Bases de données

Chapitre 2

Modèle relationnel

Sommaire

1. Modèle relationnel
2. Passage du modèle conceptuel au modèle relationnel
3. Forme normale
4. Algèbre relationnelle
5. Le langage algébrique

1 – Modèle relationnel

Concepts
Caractéristiques
Contraintes
Schéma

Origine

- proposé en 1970 par Ted Codd

  "A Relational Model for Large Shared Data Banks"

- basé sur la *théorie des ensembles*

- *Données organisées en Tables*
  (sans préjuger de la façon dont les informations sont réellement stockées en machine)

- premier système

  1980 : ORACLE avec SQL/DS

Concepts

- Domaine
- Schéma
- Relation (ou table)
- Attribut (ou colonne)
- Tuple (ou ligne, enregistrement)
- Base de données
- SGBDR

Concept de *Domaine*

- Les domaines sont les ensembles de valeurs possibles dans lesquels sont puisées les données
- Ensemble de valeurs atomiques

AGES_DES_ETUDIANTS ensemble des entiers entre 18 et 35
NUMEROS_SS, ensemble des suites de 15 chiffres,
DEPARTEMENTS, ensemble {COMPTA, CLIENT, PRODUCTION},
VILLE, ensemble {Nice, Paris, Fort de France}
NOMS, ensemble des chaînes de 12 caractères

- Un domaine est défini par
  - un nom
  - une description logique
  - un type de données

Concept de *Schéma relationnel*

Un **schéma relationnel** *R* est caractérisé par un nom et une liste d'attributs A1, ..., An

et noté R(A1, A2, . . ., An)

Chaque attribut Ak est le rôle joué par un domaine Dk

On note dom(Ak)=Dk

On note aussi le schéma **R(A1:D1, . . ., An:Dn)**

Le *degré* du schéma est le nombre de ses attributs

Definition d'une Relation

Une **relation** de schéma R(A1, ..., An)

est une relation mathématique entre les domaines

dom(A1), ..., dom(An)

i.e

un sous-ensemble de leur produit cartésien

$R(A1, ..., An) \subseteq dom(A1) x ... X dom(An)$

Concept de *Relation*

Relation (ou *Table*)

Client(numero, nom, adresse,téléphone)

numéro	nom	adresse	téléphone
101	Durand	NICE	0493942613
106	Fabre	PARIS	
110	Aurand	PARIS	
125	Carré	MARSEILLE	0491258472

Attributs

Concept de *Relation*

Relation (ou *Table*)

Client(numero, nom, adresse,téléphone)

numéro	nom	adresse	téléphone
101	Durand	NICE	0493942613
106	Fabre	PARIS	
110	Aurand	PARIS	
125	Carré	MARSEILLE	0491258472

Tuples
Ou
n-uplets

Concept de *BD et SGBDR*

Base de données

- Système dont le schéma est un ensemble de schémas de relations
- Les occurrences sont l'ensemble des tuples de toutes les relations

SGBDR

- Logiciel supportant le modèle relationnel
- Manipule les données avec des opérateurs relationnels

Contraintes liées au modèle

- Pas d'ordonnancement parmi les tuples
  - Une relation est un ensemble (au sens mathématique) donc
    - **pas de notion d'ordre,**
    - **ni de doublons parmi les tuples**
    - Il faut un identifiant unique (clé primaire) pour chaque tuple
- Ordonnancement des valeurs d'un tuple
  - un tuple est une liste ordonnée de valeurs

Notion de *clé primaire*

pour toute relation R,
il existe un ensemble d'attributs K de R tel que
pour deux tuples t1 et t2 quelconques de R,
on ait t1[K]≠ t2[K]
Avec tn[k] valeur de l'attribut K du tuple tn

On appelle K une **clé candidate**

La clé candidate choisie pour identifiant est appelée **clé primaire**

Exemple

clés primaires

Client(numéro, nom, adresse, téléphone)

Vente(numéro, ref_produit, no_client, date)
Clés candidates : numéro
(ref_produit, no_client, date)

Produit(référence, marque, prix)

Notion de *clé étrangère*

Étant données une relation R1 et une relation R2,

on dit que l'ensemble R1.K d'attributs de R1 est
clé étrangère et **référence R2**
si
les attributs R1.K ont les mêmes domaines
que les attributs de la clé primaire de la relation R2

Contraintes d'intégrité (CI)

- Les contraintes d'intégrité sont les assertions qui doivent être vérifiées par les données contenues dans une base
- La gestion automatique des contraintes d’intégrité est l’un des outils central d’une base de données.
  - Intégrité de domaine
  - Intégrité de clé
  - Intégrité référentielle
  - Intégrité liée à l’application

Contraintes d'intégrité (CI)

- **Intégrité de domaine**
  - Contrôles des valeurs des attributs
  - Les valeurs d’une colonne appartiennent toutes au domaine correspondant
- **Intégrité de clé**
  - Valeur des clés doivent être unique et non-nulles
  - Unicité des clés
  - Unicité des tuples

Contraintes d'intégrité (CI)

Intégrité référentielle

La valeur d’une clé étrangère doit être

soit NULL

soit l’une des valeurs de la clé primaire de la table qu'elle référence

Que se passe-t-il quand on supprime un tuple primaire (de la table référencée)?

