SMI6U05L : Bases de données
Le modèle relationnel

Luigi Santocanale
LIS, Aix-Marseille Université

Contenu basé sur le cours de Rémi Eyraud

Plan

Plan

Historique

- Travaux de Edgar F. Codd 1970
- Codd (IBM) :
  langage SEQUEL (Structured English Query Language)
- Oracle :
  langage SQL (Structured Query Language)

Premiers objectifs

- Idée : utiliser un modèle ensembliste pour décrire et manipuler un ensemble d'enregistrements.
- Objectifs :
  - Indépendance entre programmes d'application et représentation interne
  - Base théorique solide pour la cohérence et la non-redondance des données.

SGBDR

- SGBD Relationnel : respecte les 12 règles de Codd (voir l'appendice).
- Principales raisons du succès :
  - simplicité d'utilisation
  - bases théoriques solides
  - normalisations (standardisation) de SQL (1986, 1992, 1999, 2003, 2008, 2011, 2016).

Avant de commencer...

- Dans ce chapitre : modèle relationnel.
  Il s'agit de l'ensemble des principes théoriques supportant le développement des SGBDR.
- La maîtrise de ce domaine, avant même SQL, facilite l'adaptation à TOUS les SGBDR et la réalisation de bonne conception de BD
  C'est un minimum pour un concepteur/administrateur.
- Attention : le modèle relationnel n'est pas SQL.

Plan

Structure des données

- Vocabulaire simple
- Plusieurs termes pour le même concept car :
  - théorie/implémentation
  - informatique/mathématique

  Ex. : tuple = ligne = enregistrement
- Théorie relationnelle fondée sur la théorie des ensembles.
- 3 briques principales : domaine, attribut, relation.

Domaine

- Def. Un domaine est
  - un ensemble de valeurs,
  - caractérisé par un nom.
- Ensemble nommé
- Idée : les données peuvent prendre leurs valeurs dans cet ensemble.
- Domaine $\neq$ type de données (mais parfois confondu) : pas (forcément) d'opérations applicables aux valeurs.

Domaine (suite)

- Domaine défini en extension :
  via l'ensemble de toutes les valeurs qui le composent.
  Exemple : Devise = $\{ \textit{dollar}, \textit{euro}, \textit{livre}, \textit{yen} \}$

- Domaine défini en intention :
  via les propriétés respectées par ses valeurs (éléments).
  Exemple : N_plus = les entiers *positifs*

- Extension vs. intension : voir
  - axiome d'extensionalité dans la théorie des ensemble
  - egalité de Leibniz, axiomes de compréhension

Relation

- Produit cartésien d'une liste de domaines $D_1, D_2, ..., D_n$

$$D_1 \times D_2 \times \ldots \times D_n := \{ (v_1, v_2, \ldots, v_n) \mid v_i \in D_i, \text{ for } i = 1, \ldots, n \}.$$

- Exemple :

$$\begin{aligned} \mathit{Pays} &= \{ \mathit{France}, \mathit{Italie}, \mathit{Japon} \}, \\ \mathit{Devise} &= \{ \mathit{euro}, \mathit{yen} \}, \\ \mathit{Pays} \times \mathit{Devise} &= \{ (\mathit{France}, \mathit{euro}), (\mathit{France}, \mathit{yen}), (\mathit{Italie}, \mathit{euro}), \\ &\qquad (\mathit{Italie}, \mathit{yen}), (\mathit{Japon}, \mathit{euro}), (\mathit{Japon}, \mathit{yen}) \}. \end{aligned}$$

Relation

- Def. Une relation est
  - un sous-ensemble du produit cartésien (nommé) d'une liste finie de domaines,
  - caractérisée par un nom.
- Une relation est donc un ensemble de tuplets avec un nom.
- Exemple :

$$DeviseCourante = \{(France, euro), (Italie, euro), (Japon, yen)\}.$$

- La définition d'une relation vise à définir l'espace de définition des tuplets (pas forcément en dimension 2).

