymes): Indispensable dans certaines situations: jointure d'une relation avec elle-même

    - ★ Exemple: *Couples de stations situées dans la même région.*

```
SELECT   S1.nomStation, S2.nomStation
FROM     Station S1, Station S2
WHERE    S1.region = S2.region
```

Le langage SQL

- **Requêtes sur plusieurs tables**
  - **Union, intersection et différence** UNION
    - ★ **Conditions:** même nombre de colonnes et même type d'attributs
    - – *Tous les noms de région dans la base*

```
SELECT   region   FROM   Station
  UNION
SELECT   region   FROM   Client
```

    - – *Régions où l'on trouve à la fois des clients et des stations*

```
SELECT       region   FROM   Station
  INTERSECT
SELECT       region   FROM   Client
```

    - – *Régions où l'on trouve des stations, mais pas de clients*

```
SELECT    region   FROM   Station
  EXCEPT
SELECT    region   FROM   Client
```

Le langage SQL

- **Requêtes imbriquées**
  - Expression de conditions sur des *relations*.
  - **Exemple:** *Noms des stations où ont séjourné des clients parisiens*
    - ★ Résultat avec une jointure classique.

```
SELECT   station
FROM     Sejour, Client
WHERE    id = idClient AND ville = 'Paris'
```

    - ★ Résultat avec une requête imbriquée.

```
SELECT   station
FROM     Sejour
WHERE    idClient IN   (SELECT id FROM Client
                        WHERE ville = 'Paris')
```

      - IN peut être remplacé par = si **on est sûr que la sous-requête ramène un et un seul tuple**.

Le langage SQL

- **Requêtes imbriquées**
  - Conditions que l'on peut exprimer sur une relation R construite avec une requête imbriquée (peuvent être préfixée par `NOT` pour la négation):

1. `EXIST R`
   - Renvoie `TRUE` si R n'est pas vide, `FALSE` sinon.

2. `t IN R`
   - Où t est un tuple de même type que R. `TRUE` si $t \in R$, `FALSE` sinon.

3. `v cmp ANY R`
   - Où *cmp* est un comparateur SQL ($<, >, =, ...$). Renvoie `TRUE` si la comparaison avec *au moins un* des tuples de la relation R renvoie `TRUE`.

4. `v cmp ALL R`
   - Où *cmp* est un comparateur SQL ($<, >, =, ...$). Renvoie `TRUE` si la comparaison avec *tous* les tuples de la relation R renvoie `TRUE`.

Le langage SQL

- **Requêtes imbriquées**
  - ⋆ *Où (station, lieu) ne peut-on pas faire du ski ?*

```
SELECT   nomStation, lieu
FROM     Station
WHERE    nomStation NOT IN   (SELECT nomStation FROM Activite
                              WHERE libelle = 'Ski')
```

  - ⋆ *Quelle station pratique le tarif le plus élevé ?*

```
SELECT   nomStation
FROM     Station
WHERE    tarif >= ALL   (SELECT tarif FROM Station)
```

  - ⋆ *Station où l'on pratique une activité au même prix qu'à Santalba ?*

```
SELECT   nomStation,   libelle
FROM     Activite
WHERE    prix IN       (SELECT prix FROM Activite
                        WHERE nomStation = 'Santalba')
```

Le langage SQL

- **Agrégation**
  - Expression de conditions sur des *groupes de tuples*.
  - Constitution du résultat par *agrégation de valeurs* dans chaque groupe.

    SQL fournit:
    - ★ Le moyen de partitionner une relation une relation en *groupes* selon certains critères.
    - ★ Le moyen d'exprimer des conditions sur ces groupes
    - ★ Des fonctions d'agrégation.

    Il existe un groupe par défaut: c'est la relation toute entière.

  - **Fonctions d'agrégation**

    Fonctions qui s'appliquent à une colonne, en général de type numérique

    - ★ `COUNT` qui compte le nombre de valeurs *non nulles*.
    - ★ `MAX` et `MIN`.
    - ★ `AVG` qui calcule la moyenne des valeurs de la colonne.
    - ★ `SUM` qui effectue le cumul.