Que se passe-t-il quand on supprime un tuple primaire (de la table référencée)?

Pour chaque clé étrangère, il existe 3 possibilités :

- ***restriction***
  - interdiction de supprimer le tuple primaire s'il a des tuples associés
- ***cascade***
  - suppression du tuple primaire
  - suppression des tuples associés
- ***mise à NULL***
  - suppression du tuple primaire
  - la clé étrangère est mise à NULL dans les tuples associés

CI liées à l'application

- expriment des contraintes sémantiques

exemples :

La quantité vendue doit être inférieure à la quantité en stock

L'heure d'arrivée doit être ultérieure à l'heure de départ

Le salarié ne doit pas dépasser 35h

- spécifiées dans les programmes

ou par des **triggers**

2. Du modèle conceptuel au modèle relationnel

Transformation d'un TE
Transformation d'un TA

Transformation d’un TE

Transformation d'un TE

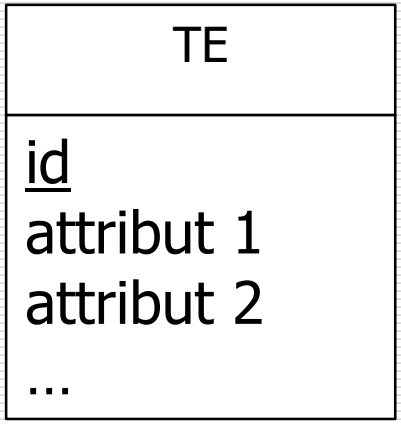
- Chaque type entité (TE) devient une **relation**
- Le nom du TE devient le nom de la relation
- Les attributs du TE deviennent les attributs de la relation
- Les identifiants du TE sont appelés **clés**
- En général la clé est soulignée

Transformation d'un TE

TE T → Relation T

Attribut du TE → Attribut de la relation

Identifiant du TE → Clé de la relation

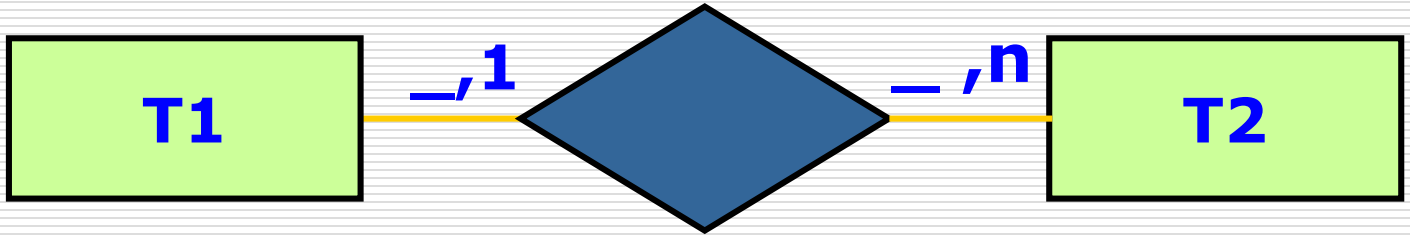


→ TE (id, attribut 1, attribut 2, …)

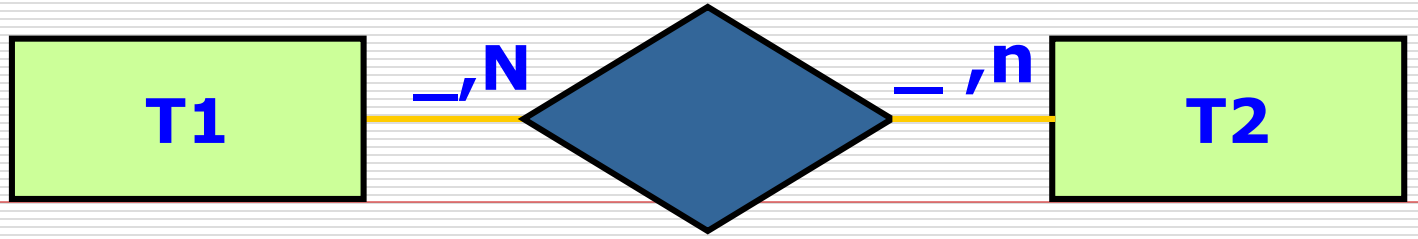
Transformation d’un TA

Transformation d'un TA

- 2 principaux cas à distinguer selon les cardinalités
  - - Cas d'une association de type (1:N)



  - - Cas d'une association de type (N:N)