Relation

- Famille d'ensembles indexée par A :

$$\{(E_a \mid a \in A\}$$

- Pour chaque $a \in A$, le couple (a, E_a) est un domaine.
- Produit cartésien nommé d'un famille de domaines

$$\prod_{a \in A} E_a = \{ f : A \to \bigcup_{a \in A} E_a \mid f(a) \in E_a, \text{ pour tout } a \in A \}.$$

Exemple :

Devise	Pays
euro	France

$$f(x) = \begin{cases} euro, & x = Devise, \\ France, & x = Pays, \end{cases}$$

Relation (fin)

- Pour représenter visuellement le contenu d'une relation on utilise les tables.
- Exemple :

DeviseCourante	Devise	Pays
	euro	France
	euro	Italie
	yen	Japon

- Chaque ligne correspond à un tuplet (donc à une donnée),
- Chaque colonne donne un sous-ensemble du domaine.
- En SQL, le mot 'table' remplace 'relation'.

Attribut

- Def. Un attribut est
  - une colonne d'une relation
- Un attribut est caractérisé par un nom unique dans cette relation (= le nom du domaine)
- Attention : la colonne n'est pas le domaine (c'est plutôt un sous-ensemble).
- Un domaine est potentiellement infini, une colonne est toujours finie.
- Par les noms, un même ensemble peut être impliqué plusieurs fois dans la même relation.
- On peut acceder à une colonne via les noms. On a pas besoin d'utiliser les indices.

Tuple(t)s et représentation des relations

- Une ligne d'une relation est un *n*-uplet de valeurs.
  Chaque valeur est accédée via le nom de l'attribut qu'elle value.
  On les appelle des tuples (lignes en SQL).
- Une relation de *n* domaines correspond à un sous espace d'un produit cartésien *n* dimensionnel.
- Elle peut donc être représentée par un diagramme à *n* dimensions dont chaque axe correspond à un domaine.
  Un tuple est un point dans cet espace.

Schéma de relation

- Une relation peut evoluer avec le temps.
- Sa définition/structure est figée en terme de domaines et d'attributs.

  $\rightarrow$ structure fixe vs contenu évolutif.

- Intention d'une relation (description par les attributs) vs extension (ensemble des tuples).
- Schéma de relation : description de l'intention (structure) d'une relation, i.e. de l'espace dans lequel évoluera l'extension, ainsi que les éventuelles contraintes.
- Exemple :

```
DeviseCourante(Devise,Pays)
```

Schéma, une définition formelle

- Def. Soit (n, R) une relation telle que $R \subseteq \prod_{a \in A} E_a$. Le schéma de (n, R) et le couple

$$(n, \{ E_a \mid a \in A \}).$$

- Exemple :

$$\begin{aligned} A &= \{ \mathit{Devise}, \mathit{Pays} \} \\ E_{\mathit{Devise}} &= \{ \mathit{euro}, \mathit{yen}, \mathit{livre} \} \\ E_{\mathit{Pays}} &= \{ \mathit{France}, \mathit{Italie}, \mathit{Japon} \} \\ n &= \mathit{DeviseCourante} \end{aligned}$$

- Dans le jargon informatique, on ecrit cela par :

```
DeviseCourante(Devise,Pays)
```

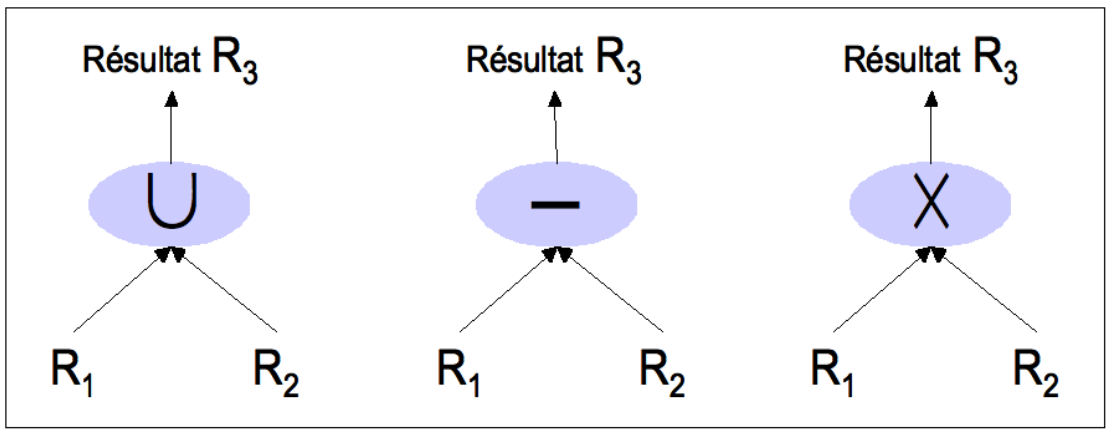
Plan

Opérations de base en Algèbre Relationnelle

- L'algèbre relationnelle est un ensemble d'opérateurs qui agissent sur des relations pour créer d'autres relations.
- Tout est relation : même si la relation résultante d'une opération n'a qu'une colonne et qu'une ligne, c'est encore un relation.
- Maîtrise de l'algèbre relationnel : essentiel pour la compréhension du langage SQL et des SGBDRs.
- 6 opérations de bases associées à une représentation

