

Module M1104

Introduction aux bases de données

Semestre 1 – 2019/2020



➤ Enseignement sur 7 semaines

1 à 2 séances de Cours-TD par semaine

1 à 2 séances de TP par semaine

➤ Contrôles (sur papier, sans documents)

Contrôle Intermédiaire : 1 heure, coefficient 15

⇒ jeudi 5 décembre après-midi

Contrôle Final : 2 heures, coefficient 20

⇒ semaine de partiels de janvier

- Introduction : BD, SGBD et modèle relationnel
- Le langage SQL
 - Langage de définition des données
 - Langage de manipulation des données
- Modélisation d'une base de données et passage au relationnel

Introduction


BD, SGBD et modèle relationnel

Une base de données : les films à l'affiche 5

ALLOCINÉ
Rechercher un film, une série, une stat...
Ex : L'odyssée, Deepwater, La Fille inconnue, Overdrive

CINÉMA SÉRIES ÉMISSIONS NEWS TRAILERS B.O. DVD VOD

Meilleurs films Films à l'affiche Prochainement Séances Box Office Films pour enfants Courts-métrages




Donne-moi des ailes
9 octobre 2019 / 1h 53min / Aventure, Famille / Français, Norvégien
De Nicolas Vanier
Avec Jean-Paul Rouve, Mélanie Doutey, Louis Vazquez

Christian, scientifique visionnaire, étudie les oies sauvages. Pour son fils, ado obnubilé par les jeux vidéos, l'idée de passer des vacances avec son père en pleine nature est un cauchemar. Pourtant, père et fils vont se rapprocher autour d'un projet fou : sauver une espèce en voie de disparition.

PRESSE ★★★★★ 3,3 SPECTATEURS ★★★★★ 4,3 MES AMIS ★★★★★ —

En VF, Numérique

14:00 16:00 18:00 20:00 22:00



J'irai où tu iras
2 octobre 2019 / 1h 40min / Comédie / Français
De Géraldine Nakache

Appareil photo numérique (...)

www.fnac.com/Photo-camescope/shi56352/w-4#bl=MMpho

Rechercher

Les plus visités Débuter avec Firef... À la une Ville de Saint-M... Save to Mendeley

fnac
Culture, High-tech, Loisirs >

Rayons Que recherchez-vous ?

Besoin d'aide ? Magasins Me connecter Mon panier

TOUS NOS RAYONS

PHOTO, CAMÉSCOPE

TOUTES NOS OFFRES

Voir tout

Ventes Flash Offres Adhérents Les Packs Fnac

APPAREIL PHOTO REFLEX

Voir tout

Par marque

Canon : la gamme Pro

Nikon : la gamme Pro

Par prix

Par usage

APPAREIL PHOTO HYBRIDE

APPAREIL PHOTO COMPACT


BRIDGE

Experts Photos Fnac. Conservez vos plus beaux souvenirs grâce aux livres et tirages photo.
Retrouvez le meilleur de la technologie Reflex, Compact, Hybride ou Bridge. Choisissez vos accessoires parmi une large sélection : carte mémoire 32Go, Objectif pour Nikon

TOUTES NOS OFFRES

Ventes Flash Offres Adhérents Les Packs Fnac

BON PLAN -14%




Panasonic Lumix DMC-LX100EFK
Noir + Carte Mémoire Sandisk ultra

Appareil photo numérique compact - Panasonic

★★★★★

599€99 699€99

BON PLAN -22%



Drone DJI Phantom 3 4K


Autre objet connecté - Dji

★★★★★

699€99 899€99

Plus d'offres dès **530€90**

BON PLAN -32%




Hybride Olympus E-PL6 Noir +
Objectif 14-42 mm + Objectif

Appareil photo hybride - Olympus

449€98 659€98

BON PLAN -25%



Hybride Olympus E-PL6 Noir +
Objectif 14-42 mm

Appareil photo hybride - Olympus

★★★★★

299€99 399€99

Voir tout

- Donnée : information qui peut être codifiée et enregistrée
- Bases de Données : Ensemble structuré de données accessibles par l'ordinateur et partagées par plusieurs utilisateurs

Exemples :

- *Films, salles, horaires, ... dans un cinéma*
- *CD, DVD, livres, prix, stock, ... des articles de la FNAC*
- *Etudiants, enseignants, cours, ... d'une université*

➤ Schéma : **description** de la base de données

- Le schéma est spécifié lors de la conception de la base de données, il ne doit pas changer
- Il est décrit à l'aide d'un modèle de données

Ex : Un étudiant a un nom et une année de naissance

➤ Instance : **état** de la BD à un instant donné

- Une BD peut avoir des instances différentes à des moments différents
- On peut avoir plusieurs BD qui ont le même schéma, mais qui possèdent des instances différentes

Ex : Les étudiants sont Durand(1978), Dupond(1982), ...

➤ Système de Gestion de Bases de Données : Logiciel qui permet aux utilisateurs d'utiliser une base de données

c-à-d :

- **Définir** la BD = spécifier la structure (le schéma) des données
- **Construire** la BD = stocker l'information (les instances) sur un support de stockage physique
- **Manipuler** la BD = l'interroger pour retrouver des données spécifiques et la modifier pour refléter des changements du monde réel

Exemples de SGBD : MySQL, Oracle, Postgres, Access ...