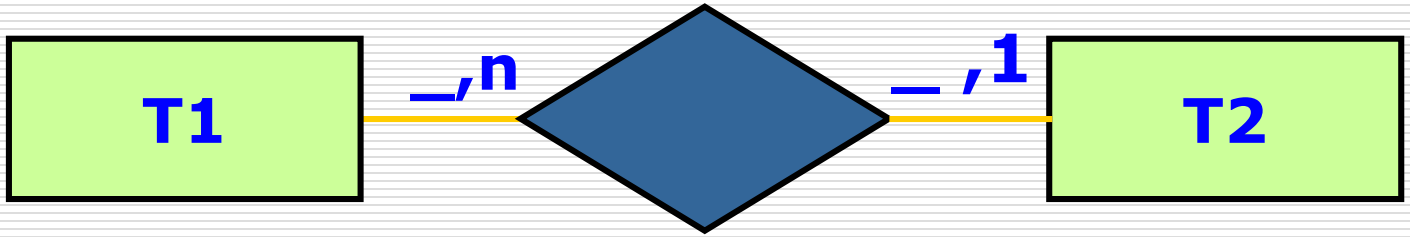
Transformation d'un TA

Association (1:N)

- Les deux TE impliqués sont évidemment des relations
- La clé primaire du TE qui porte la cardinalité (_, N) se retrouve comme attribut dans la relation qui représente le TE qui porte la cardinalité (_, 1)
- On parle alors de **Clé étrangère**
- La clé étrangère est généralement notée avec # devant le nom
- Les attributs de l'association (s'il y en a) sont également ajoutés

Transformation d'un TA

Association binaire de 1 à plusieurs (1:n



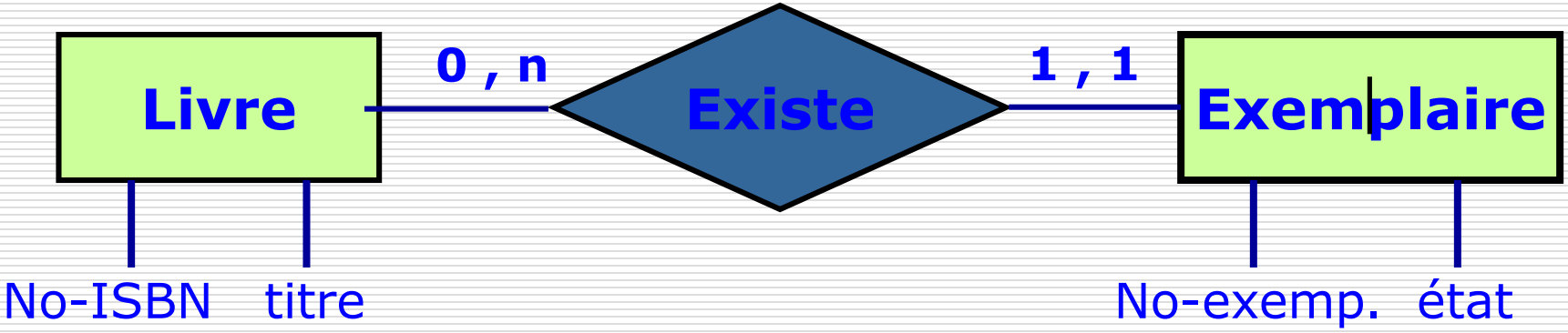
Ajout de l'identifiant de T2 dans la relation T1

T1(<u>K1</u>, …)

T2(<u>K2</u>, …, #K1) *#K1 : clé étrangère*

Transformation d'un TA

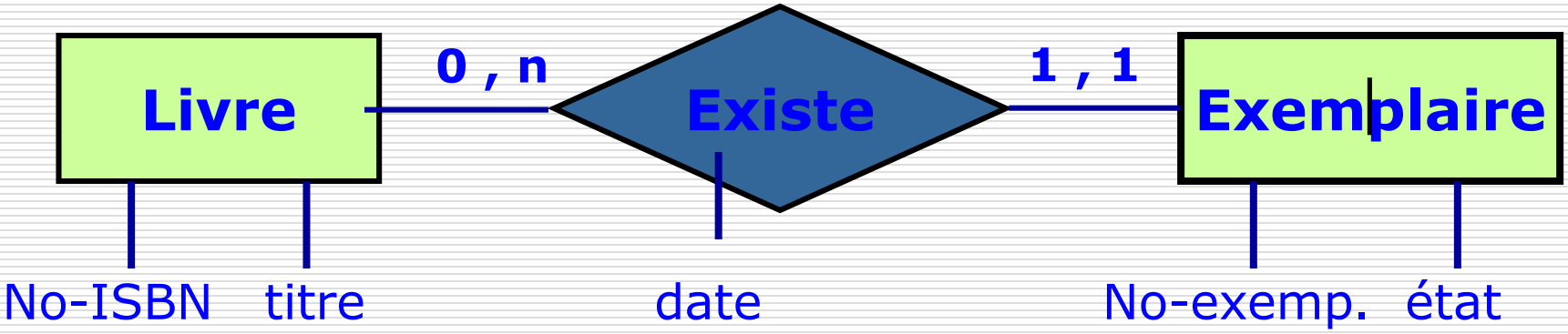
Exemple



Livre(no_isbn, titre)
Exemplaire(nb_exemplaire, état, #no_isbn)