Opérations ensemblistes

- 3 opérations directement adaptées de la théorie des ensembles : union, différence, produit cartésien.
- Ne prennent pas en compte les contraintes de clés.
- Le résultat est une relation
  (pas de tuplets/lignes doublées, l'ordre ne compte pas, ...)



Union

- Def. L'union des relations R_1 et R_2 de mêmes schémas, est la relation R_3, toujours de même schéma, telle que l'ensemble des tuples de R_3 est l'union (sans doublons) de l'ensemble des tuples de R_1 et de l'ensemble des tuples de R_2.
- Opération commutative ! ! !
- Notations :
  - $R_1 \cup R_2$,
  - `UNION(`R_1`,` R_2`)`
- Note : les relations opérandes d'opérations d'algèbre relationnel ne sont pas forcément stockés en tant que telle dans la base.

Union : exemple

Acteur1

nom	prénom	datenaissance
Braschi	Nicoletta	10/08/1960
Depardieu	Gérard	27/12/1948
Benigni	Roberto	27/10/1952
Casta	Laetitia	11/05/1978

Acteur2

nom	prénom	datenaissance
Blanc	Michel	16/04/1952
Waits	Tom	NULL
Clavier	Roberto	06/05/1952
Casta	Laetitia	11/05/1978
Depardieu	Gérard	27/12/1948

Acteur1 ∪ Acteur2

nom	prénom	datenaissance
Braschi	Nicoletta	10/08/1960
Depardieu	Gérard	27/12/1948
Benigni	Roberto	27/10/1952
Casta	Laetitia	11/05/1978
Blanc	Michel	16/04/1952
Waits	Tom	NULL
Clavier	Roberto	06/05/1952

Différence

- Def. La différence entre les relations R_1 et R_2 de mêmes schémas, est la relation R_3, de même schéma, telle que l'ensemble des tuples de R_3 est l'ensemble des tuples de R_1 auquel on a enlevé l'ensemble des tuples de R_3.
- La différence n'est pas commutative ! ! !
- Notations :
  - $R_1 \setminus R_2$
  - $R_1 - R_2$
  - `EXCEPT(`R_1 `,` R_2`)`

Différence : exemple

Acteur1	nom	prénom	datenaissance
	Braschi	Nicoletta	10/08/1960
	Depardieu	Gérard	27/12/1948
	Benigni	Roberto	27/10/1952
	Casta	Laetitia	11/05/1978

Acteur2	nom	prénom	datenaissance
	Blanc	Michel	16/04/1952
	Waits	Tom	NULL
	Clavier	Roberto	06/05/1952
	Casta	Laetitia	11/05/1978
	Depardieu	Gérard	27/12/1948

Acteur1 – Acteur2	nom	prénom	datenaissance
	Braschi	Nicoletta	10/08/1960
	Benigni	Roberto	27/10/1952

Produit cartésien de relations

- Def. Le *produit cartésien* entre les relations R_1 et R_2 de schémas quelconques, est la relation R_3 qui a pour schéma la concaténation de ceux de R_1 et R_2 et, pour extension, l'ensemble de toutes les combinaisons possibles entre les tuples de R_1 et ceux de R_2.
- Le produit cartésien de deux relations est une opération commutative (si on ne considère pas d'ordre sur les colonnes).
- Opération très courante : à la base des requètes de confrontations.
- Notations :
  - $R_1 \times R_2$
  - `TIMES(`R_1`,` R_2`)`
  - `PRODUCT(`R_1`,` R_2`)`

Produit cartésien : exemple

Pays	nom	capitale	monnaie
	Italie	Roma	3
	France	Paris	3
	Gabon	Libreville	6
	Bénin	Porto-Novo	6