- **Persistance des données** : sauvegarder les données
- **Intégrité** : pour être cohérentes, les données doivent vérifier des règles (*Contraintes d'Intégrité*)
- **Redondance faible** : chaque information n'est stockée qu'une fois. Dans le cas contraire, les informations dupliquées doivent rester valides
- **Partage des données et gestion de la concurrence** : les utilisateurs doivent pouvoir accéder en même temps aux mêmes données
- **Confidentialité** : les données sensibles doivent être protégées (autorisations, droits d'accès aux données)
- **Sécurité des données** : il faut gérer les reprises en cas de pannes logicielles ou matérielles

- **L'administrateur de la base de données**
Gère les utilisateurs, les droits d'accès, les ressources, ...
- **Le concepteur de la base de données**
Définit et crée la base de données
- **Les programmeurs d'applications**
Conçoivent les applications (ex : sites web) qui accèdent à la base de données
- **Les utilisateurs finaux**
Utilisent la base de données via une application ou un langage de requêtes ou une interface graphique

Le modèle de données relationnel

➤ Modèle de données : ensemble de concepts qui peuvent être utilisés pour décrire **la structure** d'une base de données c-à-d :

- les **données** et leurs types

Ex : le nombre d'heures d'un cours (un entier)

- les **relations** (liens) entre les données

Ex : un étudiant suit un cours

- les **contraintes** qui s'appliquent sur les données

Ex : une note est entre 0 et 20

➤ Un modèle de données offre aussi un **ensemble d'opérateurs** pour spécifier les recherches sur la base de données

- Modèle relationnel : modèle de données basé sur des notions mathématiques simples (ensemble, produit cartésien, relation, ...) et défini par Codd en 1970
 - L'information est décrite dans des tableaux à 2 dimensions appelés tables ou **relations** qui permettent de représenter à la fois les données et la façon dont ces données sont reliées entre elles
 - Les **opérateurs** permettent d'interroger la base de données. Ils produisent des relations résultats à partir des relations existantes
 - Les **contraintes d'intégrité** sont prises en compte

- Domaine : ensemble fini ou infini de valeurs atomiques

Ex : entier

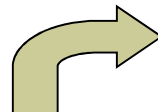

- Schéma d'une relation : ensemble d'**attributs** $\{A_1, A_2, \dots, A_p\}$
où chaque attribut A_j est un nom associé à un domaine D_j .

*Ex : Etudiant (Numetu : entier, Nom : chaîne, AnNaiss : entier)
abrégé en Etudiant (Numetu, Nom, AnNaiss)*

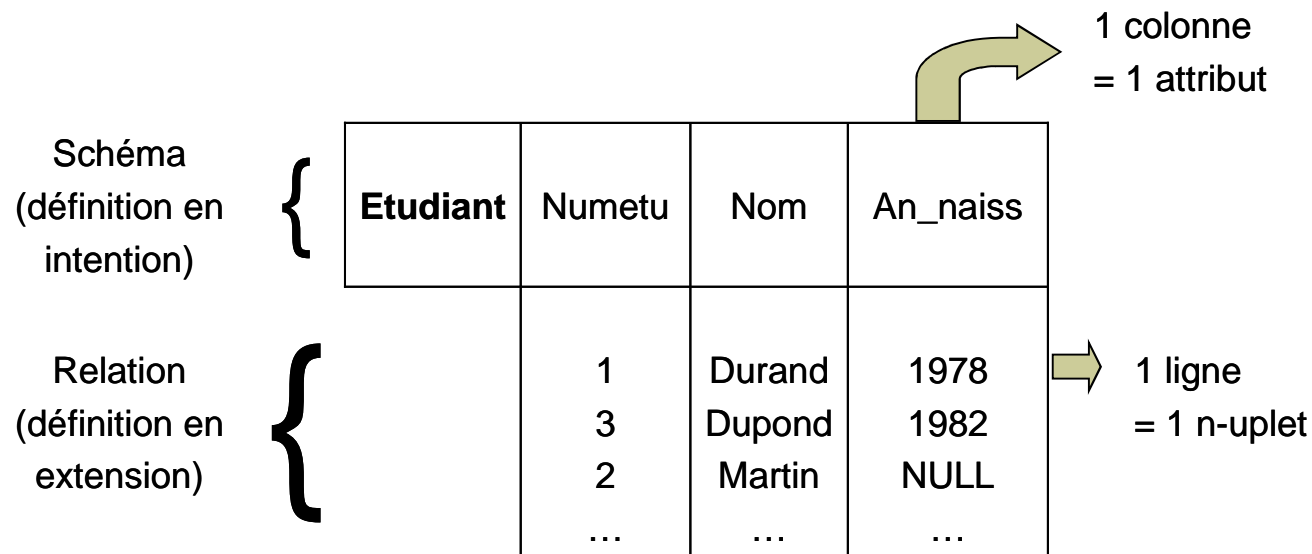
- Relation (ou instance) : ensemble des **n-uplets** $(val_{i1}, val_{i2}, \dots, val_{ip})$
où la valeur val_{ij} d'un attribut A_j est prise dans le domaine D_j .