Transformation d'un TA

Exemple



Livre(no_isbn, titre)
Exemplaire(nb_exemplaire, état, #no_isbn, date)

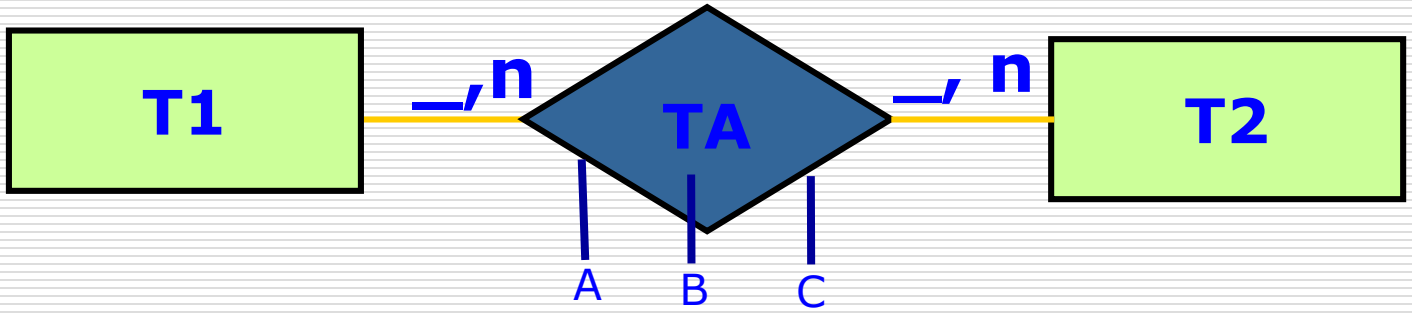
Transformation d’un TA

Association (N:N)

- Association traduite en une relation ayant pour attributs
  - les identifiants et les attributs de chaque TE participant à l'association et
  - Les attributs propres de l'association (si présents)
- la clé est formée de l’identifiant des 2 TE associés
- La clé peut également intégrer des attributs de la relation selon les besoins

Transformation d'un TA

Association binaire de plusieurs à plusieurs



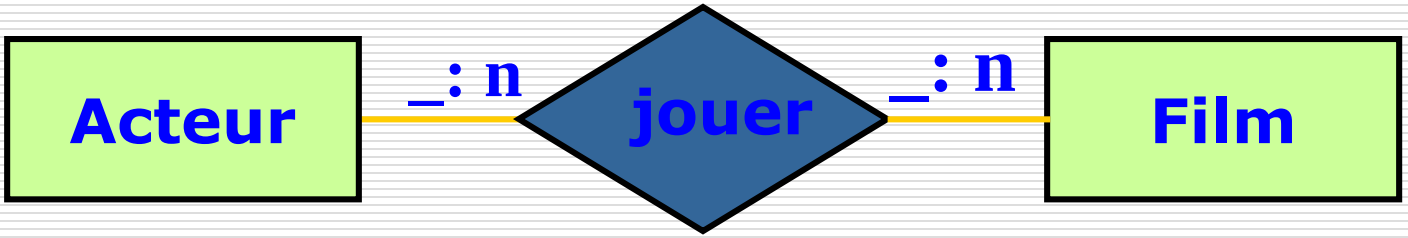
Créer une relation dont la clé représente l'identifiant de TA et dont les attributs sont ceux de TA

T1(<u>ID1</u>, …)

T2(<u>ID2</u>, …)

TA(<u>ID1, ID2</u>, A, B, C) ou TA(<u>ID1, ID2, A</u>, B, C)