X

Monnaie	num	nom
	1	Dollar US
	3	Euro
	6	Franc CFA

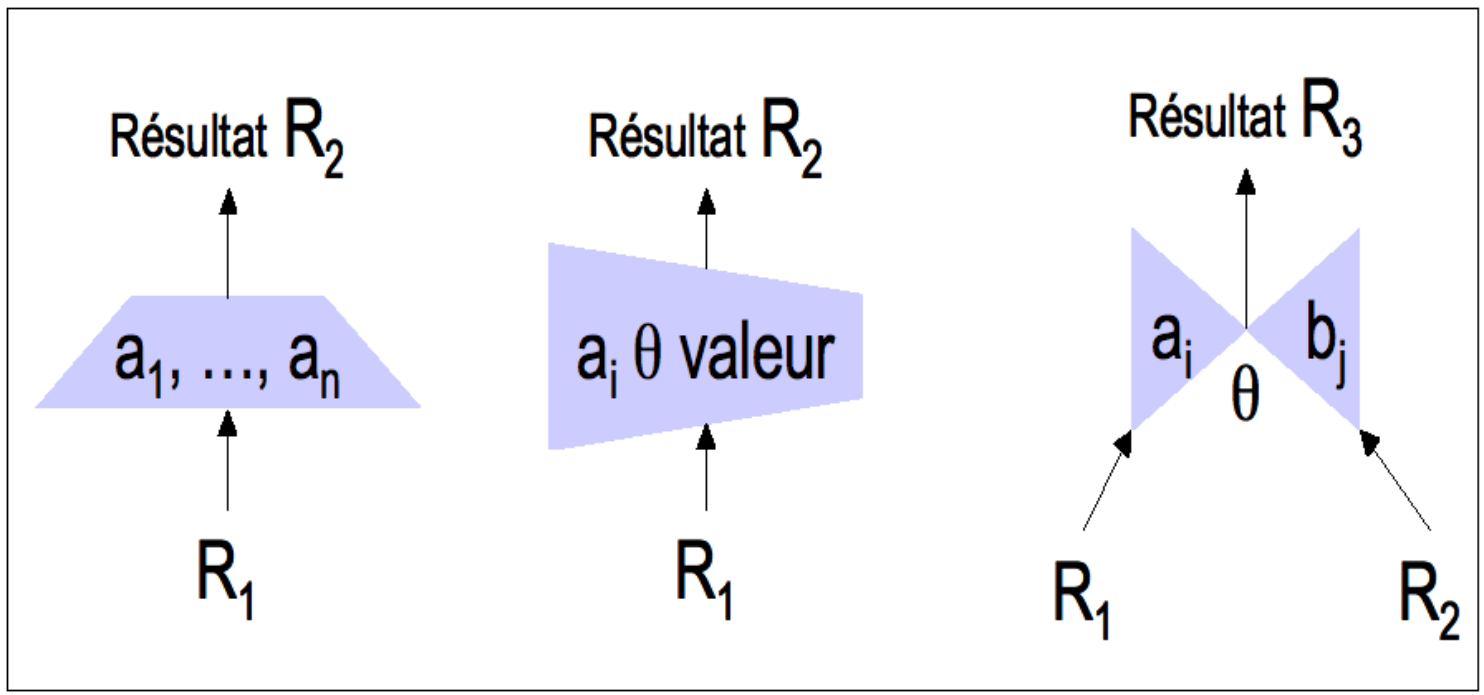
Pays P
X
Monnaie M

P.nom	capitale	monnaie	num	M.nom
Italie	Roma	3	1	Dollar US
France	Paris	3	1	Dollar US
Gabon	Libreville	6	1	Dollar US
Bénin	Porto-Novo	6	1	Dollar US
Italie	Roma	3	3	Euro
France	Paris	3	3	Euro
Gabon	Libreville	6	3	Euro
Bénin	Porto-Novo	6	3	Euro
Italie	Roma	3	6	Franc CFA
France	Paris	3	6	Franc CFA
Gabon	Libreville	6	6	Franc CFA
Bénin	Porto-Novo	6	6	Franc CFA

Attention : Préfixage pour 2 colonnes partageant le même nom.

Opérations propre de l'algèbre relationnelle

- 3 opérations spécifiques, essentielles à la mise en œuvre du modèle dans le cadre opératoire.
- Projection, Restriction, Jointure



Projection

- Permet de ne retenir que quelques attributs d'une relation.
- Def. Soit
  - R une relation de schéma $R(a_1, \ldots, a_n)$,
  - $B = \{ a_{i_1}, a_{i_2}, \ldots, a_{i_m} \} \subseteq \{ a_1, \ldots, a_n \}$.