Ex : $\{ (1, Durand, 1978), (2, Dupond, 1982), (3, Martin, ?), \dots \}$

Exemple : la relation *Etudiant* (Numetu, Nom, AnNaiss)

Schéma (définition en intention)	{	Etudiant	Numetu	Nom	An_naiss	 1 colonne = 1 attribut	
Relation (définition en extension)			1	Durand	1978		 1 ligne = 1 n-uplet
			3	Dupond	1982		
			2	Martin	NULL		
				

- Toutes les valeurs d'attributs sont **atomiques**
- **Il n'y a pas de n-uplet dupliqué**
- Les n-uplets ne sont pas ordonnés
- Les attributs ne sont pas ordonnés



Un exemple de BD, avec des données

18

Etudiant	NumEtu	Nom	An_Naiss
	1	Dupont	1998
	2	Martin	1999
	3	Durand	1996
	4	Durand	1998

COURS	Code	Intitulé	VolHoraire
	M1104	BD1	55
	M1105	Web	70
	M2106	BD2	20
	M3106	BD3	20

RESULTAT	NumEtu	CodeCours	Notef
	1	M1104	12,5
	1	M1105	16
	1	M2106	8
	2	M1104	7
	2	M1105	10
	3	M1105	18

PROF	NumProf	Nom	Specialité
	1	Jean	BD
	2	Jeannine	Maths

ENSEIGNE	NumProf	CodeCours
	1	M1104
	1	M2106
	2	M1104
	3	M1105

- Clé candidate : un ou plusieurs attributs (le moins possible) permettant d'identifier de manière unique un n-uplet parmi les autres
- Plus formellement : une clé candidate K d'une relation R est un sous-ensemble des attributs qui possède les propriétés suivantes :
 - Propriété d'**unicité** : 2 n-uplets distincts de R ont des valeurs distinctes pour K
 - Propriété d'**irréductibilité** : Aucun sous-ensemble propre (plus petit) de K ne possède la propriété d'unicité
- Il est possible qu'une relation ait plusieurs clés candidates

Ex : Cours (Code, Intitulé, VolHoraire)

Code et Intitulé sont deux clés candidates pour Cours

(en supposant que 2 cours différents n'ont pas le même intitulé)

- Lorsqu'une relation a plusieurs clés candidates, le modèle relationnel impose que l'une des clés candidates soit choisie comme clé primaire (ou simplement "clé")
- Il n'y a pas de méthode pour le choix
- Notation : la clé primaire est soulignée

Ex : Etudiant (Numetu, Nom, AnNaiss)

Numetu est clé primaire (ici unique clé candidate)

Cours (Code, Intitulé, VolHoraire)

Code est choisie comme clé primaire

Intitulé est une clé candidate alternative

Resultat (Numetu, CodeCours, NoteF)

Une seule clé primaire double (Numetu, CodeCours)

- Clé étrangère (ou contrainte d'intégrité référentielle) : un ou plusieurs attributs qui correspondent à un ou plusieurs attributs d'une autre relation
- Plus formellement : une clé étrangère K2 d'une relation R2 est un sous-ensemble des attributs de R2 tel que :
 - Il existe une relation R1 ayant K1 comme clé candidate
 - A tout moment, chaque valeur de K2 dans R2 est identique à la valeur de K1 d'un n-uplet de R1

En d'autres termes, pour exister dans R2, toute valeur de K2 doit déjà être présente dans R1 comme valeur de K1
- Notation : une clé étrangère est précédée du signe #

Etudiant (Numetu, Nom, AnNaiss)

Cours (Code, Intitulé, VolHoraire)

Resultat (#Numetu, #CodeCours, NoteF)

Pour qu'un étudiant ait un résultat à un cours, il faut déjà que cet étudiant soit un étudiant (dans Etudiant) et que ce cours soit un cours (dans Cours)

⇒ 2 clés étrangères :

Numetu de Resultat	référence Numetu de Etudiant
CodeCours de Resultat	référence Code de Cours

- Les contraintes d'intégrité référentielles impliquent un ordre de création et de destructions des relations et des n-uplets



A vous

23

Etudiant (Numetu, Nom, AnNaiss)
Cours (Code, Intitulé, VolHoraire)
Resultat (#Numetu, #CodeCours, NoteF)

Quelles sont les clefs primaires et étrangères des relations Prof et Enseigne :

Prof (NumProf, Nom, Spécialité)
Enseigne (NumProf, CodeCours)