Le langage SQL

- **Agrégation**
  - **Fonctions d'agrégation** - Exemples.

```
SELECT   COUNT(nomStation), AVG(tarif), MIN(tarif), MAX(tarif)
FROM     Station
```

  - **Remarque:** on ne peut pas utiliser simultanément dans la clause SELECT des fonctions d'agrégation et des noms d'attributs (sauf dans le cas d'un GROUP BY).
  - Pourquoi la requête suivante est-elle incorrecte ?

```
SELECT   nomStation, AVG(tarif)
FROM     Station
```

  - *Combien de places a réservé Mr Kerouac pour l'ensemble des séjours ?*

```
SELECT   SUM(nbPlaces)
FROM     Client, Sejour
WHERE    nom = 'Kerouac' AND id = idClient
```

Le langage SQL

- **Agrégation**
  - **La clause** `GROUP BY`
    - ★ Construction de groupes en associant les tuples partageant la même valeur pour une ou plusieurs colonnes.
    - ★ *Afficher les régions avec le nombre de stations*

```
SELECT     region, COUNT(nomStation)
FROM       Station
GROUP BY   region
```

  - Il n'est pas nécessaire de faire figurer tous les attributs du `GROUP BY` dans la clause `SELECT`
    - ★ *On souhaite consulter le nombre de places réservées par client.*

```
SELECT     nom , SUM(nbPlaces)
FROM       Client, Sejour
WHERE      id = idClient
GROUP BY   id, nom
```

Le langage SQL

- **Agrégation**
  - **La clause HAVING**
    - ★ Pour faire porter des *conditions sur les groupes*.
    - ★ WHERE ne peut exprimer des conditions que sur les tuples pris un à un.
    - ★ HAVING est pour le groupe ce que WHERE est pour le tuple.

    - ★ *On souhaite consulter le nombre de places réservées, par client, pour les clients ayant réservé plus de 10 places.*

```
SELECT     nom , SUM(nbPlaces)
FROM       Client, Sejour
WHERE      id = idClient
GROUP BY   id, nom
HAVING     SUM(nbPlaces) >= 10
```

    - ★ La condition porte sur une propriété de l'ensemble des tuples du groupe.
    - ★ La clause HAVING est exprimée sur le résultat de fonctions d'agrégation.

Le langage SQL

- **Mises-à-jour**
  - **Insertion**
    - ★ Elle s'effectue avec la commande `INSERT`. Syntaxe:

```
INSERT INTO  R(A1, A2, ..., An) VALUES (v1, v2, ..., vn)
```

  - `R`: nom d'une relation.
  - `A1, A2, ... An` sont les noms des attributs dans lesquels on veut placer une valeur. Les autres attributs seront donc à `NULL` (ou à la valeur par défaut). Tous les attributs spécifiés `NOT NULL` (et sans valeur par défaut) doivent figurer dans la clause `SELECT`.
  - `v1, v2, ..., vn` sont les valeurs des attributs.

```
INSERT INTO  Client (id, nom, prenom) VALUES (40, 'Moriarty', 'Dean')
```

```
INSERT INTO  R VALUES (v1, v2, ..., vn)
```

```
INSERT INTO  Sites (lieu, region)
  SELECT lieu, region FROM Station
```

Le langage SQL

- **Mises-à-jour**
  - **Destruction**
    - ★ Elle s'effectue avec la clause `DELETE`. Syntaxe:

```
DELETE   FROM R
WHERE    condition
```

    - ★ *Destruction de tous les clients dont le nom commence par 'M'*

```
DELETE   FROM Client
WHERE    nom LIKE 'M%'
```

Le langage SQL

- **Mises-à-jour**
  - **Modification**
    - ★ Elle s'effectue avec la clause `UPDATE`. Syntaxe:

```
UPDATE   R SET A1=v1, A2=v2, ..., An=vn
WHERE    condition
```

- ★ *Augmenter le prix des activités de la station Passac de 10%*

```
UPDATE   Activite
SET      prix = prix × 1.1
WHERE    nomStation = 'Passac'
```

Remarque importante:

- ★ Toutes les mises-à-jour ne deviennent définitives qu'à l'issue d'une validation par *commit*.
- ★ Elles peuvent être annulées par *rollback*

FIN DU COURS