  La projection de R sur les attributs B, est la relation R' dont le schéma est $R'(a_{i_1}, a_{i_2}, \ldots, a_{i_m})$ (avec même contraintes de domaine que R) et qui a la même extension que celle de R, mais dont les tuples ont moins attributs.
- Notations :
  - $\Pi_{(a_{i_1},\ldots,a_{i_m})}(R)$
  - $R[a_{i_1}, \ldots, a_{i_m}]$,
  - $\texttt{PROJECT}(R, a_{i_1}, \ldots, a_{i_m})$
- Pour $R \subseteq \prod_{a \in A} E_a$ et $B \subseteq A$, on a

$$\Pi_B(R) := \{ f \restriction_B \mid f \in R \}.$$

Projection : exemple

Acteur	nom	prénom	datenaissance
	Braschi	Nicoletta	10/08/1960
	Depardieu	Gérard	27/12/1948
	Benigni	Roberto	27/10/1952
	Casta	Laetitia	11/05/1978
	Blanc	Michel	16/04/1952
	Waits	Tom	NULL
	Clavier	Roberto	06/05/1952

(Acteur)[nom, prénom]	nom	prénom
	Braschi	Nicoletta
	Depardieu	Gérard
	Benigni	Roberto
	Casta	Laetitia
	Blanc	Michel
	Waits	Tom
	Clavier	Roberto

Restriction

- Réduit le nombre de tuples mais même nombre d'attributs.
- Def. Soit P un prédicat (critère logique) portant sur les attributs des tuples d'une relation R.
  La restriction de la relation R à P est la relation R' dont l'extension sont les tuples de R satisfaisant le prédicat P.
- Critère de restriction :
  - `<attribut> <comparaison> <valeur>`
    où `<comparaison>` dans $\{=, <, >, \leq, \geq, \neq\}$,
    + combinaison logique
  - Dialecte SQL :
    - `(LIKE, IN, NOT NULL, etc.)`
    - application de fonctions sur opérandes `(TO_DATE(), ...)`
- Notations :
  - $\sigma_{\texttt{critère}}(R)$
  - R`[critère]`
  - `RESTRICT(`R`, critère)`

Restriction : exemple

Acteur	nom	prénom	datenaissance
	Braschi	Nicoletta	10/08/1960
	Depardieu	Gérard	27/12/1948
	Benigni	Roberto	27/10/1952
	Casta	Laetitia	11/05/1978
	Blanc	Michel	16/04/1952
	Waits	Tom	NULL
	Clavier	Roberto	06/05/1952

(Acteur) [datenaissance < 1970]

nom	prénom	datenaissance
Braschi	Nicoletta	10/08/1960
Depardieu	Gérard	27/12/1948
Benigni	Roberto	27/10/1952
Blanc	Michel	16/04/1952
Clavier	Roberto	06/05/1952

Jointure

- Def. La jointure des deux relations R_1 et R_2 (de schémas quelconques) selon un critère logique donné portant sur au moins un attribut de chaque relation, est la relation R_3 contenant l'ensemble de
  - tous les tuples obtenus en concaténant chaque tuple de R_1 et chaque tuple de R_2
  - qui vérifient, ensemble, le critère logique.

- Cas simple :
  - a_1 et a_2 des attributs respectivement de R_1 et R_2,
  - critère de jointure de la forme : $a_1 \theta a_2$, où θ est un opérateur de comparaison (cf. restriction)
  - on parle de θ-jointure.
- Pas vraiment une opération de base : peut être définie à partir du produit cartésien et d'une restriction.

Jointure : exemples

Pays

nom	capitale	monnaie
Italie	Roma	3
France	Paris	3
Gabon	Libreville	6
Bénin	Porto-Novo	6

Monnaie

num	nom
1	Dollar US
3	Euro
6	Franc CFA

(a) **Pays P** ⋈ **Monnaie M**

P.monnaie = M.num

P.nom	capitale	monnaie	num	M.nom
Italie	Roma	3	3	Euro
France	Paris	3	3	Euro
Gabon	Libreville	6	6	Franc CFA
Bénin	Porto-Novo	6	6	Franc CFA

(b) **Pays P** ⋈ **Monnaie M**

nom	capitale	monnaie	num

Jointure (fin)

- Notations :
  - $R \bowtie_\theta R'$
  - `JOIN(R, R', θ)`
- Si R a n attributs et t tuplets, R' a n' attributs et t' tuplets, alors `JOIN(R, R', θ)` a $n+n'$ attributs et au max $t+t'$ tuplets.
- Relation fondamentale :

```
JOIN(R, R', θ) = RESTRICT(TIMES(R, R'), θ)
```

- Jointure naturelle :
  - jointure entre 2 relations avec critère d'égalité (equi-jointure) entre 2 attributs de même noms
  - fusion des colonnes de même nom(s).