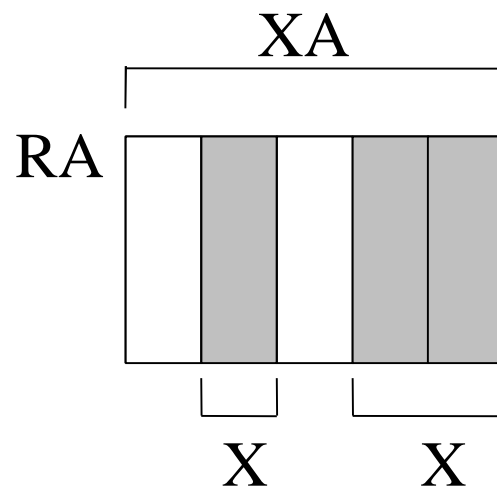
Le langage algébrique fournit un ensemble **d'opérateurs** permettant **d'interroger** une BD

- Opérateurs unaires : $1 \text{ relation} \rightarrow 1 \text{ relation}$
 - **Sélection, Projection**
- Opérateurs binaires : $2 \text{ relations} \rightarrow 1 \text{ relation}$
 - **Union, Intersection, Différence** (Opérateurs binaires ensemblistes) : le schéma des 2 relations doit être le même
 - **Produit Cartésien, Jointure, Division** : le schéma des 2 relations n'est pas le même (en général)
- Comme chaque opération retourne une relation, les opérations peuvent être composées

Soit RA une relation définie sur le schéma XA.

➤ Projection :

- Le résultat R est défini sur un **schéma** $X \subset XA$
- Il contient les "**sous n-uplets**" contenant uniquement les attributs correspondant à X



$$R \leftarrow RA[X]$$

Le résultat est la partie grisée



À vous...

26

Soit le schéma de BD :

Etudiant (NumEtu, Nom, AnNaiss)

Cours(Code, Intitulé, VolHoraire)

Resultat(#NumEtu, #CodeCours, NoteF)

Donner en langage algébrique :

➤ *les noms de tous les étudiants,*

➤ Sélection :

- Le résultat R est défini sur le **même schéma** XA.
- Il contient les n-uplets satisfaisant une **condition**
- La condition porte sur les attributs de RA, les valeurs des domaines et s'exprime à l'aide de :

$=, \neq, <, >, \leq, \geq, \wedge, \vee, \neg$

RA

R ← RA / Condition

Le résultat est la partie grisée



Etudiant (NumEtu, Nom, AnNaiss)
Cours(Code, Intitulé, VolHoraire)
Resultat(#NumEtu, #CodeCours, NoteF)

28

Donner en langage algébrique :

➤ *Les cours dont le volume horaire est inférieur à 30h*

➤ *L'année de naissance des étudiants de nom Durand*

Soient RA et RB 2 relations définies **sur le même schéma X**

Le résultat R est défini sur ce schéma X.

➤ Union : **$R \leftarrow RA \cup RB$**

- R contient les n-uplets appartenant à RA **ou** RB (ou les 2)

➤ Intersection : **$R \leftarrow RA \cap RB$**

- R contient les n-uplets appartenant à RA **et** RB

➤ Différence : **$R \leftarrow RA - RB$**

- R contient les n-uplets appartenant à RA **mais pas** à RB



Étudiant (NumEtu, Nom, AnNaiss)
Cours(Code, Intitulé, VolHoraire)
Resultat(#NumEtu, #CodeCours, NoteF)

30

Donner en langage algébrique :

- *les numéros des étudiants qui suivent le cours M1104 ou le cours M2106*

- *les numéros des étudiants qui suivent le cours M1104 et le cours M2106*

Soit RA et RB 2 relations définies sur les schémas XA et XB.

➤ Produit cartésien :

- Les schémas XA et XB doivent être disjoints.
- Le résultat R est défini sur le schéma $XA \cup XB$
- Il contient les n-uplets qui sont la concaténation des n-uplets de RA avec ceux de RB

RA	U
	u1
	u2
	u3

RB	V
	v1
	v2

R	U	V
	u1	v1
	u1	v2
	u2	v1
	u2	v2
	u3	v1
	u3	v2

$$R \leftarrow RA \times RB$$

➤ Jointure :

- XA et XB doivent avoir une **intersection non vide**
- Le résultat R est défini sur le schéma **$\mathbf{XA} \cup \mathbf{XB}$**
- Il contient les n-uplets qui sont la concaténation des n-uplets de RA avec ceux de RB s'ils ont la même valeur pour le ou les attributs en commun

RA	U	VA
	u1	v1
	u2	v1
	u3	v3
	u4	v4

RB	VB	W
	v1	w1
	v2	w2
	v3	w3
	v3	w4

R	U	?	W
	u1	v1	w1
	u2	v1	w1
	u3	v3	w3
	u3	v3	w4

$$\mathbf{R} \leftarrow \mathbf{RA} * (\mathbf{VA} = \mathbf{VB}) \mathbf{RB}$$



Etudiant (NumEtu, Nom, AnNaiss)
Cours(Code, Intitulé, VolHoraire)
Resultat(#NumEtu, #CodeCours, NoteF)

33

Donner en langage algébrique :

- *Les noms des étudiants ayant eu une note finale supérieure ou égale à 10 dans un cours (2 versions)*

➤ Division :

- Le schéma **XB** doit être inclus dans **XA**
- Le résultat est défini sur le schéma **XA – XB**
- Il contient les n-uplets dont la concaténation avec **tous les n-uplets de RB** appartient à RA