Transformation d'un TA



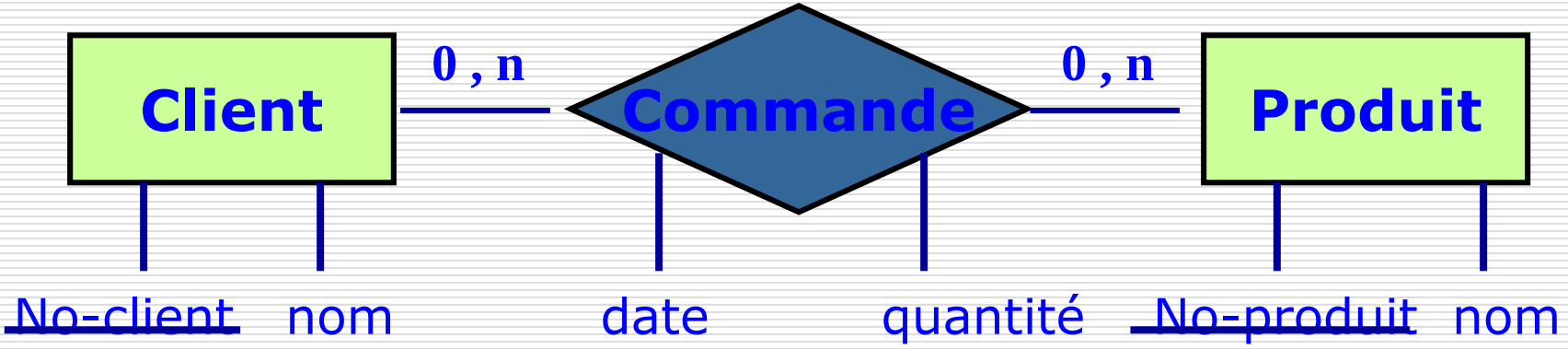
Relation à créer

Acteur(idActeur, nom, prenom, role, …)

Film(idFilm, nom, genre, …)

joeur(idActeur, idFilm)

Transformation d'un TA



Identifiant de Commande:

Client(noClient, nom, …)

Produit(noProduit, nom, …)

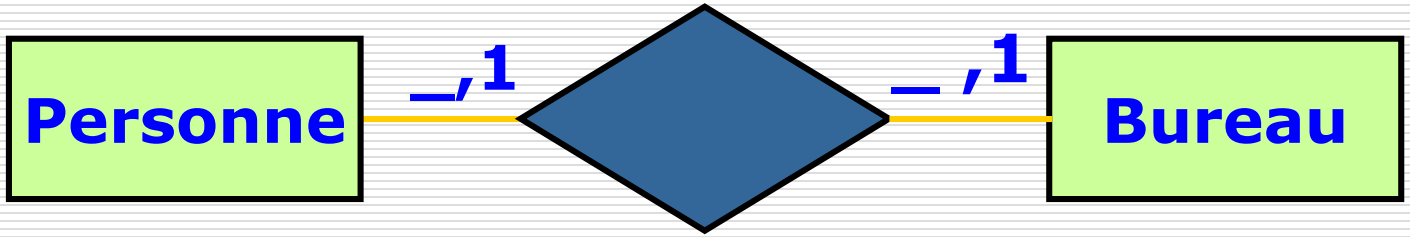
Commande(noClient, noProduit, date, quantité, …)

OU

Commande(noClient, noProduit, date, quantité, …)

Transformation d'un TA

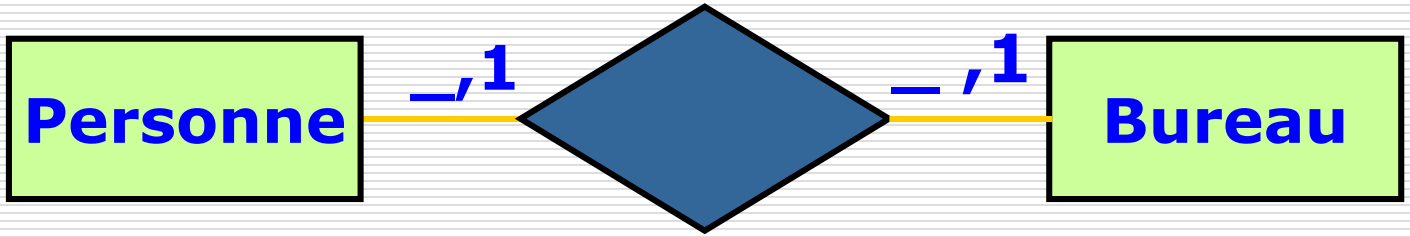
- Cas problématique



- Comment passer au modèle relationnel?

Transformation d'un TA

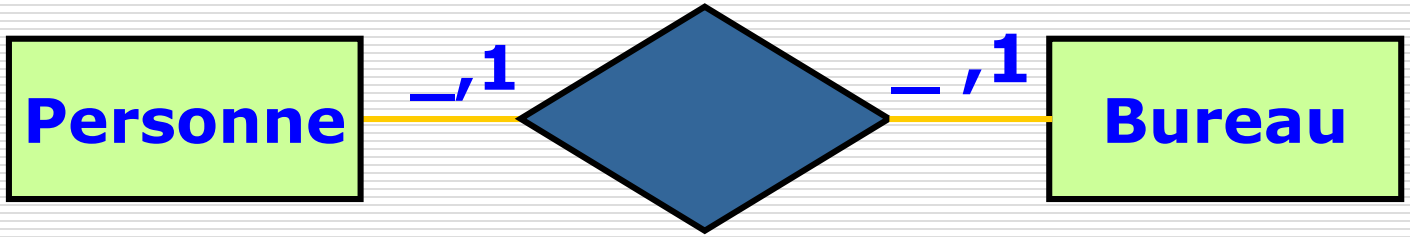
- Cas problématique



  - Personne (idPersonne, nom, prenom, …, #idBureau)
  - Bureau(idBureau, type, salle, …)