Jointure et produit fibré (pullback), composition de relations

Au tableau

Plan

Expressions algébriques

- Le langage algébrique dérivé des opérateurs de l'algèbre relationnelle constitue un langage complet (= équivalent à la logique du premier ordre).
- La notion de relation reste essentielle.

Expressions algébriques : exemples

- Considérons la base constituée des relations :

```
Film(numfilm, titre, réalisateur, année, durée)
Acteur(numacteur, nom, prénom,dateNaissance)
Casting(numfilm, numacteur, personnage)
```

- Quel sont les films (titre, réalisateur) qui durent plus de deux heures ?

```
PROJECT(
  RESTRICT(Film, duree >= 120 ),
    titre, réalisateur)
```

- On peut utiliser des noms :

```
R1 = RESTRICT(Film, duree >= 120 )
RESULTAT = PROJECT(R1, titre, réalisateur)
```

Utilisation des noms dans les SGDBs

- Dans le jargon, 3 types de relation :
  - relations ni nommées ni stockées, appelées relations intermédiaires
  - relations nommées
    - stockées dans le SGBD
    - non stockées dans la SGDB

Expressions algébriques : exemples

- Quels sont les numéros des acteurs se prénommant Ryan ?

```
R1 = RESTRICT(Acteur, prénom = 'Ryan')
RESULTAT = PROJECT(R1,numacteur)
```

- Quels sont les films dont un des personnages est Astérix ?

```
R1 = NATURAL JOIN(Film, Casting)
-- ou
R1 = JOIN(Film, Casting, Film.numfilm = Casting.numfilm)
R2 = RESTRICT(R1, personnage ='Astérix')
RESULTAT = PROJECT(R2,
  numfilm, titre, réalisateur,année, durée)
```

Expressions algébriques : exemples

- Quels sont les acteurs (nom, prénom et mois de naissance) nés la même année ?

```
R1 = JOIN(Acteur A1, Acteur A2,
       YEAR(A1.dateNaissance) = YEAR(A2.dateNaissance))
R2=RESTRICT(R1, A1.numacteur ≠ A2.numacteur)
RESULTAT = PROJECT(R2,
       A1.nom, A1.prénom, MONTH(A1.dateNaissance),
       A2.nom, A2.prénom, MONTH(A2.dateNaissance))
```

Expressions algébriques : exemples

- Quels sont les acteurs qui ont joué dans un film réalisé par "Besson" et qui n'ont jamais joué dans un film réalisé par "Benigni" (nom et prénom) ?

```
R1 = NATURAL JOIN(Acteur, Casting)
R2 = RESTRICT(Film, réalisateur = 'Besson')
R3 = RESTRICT(Film, réalisateur = 'Benigni')
R4 = NATURAL JOIN(R1, R2)
R5 = NATURAL JOIN(R1, R3)
R6 = PROJECT(R4, nom, prénom)
R7 = PROJECT(R5, nom, prénom)
RESULTAT = EXCEPT(R6 , R7)
```

Expressions algébriques : exemples

- Quels sont les réalisateurs qui ont dirigé Emma Watson mais pas Emma Stone ?

```
R1 = RESTRICT(Acteur, nom = 'Watson' AND prénom = 'Emma')
R2 = RESTRICT(Acteur, nom = 'Stone' AND prénom = 'Emma')
R3 = NATURAL JOIN (Film, Casting)
R4 = NATURAL JOIN (R2, R3)
R5 = NATURAL JOIN (R1, R3)
R6 = PROJECT(R4, realisateur)
R7 = PROJECT(R5, realisateur)
RESULTAT = EXCEPT(R6,R7)
```

Plan

Introduction

- Une relation est un couple $(\text{nom}, \text{ensemble})$
  (car une relation est en sous-ensemble, donc un ensemble).
  Donc, c'est un nouveau domaine.
- Comment utiliser les valeurs du nouveau domaine, dans un autre relation ?
- Le "référencement" par "clés" réponde à cette question.
- Quoi faire si une relation évolue, et que elle est référencée par une autre relation ?