RA	U	V
	a	x
	a	y
	a	z
	b	x
	b	z
	c	x

RB	V
	x
	y

R	U
	a

$$R \leftarrow RA \div RB$$



Etudiant (NumEtu, Nom, AnNaiss)
Cours(Code, Intitulé, VolHoraire)
Resultat(#NumEtu, #CodeCours, NoteF)

35

Donner en langage algébrique :

➤ *les numéros des étudiants qui suivent **tous les** cours.*

- Sélection $R1 / C$
Un sous-ensemble de lignes d'une relation R1 vérifiant la condition C
- Projection $R1 [VA, \dots]$
Un sous-ensemble de colonnes (VA, ...) d'une relation R1
- Jointure $R1 * (VA = VB) R2$
Combinaison de deux relations R1 et R2 dont la valeur d'un attribut (ou des attributs VA et VB) est la même
- Différence $R1 - R2$
Ensemble des n-uplets de la relation R1 qui ne sont pas dans la relation R2
- Union $R1 \cup R2$
Ensemble des n-uplets de la relation R1 ou de la relation R2
- Intersection $R1 \cap R2$
Ensemble des n-uplets de la relation R1 qui sont aussi dans la relation R2
- Division $R1 \div R2$
Ensemble des n-uplets dont la concaténation avec tous les n-uplets de la relation R2 appartient à la relation R1

Le langage SQL

- Le langage relationnel standard, norme SQL2011
- Un Langage de Définition des Données (LDD) :

{	CREATE	sur	{	TABLE
	ALTER			VIEW
	DROP			INDEX...
- Un Langage de Manipulation des Données (LMD) :
SELECT (consultation)
INSERT, UPDATE, DELETE (mises à jour)
- Un Langage de Contrôle des Données (LCD) :
GRANT, REVOKE (droits des utilisateurs)
- Et la gestion des transactions, des connexions, ...

SELECT
pour la consultation (interrogation)
des données

➤ Forme simple

```
SELECT liste_attributs  
FROM    liste_relations  
WHERE    condition (ou expression logique);
```

➤ Forme générale complète

```
SELECT liste_attributs  
FROM liste_relations  
[ WHERE condition ]  
[ GROUP BY liste_attributs ]  
[ HAVING condition ]  
[ ORDER BY liste_attributs [ DESC ] ] ;
```

- Les crochets indiquent des clauses optionnelles
- Les résultats peuvent contenir des n-uplets identiques !

- Clause **FROM** : relations, définition d'alias, ...
- Clause **SELECT** : attributs, **DISTINCT**, *****, ...
- Clause **WHERE** : condition sur un n-uplet définie avec :
 - NOT, AND, OR**
 - =, < > (ou !=), <, <=, >, >=**
 - BETWEEN, IN, LIKE, EXISTS**
 - IS NULL, IS NOT NULL**
- Expression numérique : **+, -, *, /**
- Clause **ORDER BY** : Tri

➤ Sélection : R / condition

```
SELECT  *  
FROM    R  
WHERE    condition ;
```

➤ Projection : $R [B1, B2, \dots, Bp]$

```
SELECT  distinct B1, B2, ..., Bp  
FROM    R ;
```



A vous... (cf. feuille « Base Etudiant »)

43

Soit le schéma

Etudiant (Numetu, Nom, AnNaiss)

Prof (Numprof, Nom, Spécialité)

Cours (Code, Intitulé, VolHoraire)

Enseigne (#NumProf, #CodeCours)

Resultat (#NumEtu, #CodeCours, NoteF)

➤ Requêtes R1, R2

Traduction des opérateurs de l'algèbre

➤ **Produit cartésien : $RA \times RB$**

SELECT *

FROM RA, RB;

➤ **Jointure : $RA * (VA = VB) RB$**

SELECT *

FROM RA, RB

WHERE RA.VA = RB.VB;



Requêtes R3, R4

➤ Opérations ensemblistes (Union, Intersection, Différence)

SELECT ... FROM ... WHERE ...

OP

SELECT ... FROM ... WHERE ...;

où **OP** \in {**UNION, INTERSECT, EXCEPT**}



Requêtes R5, R6

où **Op** $\in \{=, <, >, !=, <=, >=, \text{IN}, \text{NOT IN}, \text{EXISTS}\}$

- Si **Op** vaut "=" ou "IN", une requête imbriquée traduit souvent une jointure.

Exemple : les étudiants qui suivent le cours de code M1104

Solution 1 (jointure)

```
SELECT E.*  
FROM Etudiant E, Resultat R  
WHERE CodeCours = 'M1104'  
AND E.NumEtu = R.NumEtu;
```



Solution 2 (jointure avec IN)

```
SELECT *  
FROM Etudiant  
WHERE NumEtu IN(  
    SELECT NumEtu  
    FROM Resultat  
    WHERE CodeCours = 'M1104');
```

Requêtes R3, R7

- **Sous-requête corrélatrice** : Sous-requête qui s'exécute pour chaque ligne de la requête principale

```
SELECT ...  
FROM      ... , RA  
WHERE     ... Op      ( SELECT      ...  
                        FROM      RB  
                        WHERE  RB... = RA. ... ) ;
```


Opérations et fonctions d'agrégation

➤ **Opérations** **Op** $\in \{+, -, *, /\}$

SELECT ... **C_i** **Op** **C_j** ...

