

---

## Contraintes d'Intégrité dans le Modèle Relationnel

---

Marc Plantevit



`marc.plantevit@liris.cnrs.fr`

# Objectifs

## Structure du modèle relationnel (cours préc.)

- Permet de modéliser la réalité à un certain niveau de granularité :
  - les attributs vont décrire les objets ;
  - on peut séparer les données dans différentes relations avec des noms explicites ;
- Largement insuffisant pour représenter plus finement les différents aspects des données réelles.
- L'incapacité à représenter des *méta-données* conduit sans conteste à un certain nombre de problèmes.

## Contraintes d'intégrité

- Un cadre pour ajouter une sémantique au modèle relationnel.
- Propriétés supposées être satisfaites par toutes les instances d'un schéma de bases de données.
- Ex. : numéro de sécurité sociale (clé).

# Plan

---

- 1 Motivations
- 2 Dépendances fonctionnelles et clés
- 3 Dépendances d'inclusion

<i>Films</i>	<i>Titre</i>	<i>Metteur</i>	<i>Acteur</i>
	Les oiseaux	Hitchcock	Hedren
	Les oiseaux	Hitchcock	Taylor
	Bladerunner	Scott	Hannah
	Apocalypse Now	Coppola	Brando

<i>Programme</i>	<i>Ciné</i>	<i>Salle</i>	<i>Titre</i>	<i>Friandise</i>
	Rex	1	Les oiseaux	café
	Rex	1	Les oiseaux	popcorn
	Rex	2	Bladerunner	café
	Rex	2	Bladerunner	popcorn
	ArtC	1	Les oiseaux	thé
	ArtC	1	Les oiseaux	popcorn
	Cinoche	1	Les oiseaux	Coca Cola
	Cinoche	1	Les oiseaux	vin
	Cinoche	2	Bladerunner	Coca Cola
	Cinoche	2	Bladerunner	vin

TABLE: Exemple BD

# Introduction dépendances

- Le schéma en lui-même ne fait aucune restriction sur les données qui peuvent être stockées.
- Toutefois, l'application attendue de ce schéma peut impliquer plusieurs de ces restrictions.

## Exemple : les dépendances fonctionnelles (df)

- Un seul metteur en scène associé à chaque titre de film.
- Dans *Programme*, un seul titre de film est associé à un couple cinéma-salle donné.
- Les valeurs de certains des attributs d'un tuple déterminent de façon unique, ou *fonctionnellement*, les valeurs des autres attributs du tuple.

*Films* : Titre  $\rightarrow$  Metteur

*Programme* : Ciné Salle  $\rightarrow$  Titre.

Une relation  $I$  **satisfait** la dépendance fonctionnelle  $X \rightarrow Y$  ssi pour tout couple  $s, t$  de tuples de  $I$  on a :

$$\pi_X(s) = \pi_X(t) \Rightarrow \pi_Y(s) = \pi_Y(t).$$

Une des notions importantes de la théorie des dépendances est celle d'implication. On peut observer que toute relation qui satisfait la dépendance :

*Titre  $\rightarrow$  Metteur,*

satisfait également la dépendance :

*Titre, Acteur  $\rightarrow$  Metteur.*

On dit que la première df implique la seconde.

# Dépendance de clé

---

- Une *dépendance de clé* est une dépendance fonctionnelle

$$X \rightarrow U,$$

où  $U$  est l'ensemble de tous les attributs de la relation.

- $X$  est appelée *clé*.

Ex. :

*Titre, Acteur*  $\rightarrow$  *Titre, Acteur, Metteur*

est une dépendance de clé.

## Dépendances de jointure (dj)

- $(th, sc, ti, sn)$  est dans *Programme* si le cinéma  $th$  passe le film  $ti$  dans la salle  $sc$  et si le cinéma  $th$  propose la friandise  $sn$ .
- Intuitivement, on peut deviner une certaine **indépendance** entre les attributs *Salle*, *Titre* et l'attribut *Friandise*, pour une valeur donnée de *Ciné*.