Gestion de l'intégrité des structures dans un SGBDR

- Les règles d'intégrité structurelle visent à assurer la cohérence des données.
- Par exemple :
  - des valeurs d'attribut,
  - contrôle des données manquantes.
- Ces règles sont énoncées par des assertions sur les relations et les attributs.
- Notion principale : clé, clef.
- Extrêmement important de déclarer toutes les contraintes. Permet ensuite de se reposer sur le SGBDR pour la cohérence.

Clé

- Relation = ensemble. Donc pas de doublons de tuples !
- Une clé est un ensemble (minimal) d'attributs dont la connaissance des valeurs permet d'identifier un tuple de façon unique au sein de la relation considérée.
- Une clé n'est pas forcément un attribut unique. C'est-à-dire, un tel ensemble minimal n'est pas nécessairement un singleton.

Clé primaire

- Soit R une relation de schéma $R(a_1, \ldots, a_n)$ et soit $K \subseteq \{a_1, \ldots, a_n\}$.
- Def. La fonction

$$\pi_K : R \to \Pi_K(R)$$

est définie par

$$\pi_K(t)(a) = t(a), \quad \text{pour tout } a \in K.$$

- Remarques :
  - On a $\pi_K(t) = t \restriction_K$.
  - $\pi_K(t)$ est le tuplet des valeurs données par t aux attributs composant K.
  - Notation utilisée dans le monde des BDs : $t(K) = \pi_K(t)$
- Def. Un sous-ensemble $K \subseteq \{a_1, \ldots, a_n\}$ est une clé de R si la projection $\pi_K : R \to \Pi_K(R)$ est injective.
  C'est-à-dire : pour tout $t, t' \in R$ tel que $t \neq t'$, $\pi_K(t) \neq \pi_K(t')$.

Notion de clé primaire

- Lors de la conception de la BD, pour toute relation on choisi un sous ensemble de (noms d'attributs) sensé être une clé : c'est la clé primaire.
  Ce choix est guidé par la sémantique de la relation.
- Par convention, dans le schéma d'une relation, on représente la clé primaire en la soulignant.

Clé primaire : exemples

- Ex. 1 :

```
Acteur(nom, prénom, numéro)
```

On aurait pu choisir `(nom, prénom)` mais la simplicité pousse à prendre numéro.

- Ex. 2 :

```
Casting(film, acteur, personnage)
```

  - Le même personnage peut être joué dans plusieurs films.
  - Un acteur peut jouer qu'un seul personnage dans 1 film donné. (Mais pas au théâtre ☺)
  - La clé est miminale.

Contraintes d'unicité

- La définition d'une clé primaire (dans une schéma) implique une contrainte d'unicité.
- On peut aussi spécifier que la valeur d'un (groupe d') attribut(s) doit être unique, c'est-à-dire que deux tuples ne peuvent pas avoir la même valeur pour cet attribut.
- Exemple :

```
Film(numFilm, titre, réalisateur, année)
```

`numFilm` est la clef primaire mais on peut (doit !) spécifier que le couple `(titre, réalisateur)` est unique (donc, est une clé).

Contrainte de référence

- Identifier/référencer un tuple : utiliser sa valeur de clé primaire. Exemple :

```
('Ryan', 'Gosling', 5)
```

  tuple de la relation `Acteur(Nom, Prénom, numéro)` est référencé par le numéro 5.
  `Acteur[numéro=5]` le désigne exactement.
- La valeur des attributs référentiels dans le tuple référent est la valeur de la clé primaire dans le tuple référencé.

Contrainte de référence : exemple

- ```
  Pays(nom, devise, capitale, superficie)
  Devise(numéro, nbpieces, nbbillets, denomination)

  Devise(4, 8, 8, 'euro')
  Pays('France', 4, 'Paris', 544 435)
  Pays('Grèce', 4, 'Athène', 131 957)
  ```

- `Pays` référence la monnaie via la clé primaire de la relation `Devise`.
  Les valeurs de l'attribut *devise* de la relation `Pays` sont les valeurs de l'attribut numéro de la relation `Devise`.
  ```

Clé étrangère

- Une contrainte référentielle exprime un lien obligatoire entre deux relations.
- Une clé étrangère est un groupe d'attributs (ex : *devise*) dans une relation R (ex : `Pays`) qui doit correspondre à la clé primaire (ex : `numéro`) d'une autre relation R' (ex : `Devise`).
- (Le schèma d') Une relation
  - possède une seule clé primaire,
  - peut spécifier plusieurs clés (contraintes d'unicités),
  - peut spécifier plusieurs clés étrangères.
- Convention d'écriture d'une clé étrangère : *italique*.

Clé étrangère (suite)

- Un attribut peut être à la fois une clé primaire et étrangère :