FROM ...

WHERE ... **C_x** **Op** **C_y** ;

➤ **Fonctions d'agrégations**

f $\in \{\text{COUNT, SUM, MAX, MIN, AVG}\}$

SELECT ... **f**(**C_i**)...

FROM ...

WHERE ... ; (*pas possible dans le WHERE*)



Requêtes R8 à R12

- **Group By** : regroupe plusieurs lignes du résultat d'un Select en fonction de la valeur d'un ou des attributs spécifiés et renvoie une seule ligne par groupe

Exemple : Nombre de cours suivis par chaque étudiant

```
SELECT    NumEtu, COUNT ( CodeCours )  
FROM      Resultat  
GROUP BY NumEtu ;
```

- Les attributs du Select sont les mêmes que ceux du Group By + éventuellement des fonctions d'agrégation

- **Having** : condition qui porte sur chaque sous-ensemble (exprimée avec des fonctions d'agrégation)

Exemple : N° des étudiants qui suivent au moins 2 cours

```
SELECT      NumEtu
FROM        Resultat
GROUP BY    NumEtu
HAVING      COUNT ( * )  >=  2  ;
```

Exemple d'exécution d'une requête avec Group By et Having (cf fin du poly)

- Condition dans **WHERE** = sur un n-uplet
⇒ sans fonction d'agrégation
- ≠ Condition dans **HAVING** = sur un sous-ensemble
⇒ avec fonction d'agrégation
- En général, il y a une clause **GROUP BY** avant une clause **HAVING** mais ce n'est pas obligatoire



Requêtes R13 à R16

CREATE / DROP / ALTER
pour la création du schéma
de la BD

```
CREATE TABLE nom_relation  
  ( { définition_attribut  
    | contrainte_de_relation } [ , ... ]  
  )
```

où définition attribut :

nom_att nom_type [DEFAULT *val*] [*contrainte_attribut*] [...]

[] : optionnel

{ ... | ... } : choix

[, ...] ou [...] : répétition de l'élément précédent

➤ Numériques :

`smallint, integer, numeric(n, d), real, ...`

➤ Caractères :

`char, char(n), varchar(n), text`

➤ Dates :

`date, time, timestamp, interval`

➤ Booléen :

`boolean (FALSE, TRUE)`

CONSTRAINT **nom_cont** : nomme une contrainte

➤ Contraintes d'attributs :

- **primary key,**
- **unique,**
- **not null**
- **check (condition),**
- **references nom_relation(attribut)**

➤ Contraintes de relation :

- **primary key (liste_attributs),**
- **unique (liste_attributs)**
- **check (condition)**
- **foreign key (liste_attributs)**
references nom_relation (liste_attributs)



A vous...

57

Soit le schéma

Etudiant (NumEtu, Nom, AnNaiss)

Prof (NumProf, Nom, specialite)

Cours (Code, Intitule, VolHoraire)

Enseigne (# NumProf, #CodeCours)

Resultat (# NumEtu, #CodeCours, NoteF)

Créer les relations Etudiant, Prof, Cours, Enseigne, Resultat

➤ Suppression de relations

DROP TABLE liste_relations ;

➤ Modification du schéma d'une relation

ALTER TABLE nom

- **ADD COLUMN** def_att
- **DROP COLUMN** nom_att
- **ADD [CONSTRAINT** nom_cont] def_cont
- **DROP CONSTRAINT** nom_cont
- **ALTER** nom_att **TYPE** nom_type



À vous...

59

- Supprimer les tables Etudiant et Résultat
- Modifier le type de l'intitulé d'un cours (pour qu'il soit maintenant de type **text**)

INSERT / UPDATE / DELETE
pour la mise à jour des données
de la BD

➤ Ajout de n-uplets

```
INSERT INTO nom_rel [(liste_att)]  
VALUES(liste_valeurs) ;
```

```
INSERT INTO nom_rel[(liste_att)]  
SELECT... ;
```

➤ Modification de n-uplets

```
UPDATE nom_rel  
SET liste_valeurs  
[WHERE condition ] ;
```

➤ Suppression de n-uplets

```
DELETE FROM nom_rel  
[WHERE condition] ;
```



À vous...