Transformation d'un TA

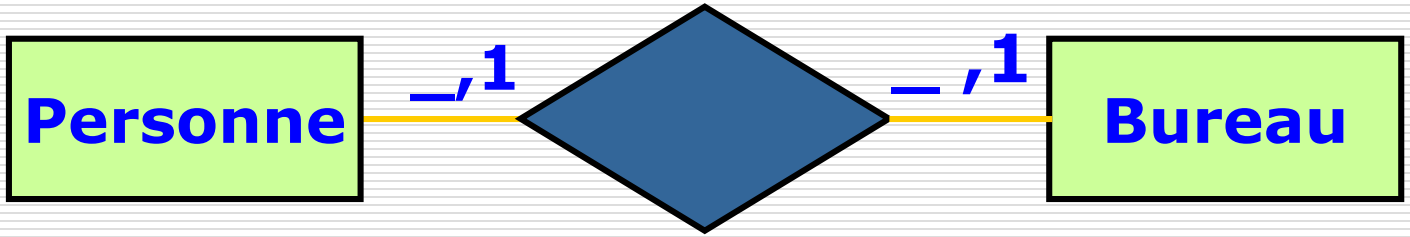
- Cas problématique



  - Personne (idPersonne, nom, prenom, …)
  - Bureau(idBureau, type, salle, …, #idPersonne)

Transformation d'un TA

- Cas problématique



  - Personne (idPersonne, nom, prenom, …, #idBureau)
  - Bureau(idBureau, type, salle, …, #idPersonne)

3. Formes normales

Concepts
1e Forme normale
2e Forme normale
3e Forme normale

3. Formes normales

Concepts

- Permet de mettre en évidence des relations "indésirables"
- Défini des critères pour la conception de "bonnes relations"
- Pour éviter les redondances et faciliter les évolutions de la base de données

3. Formes normales

Objectifs:

- Décomposer les relations du schéma relationnel
- Obtenir des relations simplifiées
- Aboutir à un schéma relationnel normalisé
- Eviter les répétitions
  - Ex.: Film(idFilm, titre, idAuteur, nomAuteur)
- Notion intuitive:
  - une « *bonne relation* » *peut être considérée comme* une fonction de la clé primaire vers les attributs restants

3. Formes normales

1e Forme normale

- Une relation est en 1FN si
  - tout attribut est atomique (non décomposable, n'a qu'une seule valeur)
  - constant dans le temps
- **Exemples:**
  - ELEVE (no_elv, nom, prenom, age, liste_notes, liste_matieres)

3. Formes normales

2e Forme normale

- **Une relation est 2FN si**
  - Elle est 1FN
  - Tout attribut n'appartenant pas à la clé ne dépend pas d'une partie de la clé, i.e. Tout attribut doit dépendre de la totalité de la clé
- **Exemples**:
  - Commande(numClient, numProduit, numFournisseur, nomProduit, prixProduit, nomFournisseur, adresseFournisseur, qte)

3. Formes normales

3e Forme normale

- Une relation est 3FN si
  - Elle est 2FN
  - Un attribut non clé ne dépend pas d'un ou plusieurs attributs ne participant pas à la clé
- **Exemple**:
  - VOITURE (<u>matricule</u>, marque, puissance, dateAchat, jour)

4 – Algèbre relationnelle

Opérateurs ensemblistes
Sélection
Projection
Jointure
Division

Algèbre relationnelle

- Théorie qui définit l’ensemble des opérations centrées sur les relations
- Donne un cadre formel pour la conception des requètes
- Utiles pour l'implémentation et l'optimisation des requêtes

Opérateurs algébriques

3 familles d'opérateurs:

- Opérateurs ensemblistes:

  UNION, INTERSECTION, DIFFERENCE, PRODUIT

  reformulés spécifiquement pour le modèle relationnel

- Opérateurs relationnels spécifiques

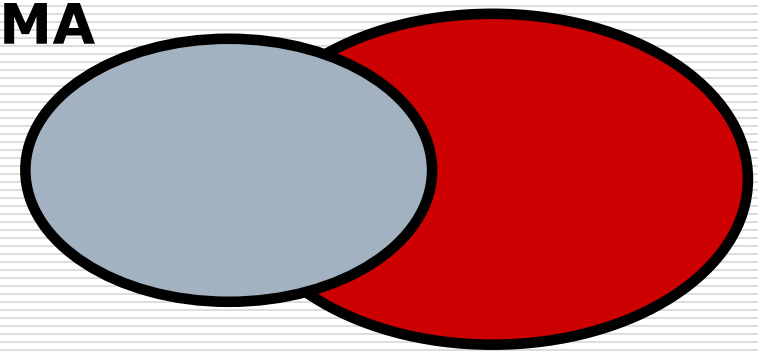
  SELECTION, PROJECTION, JOINTURE,

- Opérateurs dérivés

  JOINTURE EXTERNE, SEMI-JOINTURE

Opérations Ensemblistes

- **OPERATIONS ENSEMBLISTES POUR DES RELATIONS DE MEME SCHEMA**
  - UNION
  - INTERSECTION
  - DIFFERENCE



- **OPERATIONS ENSEMBLISTES POUR DES RELATIONS DE SCHEMAS QUELCONQUES**
  - PRODUIT CARTESIEN

- **DANS LES 2 CAS OPERATIONS BINAIRES**
  - Relation1 **op** Relation2 --> Relation3