```
Film(numFilm, titre, année)
Acteur(numActeur,nom, prénom)
Casting(numFilm, numActeur,personnage)
```

- Pour chaque paire film/acteur il existe au plus une tuple les contenant dans Casting.
  A bijection prise, `Casting` est une relation entre `Film` et `Acteur`.

$$Casting \subseteq Film \times Acteur\,.$$

Exercice

Une relation peut s'auto-référencer via une clef étrangère. Ex. :

```
Personne(numéro, nom, prénom, père, mère)
```

- `père` et `mère` sont des clés étrangères de la relation `Personne`
- elles référencent la clé primaire de la relation `Personne`.

Retrouver l'arbre généalogique :

```
(12,'Martin','Pierre', NULL, NULL)
(56, 'Castard', 'Joséphine', NULL, NULL)
(77, 'Frandier', 'Ernest', NULL, NULL)
(74, 'Bisteur', 'Annie', NULL, NULL)
(89, 'Martin', 'Eric', 12, 74)
(91, 'Frandier', 'Alain', 77, 56)
(94, 'Bisteur', 'Virginie', NULL, 74)
(103, 'Frandier', 'Mélanie', 91, 94)
```

Contraintes référentielles et SGBDR

Contraintes référentielles pour le maintiens de la cohérence lors des actions en écriture :

- Insertion d'un tuple dans une relation avec clé étrangère.
  Le SGBD doit vérifier que la valeur donnée par ce tuple à la clé étrangère correspond bien à une valeur de clé primaire.
  Même vérification en cas de modification d'une valeur de clé étrangère dans un tuple existant
- Suppression d'un tuple référencé via une clé étrangère par un ensemble d'autres tuples, plusieurs options :
  - supprimer "en cascade" tous les tuples référençant, ou
  - donner la valeur NULL à l'attribut de clé étrangère ("set null") ou
  - refuser la suppression ("restrict").

  Par défaut, le SGBD ne fait rien ("no action", presque comme "restrict").

Contrainte de valeurs d'attribut

- On ne connaît pas forcément les valeurs de tous les attributs lors de l'insertion d'un tuple ;
  Parfois : attribut pas applicable à un tuple particulier (à éviter).
- Convention : valeur `NULL`
- Contrainte de non-vide : clé primaire ; autres attributs
- Contrainte de domaine : restriction sur les valeurs d'un attribut :
  - extension : `saison IN 'hiver', 'printemps', 'été', 'automne'`
  - restriction : `durée NOT NULL AND durée > 0 (CHECK)`

Plan

Les 12 règles de Codd

1. L'information est représentée de façon logique sous forme de tables.
2. Les données doivent être accessibles de façon logique par les tables, les clés primaires et les colonnes.
3. Les valeurs nulles doivent être traitées uniformément comme des "informations absentes".
   Il ne s'agit pas des chaînes vides, ni des blancs, ni des zéros.
4. Les méta-données doivent être stockées dans la base au même titre que les données normales.
5. Un langage *unique* doit permettre de définir les données, les vues sur ces données, les contraintes d'intégrité, les autorisations d'accès, les transactions, et enfin, la manipulation des données.

Les 12 règles de Codd

6. Les vues doivent refléter les mises à jour de leurs tables de base, et vice-versa.
7. Chaque action suivante doit pouvoir être réalisée par une et une seule action : retrouver, insérer, mettre à jour et supprimer une donnée.
8. Il doit y avoir séparation logique entre les opérations (interactives ou non), et le stockage physique des données et leurs méthodes d'accès.
9. Les opérations (interactives ou non) peuvent modifier le schéma de la base de données sans qu'elle n'ait à être recréée, et sans que les applications construites au dessus d'elle n'aient à être réécrites

10. Les contraintes d'intégrité doivent être disponibles, et stockées dans les meta-données et non pas dans un quelconque programme d'application.
11. Le langage de manipulation des données (LMD) ne doit pas se soucier d'où ni de comment les données sont stockées et/ou distribuées.
12. Le traitement d'une ligne doit respecter les mêmes règles et contraintes d'intégrité que les opérations portant sur des ensembles.