62

- Insérer l'étudiant de numéro 57, de nom « Sam » né en 1998
- Inscrire tous les étudiants au cours M1104
- Modifier la note de l'étudiant de numéro 56 au cours M1104
- Augmenter de 1 la note de tous les étudiants au cours M1104
- Supprimer l'étudiant de numéro 28 et tous les résultats qu'il a obtenus à des cours

Relations résultats et vues

- Une **relation résultat** est une nouvelle relation créée comme résultat d'un select à partir d'autres relations.
- Elle peut être permanente ou temporaire (jusqu'à la fin de la session)

```
CREATE [TEMP] TABLE nom [(liste-attributs) ]  
AS SELECT ... FROM ... ;
```

- Les commandes **ALTER TABLE** et **DROP TABLE** existent pour les relations résultats

Exemple : Etudiants nés en 2000 puis nombre de ces étudiants

```
CREATE TABLE Etu2000 AS  
  
    SELECT    NumEtu  
  
    FROM      Etudiant  
  
    WHERE     an_naiss = 2000;  
  
SELECT    count(*) FROM Etu2000;
```

- Il y a création de nouveaux n-uplets
- On peut faire des modifications de ces n-uplets (**INSERT/UPDATE/DELETE**)
- La nouvelle relation est indépendante des relations qui ont permis de la créer

- Une **vue (view)** se crée aussi comme résultat d'un select à partir d'autres relations.
- Elle se manipule comme une relation. Elle ressemble à une relation.
- Mais c'est une relation **virtuelle** :
 - Seule la définition de la vue est stockée. Il n'y a pas création de nouveaux n-uplets
 - Son contenu est dynamique (calculé à l'exécution)

- Création et suppression de vue :

```
CREATE VIEW nomVue [(liste-attributs) ]  
AS SELECT ... FROM ... ;
```

```
DROP VIEW nomVue;
```

- Le **ALTER VIEW** n'existe pas.

Exemple : Etudiants nés en 2000 puis nombre de ces étudiants

```
CREATE VIEW V2000 AS  
SELECT    NumEtu  
FROM      Etudiant  
WHERE     an_naiss=2000;  
SELECT    count(*) FROM V2000;
```

La table résultat `Etu2000` et la vue `V2000` ont été créées puis on ajoute un nouvel étudiant né en 2000 dans la relation `Etudiant` (par un `INSERT`).

```
INSERT INTO Etudiant VALUES (100, 'Léo', 2000);
```

```
SELECT * FROM Etu2000 ;
```

-- le nouvel étudiant n'est pas retourné

```
SELECT * FROM V2000 ;
```

-- le nouvel étudiant apparaît

- On peut faire des **SELECT** et des manipulations (**INSERT/UPDATE/DELETE**) dans les relations résultats
- Par défaut, dans une vue, seul le **SELECT** est autorisé
- Les manipulations sont possibles dans certains SGBD uniquement sur les vues « simples » (avec répercussions dans les tables d'origine)

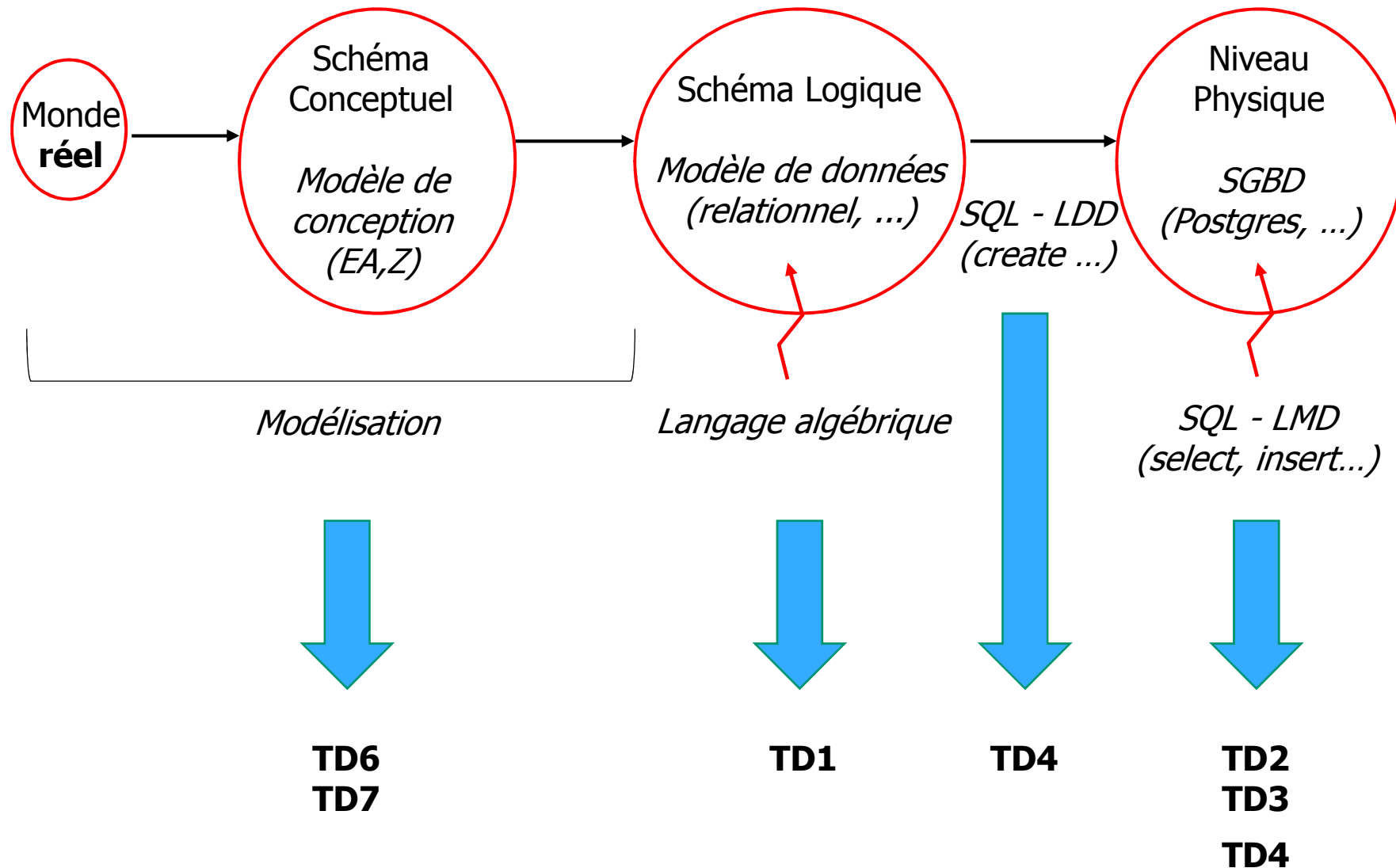
- Dans Postgres, une vue "simple" est modifiable si :
 - Une seule entrée dans le from
 - Pas de **DISTINCT**, **GROUP BY**, **HAVING**, **LIMIT**, **OFFSET**, **UNION**, **INTERSECT**, **EXCEPT** au plus haut niveau
 - Pas de fonction d'agrégation