Ex. :  $(Cinoche, 1, Les\ oiseaux, Coca\ Cola)$  et  $(Cinoche, 2, Bladerunner, vin)$  sont dans *Programme*,

⇒  $(Cinoche, 1, Les\ oiseaux, vin)$  et  $(Cinoche, 2, Bladerunner, Coca\ Cola)$  devraient également être présents

Plus précisément si une relation  $I$  possède cette propriété, alors on a :

$$I = \pi_{\text{Ciné}, \text{Salle}, \text{Titre}}(I) \bowtie \pi_{\text{Ciné}, \text{Friandise}}(I).$$

C'est un exemple simple de *dépendances de jointure* (dj), exprimée formellement par :

$$\text{Programme} : \bowtie [\{ \text{Ciné}, \text{Salle}, \text{Titre} \}, \{ \text{Ciné}, \text{Friandise} \}].$$

- Une *dj* peut faire intervenir plus de deux ensembles d'attributs !
- Une *dépendance multivaluée* (dmv) est un cas particulier de dj qui possède au plus deux ensembles d'attributs.

## Dépendances d'inclusion (di)

---

- Un troisième type de dépendances, appelées *dépendances d'inclusion (di)* ou « contraintes référentielles ».
- Entre deux relations.

Ex. : *Tout titre projeté actuellement (présent dans la relation Programme) est le titre d'un film (c'est-à-dire apparaissant dans la relation Films).*

$$\text{Programme}[\text{Titre}] \subseteq \text{Films}[\text{Titre}].$$

- Les di peuvent faire intervenir des *séquences* d'attributs de chaque côté.

# Un mécanisme formel pour exprimer des propriétés attendues des données stockées

Si on sait que la BD satisfait un ensemble de dépendances, alors cette information peut être utilisée :

- pour améliorer la conception d'un schéma ;
- pour protéger les données en se prémunissant contre certaines mise à jour erronées ;
- pour améliorer les performances.

# Conception du schéma et anomalies de mise à jour

Les dépendances sont utilisées pour fournir des informations sur la sémantique de l'application afin que le système puisse aider l'utilisateur à choisir, parmi tous les schémas possibles, le plus approprié.

Il existe plusieurs façons, pour un schéma, de ne pas être approprié.

## Informations incomplètes :

Supposons qu'il faille insérer le titre d'un nouveau film et son metteur en scène sans encore connaître les acteurs.

- Impossible avec le schéma décrit précédemment. **Anomalie d'insertion.**
- Phénomène analogue pour les **anomalies de suppression.**

## Redondance

- Le fait que *Coca Cola* puisse être trouvé au *Cinoche* est enregistré **plusieurs fois**.
- Supposons de plus que le directeur du *Cinoche* décide de vendre du *Pepsi Cola* au lieu du *Coca Cola*.
- Il faut modifier plusieurs tuples sinon violation de la dépendance de jointure. **Anomalie de modification possible**.
- Des anomalies d'insertion et de suppression sont également causées par la redondance.

## Schéma plus approprié :

- La relation *Films* pourrait être « décomposée » en deux relations *M-Metteur*[Titre, Metteur] et *M-Acteur*[Titre, Acteur] avec
  - la df *M-Metteur* : *Titre* → *Metteur*.
- *Programme* pourrait être décomposée en deux relations *ST-Programme*[Ciné, Salle, Titre] et *S-Programme*[Ciné, Friandise], avec
  - La df *ST-Programme* : *Ciné, Salle* → *Titre*.

## Dépendances des données servent également de filtres aux m.a.j

- Une proposition de m.a.j conduisant à une violation d'une dépendance  $\sigma$ , alors la m.a.j est rejetée.
- Le système supporte les transactions.
- Durant une transaction, la BD peut être dans un état inconsistant mais, à la fin de celle-ci, le système doit vérifier l'intégrité de la BD.
- Si les dépendances sont violées, alors la transaction entière est rejetée, sinon elle est acceptée (validée).

# Implémentation efficace et optimisation de requêtes

---

- La connaissance des propriétés structurelles des données stockées est utile à l'amélioration des performances d'un système pour une application particulière.
- La satisfaction des dépendances conduit à une grande variété d'alternatives pour les structures de stockage et d'accès.

Ex. : La satisfaction d'une  $df$  ou d'une  $dj$  implique qu'une relation peut être stockée physiquement sous forme décomposée.

Ex. ++ : La satisfaction d'une dépendance de clé peut être utilisée pour réduire l'espace d'indexation.

# Retour sur les dépendances

---

- *Les dépendances fonctionnelles*
- *Les dépendances d'inclusion*
- Les dépendances de jointure
- Les dépendances multi-valuées

# Plan

---

- 1 Motivations
- 2 Dépendances fonctionnelles et clés
- 3 Dépendances d'inclusion

- La forme la plus fréquemment rencontrée de dépendances.
- Sont à l'origine de l'approche par décomposition des schémas.