UNION, INTERSECTION, DIFFERENCE

Etant données deux relations R1 et R2 de même schéma

R1$\cup$ **R2** est la relation contenant les tuples appartenant à R1 ou à R2

R1$\cap$**R2** est la relation contenant les tuples appartenant à R1 et à R2

R1 - R2 est la relation contenant les tuples de R1 n'appartenant pas à R2

UNION, INTERSECTION, DIFFERENCE

R1

A1	A2	A3
a1	a2	a3
b1	b2	b3
c1	c2	c3
d1	d2	d3

R2

A1	A2	A3
a1	a2	a3
e1	e2	e3
b1	b2	b3

UNION ?

R1∪R2

INTERSECTION ?

R1∩R2

DIFFERENCE ?

R1-R2

UNION, INTERSECTION, DIFFERENCE

R1

A1	A2	A3
a1	a2	a3
b1	b2	b3
c1	c2	c3
d1	d2	d3

R2

A1	A2	A3
a1	a2	a3
e1	e2	e3
b1	b2	b3

UNION

R1∪R2

A1	A2	A3
a1	a2	a3
b1	b2	b3
c1	c2	c3
d1	d2	d3
e1	e2	e3

INTERSECTION

R1∩R2

A1	A2	A3
a1	a2	a3
b1	b2	b3

DIFFERENCE

R1-R2

A1	A2	A3
c1	c2	c3
d1	d2	d3

PRODUIT CARTESIEN

Soient les relations R(A1, ..., An) et S(B1, ..., Bp)
avec {A1, ..., An} $\cap$ {B1, ...,Bp} éventuellement vide

Le *produit cartésien* de S et de R noté **R x S**
est défini par la relation Q(A1, ..., An, B1, ..., Bp)
telle que
(a1, ..., an, b1, ..., bp) $\in$Q
ssi
(a1, ..., an)$\in$R et (b1, ..., bp) $\in$S

Produit cartésien

R1

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

R2

X	Y
x1	y1
x2	y2

PRODUIT CARTESIEN ?

R1XR2

Produit cartésien

R1

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

R2

X	Y
x1	y1
x2	y2

PRODUIT CARTESIEN

R1XR2

A	B	C	X	Y
a1	b1	c1	x1	y1
a2	b2	c2	x1	y1
a3	b3	c3	x1	y1
a1	b1	c1	x2	y2
a2	b2	c2	x2	y2
a3	b3	c3	x2	y2

Propriétés

degré(R1∪R2)=?

degré(R1∩R2)=?

degré(R1−R2)=?

degré(R1XR2)=?

*Rappel: degré = nombre d'attributs de la relation

Propriétés

degré(R1∪R2)=degré(R1)=degré(R2)

degré(R1∩R2)=degré(R1)=degré(R2)

degré(R1−R2)=degré(R1)=degré(R2)

degré(R1XR2)=degré(R1)+degré(R2)

SELECTION

La sélection : opérateur SELECT utilisé pour sélectionner un sous-ensemble de tuples d'une relation qui vérifient une condition

exemple : $\sigma_{\text{adresse=PARIS}}(\text{Client})$

Client

numéro	nom	adresse	téléphone
101	Durand	NICE	0493942613
106	Fabre	PARIS	
110	Aurand	PARIS	
125	Carré	MARSEILLE	0491258472

relation résultante

La relation résultante a le même schéma que la relation sur laquelle porte la requête

PROJECTION

La projection : opérateur PROJECT utilisé pour sélectionner certaines colonnes d'une relation

exemple : $\pi_{\text{nom, téléphone}}$(Client)

Client

numéro	nom	adresse	téléphone
101	Durand	NICE	0493942613
106	Fabre	PARIS	
110	Aurand	PARIS	
125	Carré	MARSEILLE	0491258472

relation résultante

JOINTURE

- La jointure : opérateur JOIN, noté ⋈ , utilisé pour combiner une paire de tuples de deux relations en un seul tuple
- Une condition de jointure doit être introduite

Client ⋈ Vente

numéro=no_client

Client

numéro	nom	adresse	téléphone
101	Durand	NICE	0493942613
106	Fabre	PARIS	
106	Fabre	PARIS	
125	Carré	MARSEILLE	0491258472

Vente

numéro	ref_produit	**no_client**	date
00102	AF153	**101**	12/10/04
00809	BG589	**106**	18/10/04
11005	VF158	**106**	05/10/04
12005	BG589	**125**	25/10/04

Exemple

Afficher le nom des clients avec les dates de leurs achats

Exemple

Afficher le nom des clients avec les dates de leurs achats

$$\pi_{\text{Client.nom, Vente.date}}(\text{Client} \underset{\text{numéro=no\_client}}{\rhd\!\lhd} \text{Vente})$$

JOINTURE

$$\text{Relation1} \underset{\text{condition}}{\bowtie} \text{Relation2}$$

- Degré relation résultante : n1 + n2 attributs
- résulte d'une sélection des tuples du produit cartésien R1xR2 qui vérifient la condition

$$\sigma_{R1.x = R2.x}(R1 \times R2)$$

- la condition est de la forme

R1.attribut **op** R2.attribut

Exercice

Attention: toutes les conditions acceptées !

Afficher la référence des produits dont le prix est supérieur au produit de ref AF153

Exercice

Attention: toutes les conditions acceptées !

Afficher la référence des produits dont le prix est supérieur au produit de ref AF153

$$P1 = \sigma_{\text{référence=AF153}}(\text{Produit})$$

$$\text{Res} = \pi_{\text{Produit.référence}}(\text{Produit} \rtimes_{\text{Produit.prix>P1.prix}} P1)$$

Equijointure
Jointure naturelle

- Equijointure
  - la condition fait appel à l'opérateur =

- Jointure naturelle
  - équijointure dont la condition porte sur des attributs identiques (de même domaine)
  - un seul des deux attributs est conservé dans le résultat

Exemple de jointure naturelle

Afficher le nom des clients avec les dates de leurs achats

$\pi_{\text{Client.nom, Vente.date}}(\text{Client} \underset{\text{numéro=no\_client}}{\rhd\!\lhd} \text{Vente})$

numéro	nom	adresse	téléphone	numéro	ref_produit	date
101	Durand	NICE	0493942613	00102	AF153	12/10/04
106	Fabre	PARIS		00809	BG589	18/10/04
106	Fabre	PARIS		11005	VF158	05/10/04
125	Carré	MARSEILLE	0491258472	12005	BG589	25/10/04

JOINTURE NATURELLE

$$\text{Relation1} \underset{\text{condition}}{\bowtie} \text{Relation2}$$

- relation résultante : n1 + n2-1 attributs
- résulte d'une sélection des tuples du produit cartésien R1xR2 qui vérifient la condition d'égalité

Autres opérateurs

- Semi-jointure gauche, droite
- Jointure externe

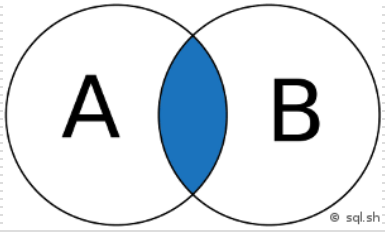
SEMI-JOINTURE

Semi-jointure gauche entre les relations S et R

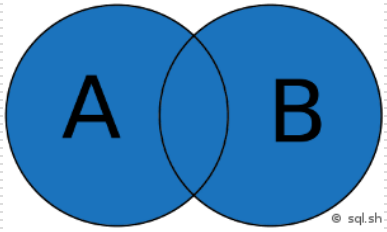
notée **S**  **R**

- contient tous les enregistrements de la table de gauche (LEFT = gauche) même si la condition n'est pas vérifié dans l'autre table.

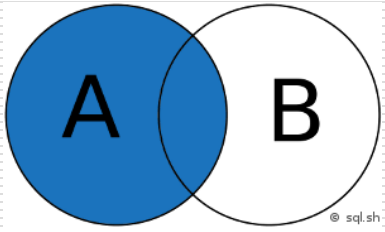
Différents types de jointure



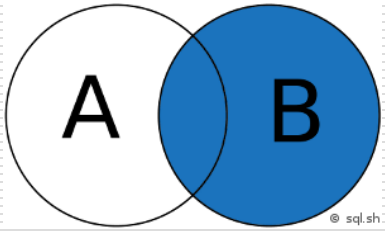
Jointure naturelle



Jointure externe



Jointure à gauche



Jointure à droite

5- Le langage algébrique

- Le langage algébrique permet de formuler une question par une suite d'opérations de l'algèbre relationnelle
- Comme le langage algorithmique
  - Indépendant du langage de programmation utilisé
  - Permet d’évaluer les performances et d’optimiser les requêtes avant leur implémentation

5- Le langage algébrique

- Décrire la suite d'opérations à réaliser pour arriver à un résultat
- Similaire à l'algorithmique
- Exemple:

  Ensemble des commandes passées en 2015, dont le montant est supérieur à 100 euros

5- Le langage algébrique

- Décrire la suite d'opérations à réaliser pour arriver à un résultat
- Exemple:

$$R1 = \sigma_{date=2015}(Commande)$$

$$R2 = \sigma_{montant>100}(Commande)$$

$$R3 = R1 \cap R2$$

Autre version ??