## Classe de dépendances

On doit définir la syntaxe et la sémantique des dépendances concernées.

- La **syntaxe** : c'est le langage logique autorisé pour définir une contrainte. C'est la "*forme*" de la contrainte.
- La **sémantique** : ensemble de conditions devant être remplies pour que la contrainte soit *satisfaite* (c'est à dire juste) dans les données. C'est le *sens* de la contrainte.

## Définition DF

La syntaxe des dépendances fonctionnelles est définie ainsi :

### Dépendance fonctionnelle (DF)

Une *dépendance fonctionnelle (DF)* sur un schéma de relation  $R$  est une expression de la forme  $R : X \rightarrow Y$  (ou simplement  $X \rightarrow Y$  lorsque  $R$  est implicite), où  $X, Y \subseteq R$ .

- Une DF  $X \rightarrow Y$  est dite *triviale* si  $Y \subseteq X$  et *standard* si  $X \neq \emptyset$ .

- Une DF est donc une contrainte définie sur un schéma.
- Une DF est dite *valide* sur un schéma  $R$  si elle est satisfaite dans toute relation  $r$  sur  $R$ , selon de la définition suivante pour la satisfaction :

### Satisfaction d'une DF dans une relation

Soit  $r$  une relation sur  $R$ . Une DF  $R : X \rightarrow Y$  est *satisfaite* dans  $r$  si  $\forall t_1, t_2 \in r$  on a  $t_1[X] = t_2[X] \implies t_1[Y] = t_2[Y]$ .

### Exemple

Si  $X \rightarrow Y$  est satisfaite dans une relation  $r$ , on dit aussi que  $X$  *détermine*  $Y$  dans  $r$ . De façon intuitive, on définit une DF  $X \rightarrow Y$  pour exprimer le fait que lorsqu'on connaît la valeur de  $X$ , alors on peut déterminer (en parcourant la relation) de façon certaine la valeur de  $Y$ .

La notion de *clé d'une relation* est très connue par les utilisateurs du modèle relationnel.

## Clé

Une clé peut-être définie de deux manières :

- une clé est un ensemble d'attributs qui ne prend jamais deux fois la même valeur (pas de doublons dans les relations) ;
- A l'aide des DF : une clé est un ensemble d'attributs qui détermine (au sens des DF) tous les autres attributs de la relation.

Ces deux définitions sont rigoureusement équivalentes.

*Une clé primaire* est simplement une clé parmi les autres, choisie par le concepteur pour sa simplicité ou son aspect naturel.

# Plan

---

- 1 Motivations
- 2 Dépendances fonctionnelles et clés
- 3 Dépendances d'inclusion

Les DI se différencient des DF sur plusieurs points.

- 1 ● Peuvent être définies entre attributs de **relations différentes** et possèdent un caractère plus **global**.
- 2 ● Les DI sont définies non pas entre deux ensembles quelconques d'attributs, mais **entre deux séquences d'attributs de même taille**.

★ L'ordre des attributs est donc très important pour les DI!!!

# Syntaxe

La syntaxe (forme) des DI est la suivante.

## Dépendance d'inclusion

Soit  $\mathbf{R}$  un schéma de base de données. Une dépendance d'inclusion sur  $\mathbf{R}$  est une expression de la forme

$$R[X] \subseteq S[Y],$$

où  $R, S \in \mathbf{R}$ ,  $X$  et  $Y$  sont des séquences d'attributs distincts respectivement de  $R$  et de  $S$ , et  $|X| = |Y|$ .

Une DI est satisfaite dans une base de données si toutes les valeurs prises par la partie gauche apparaissent dans la partie droite.

## Satisfaction d'une DI

Soit  $d = \{r_1, r_2, \dots, r_n\}$  une base de données sur un schéma  $\mathbf{R} = \{R_1, \dots, R_n\}$ . Une dépendance d'inclusion  $R_i[X] \subseteq R_j[Y]$  sur  $\mathbf{R}$  est *satisfaite* dans  $d$ , noté  $d \models R_i[X] \subseteq R_j[Y]$ , si  $\forall t_i \in r_i, \exists t_j \in r_j \mid t_i[X] = t_j[Y]$  (de manière équivalente,  $\pi_X(r_i) \subseteq \pi_Y(r_j)$ ).

## Exemple (1)

---

Supposons des schémas de relation pour décrire les modules :

$$MODULE = \{\underline{NUMMODULE}; INTITULE; DESC\}$$

et un schéma de relation pour décrire les séances de cours :

$$SEANCE = \{\underline{DATE}; \underline{NUMMODULE}; NUMSALLE\}$$

Pour forcer que les numéros de modules dans les séances soient bien des modules qui existent, on devra alors définir la contrainte :

## Exemple (1)

---

Supposons des schémas de relation pour décrire les modules :

$$MODULE = \{\underline{NUMMODULE}; INTITULE; DESC\}$$

et un schéma de relation pour décrire les séances de cours :

$$SEANCE = \{\underline{DATE}; \underline{NUMMODULE}; NUMSALLE\}$$

Pour forcer que les numéros de modules dans les séances soient bien des modules qui existent, on devra alors définir la contrainte :

$$SEANCE[NUMMODULE] \subseteq MODULE[NUMMODULE]$$

## Exemple (2)

---

Supposons maintenant un schéma de relation EMPRUNTVIDEO modélisant les réservations de vidéos pour les séances, sous la forme :

$$\text{EMPRUNTVIDEO} = \{ \text{DATE}, \text{NUMMODULE}; \text{NUMPROF}; \text{NUMVIDEO} \}$$

Pour assurer la cohérence de la base, on doit préciser que les vidéos doivent être réservés pour des séances existantes dans la base ; on définira alors la DI :

## Exemple (2)

---

Supposons maintenant un schéma de relation EMPRUNTVIDEO modélisant les réservations de vidéos pour les séances, sous la forme :

$$\text{EMPRUNTVIDEO} = \{ \text{DATE}, \text{NUMMODULE}; \text{NUMPROF}; \text{NUMVIDEO} \}$$

Pour assurer la cohérence de la base, on doit préciser que les vidéos doivent être réservés pour des séances existantes dans la base ; on définira alors la DI :

$$\text{EMPRUNTVIDEO}[\text{DATE}, \text{NUMMODULE}] \subseteq \text{SEANCE}[\text{DATE}, \text{NUMMODULE}]$$

Si on inverse les attributs à gauche par exemple, la DI change complètement de signification !

Une **contrainte d'intégrité référentielle** est une DI dont la partie droite est une clé (Un attribut (ou ens. d'attributs) d'une relation apparait comme clé d'une autre relation). ;

- La partie gauche d'une contrainte d'intégrité référentielle est appelée *clé étrangère*

## Exemple

les DI ne définissent pas toujours des clés étrangères!!!

- Il suffit d'imaginer qu'on souhaite imposer que tous les cours possèdent au moins une séance dans l'année : on définira alors une DI :

$$COURS[NUMCOURS] \subseteq SEANCE[NUMCOURS]$$

- Tous les cours apparaîtront au moins une fois dans la relation des séances ;
- *NUMCOURS* n'est pas une clé de *SEANCE* (on imagine difficilement que tous les cours n'aient qu'une seule séance!)
- Donc ce n'est pas une contrainte d'intégrité référentielle.

## Exemple

---

les DI ne définissent pas toujours des clés étrangères!!!

- Il suffit d'imaginer qu'on souhaite imposer que tous les cours possèdent au moins une séance dans l'année : on définira alors une DI :

$$COURS[NUMCOURS] \subseteq SEANCE[NUMCOURS]$$

- Tous les cours apparaîtront au moins une fois dans la relation des séances ;
- *NUMCOURS* n'est pas une clé de *SEANCE* (on imagine difficilement que tous les cours n'aient qu'une seule séance!)
- Donc ce n'est pas une contrainte d'intégrité référentielle.

# Fin de la séance !!!

## Ce qu'il faut retenir

- Les dépendances apportent de la sémantique au modèle relationnel.
- Les différents types de dépendances.
- Les dépendances fonctionnelles :
  - définition (syntaxe),
  - satisfaction (sémantique)
- ⇒ Etant donnée une instance  $r$  de BD, être capable de dire si  $r$  vérifie une df  $X \rightarrow Y$ .
- ++ Etant donnée une instance  $r$  de BD, exhiber toutes les df valides.
- clé & cie
- dépendances d'inclusion
  - définition (syntaxe),
  - satisfaction (sémantique)