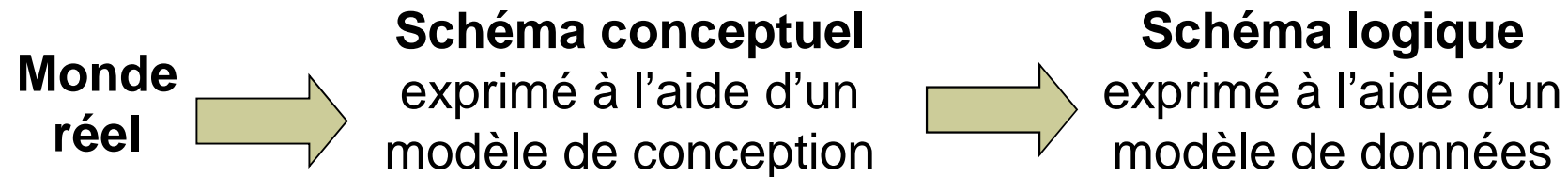
La table résultat `Etu2000` et la vue `V2000` ont été créées puis on exécute les commandes suivantes :

```
INSERT INTO Etu2000 VALUES (101, 'Tim', 2000) ;  
-- Tim n'est pas inséré dans Etudiant
```

```
INSERT INTO V2000 VALUES (102, 'Tom', 2000) ;  
-- Possible car v2000 est une vue simple  
-- Tom est inséré dans Etudiant
```

Modélisation d'une base de données





- La **modélisation** de la BD est réalisée par le **concepteur**, pour / avec l'utilisateur final, à partir des **Règles de Gestion** de l'entreprise

- Modèle de conception : outil formel pour représenter le monde réel, souvent sous forme graphique

Exemple : modèle Entité/Association

- Des règles de passage permettront de passer du schéma conceptuel (MCD) au schéma logique (relationnel)

- Les **règles de gestion** traduisent les objectifs et les contraintes du système d'information (S.I.)
- Elles décrivent **le quoi** et non **le qui fait quoi** ou **le comment**

Exemples :

- (Bowling) Un client est identifié par son nom et son prénom et est qualifié par son téléphone et son âge
- (Scolarité IUT) Un étudiant doit avoir une note pour tous les modules auxquels il assiste

- Le modèle entité/association permet de représenter le monde réel en identifiant :
 - Ensemble d'entités : ensembles d'objets de même nature
 - Associations : liens entre les ensembles
 - Contraintes sur le données

- Le MCD (modèle conceptuel de données) est une représentation graphique codifiée des données
 - Rectangles
 - Ovaes avec pattes et cardinalités

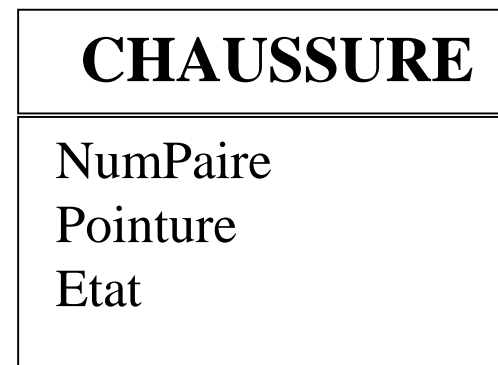
- Une **propriété** est la plus petite information que l'on souhaite manipuler

Exemple : le nom d'un client, la pointure d'une paire de chaussures

- Une propriété a un nom et un domaine
- Une propriété est atomique (elle est non décomposable)

- Une **entité** représente une famille d'objets du monde réel
- Une entité est définie par un regroupement de propriétés

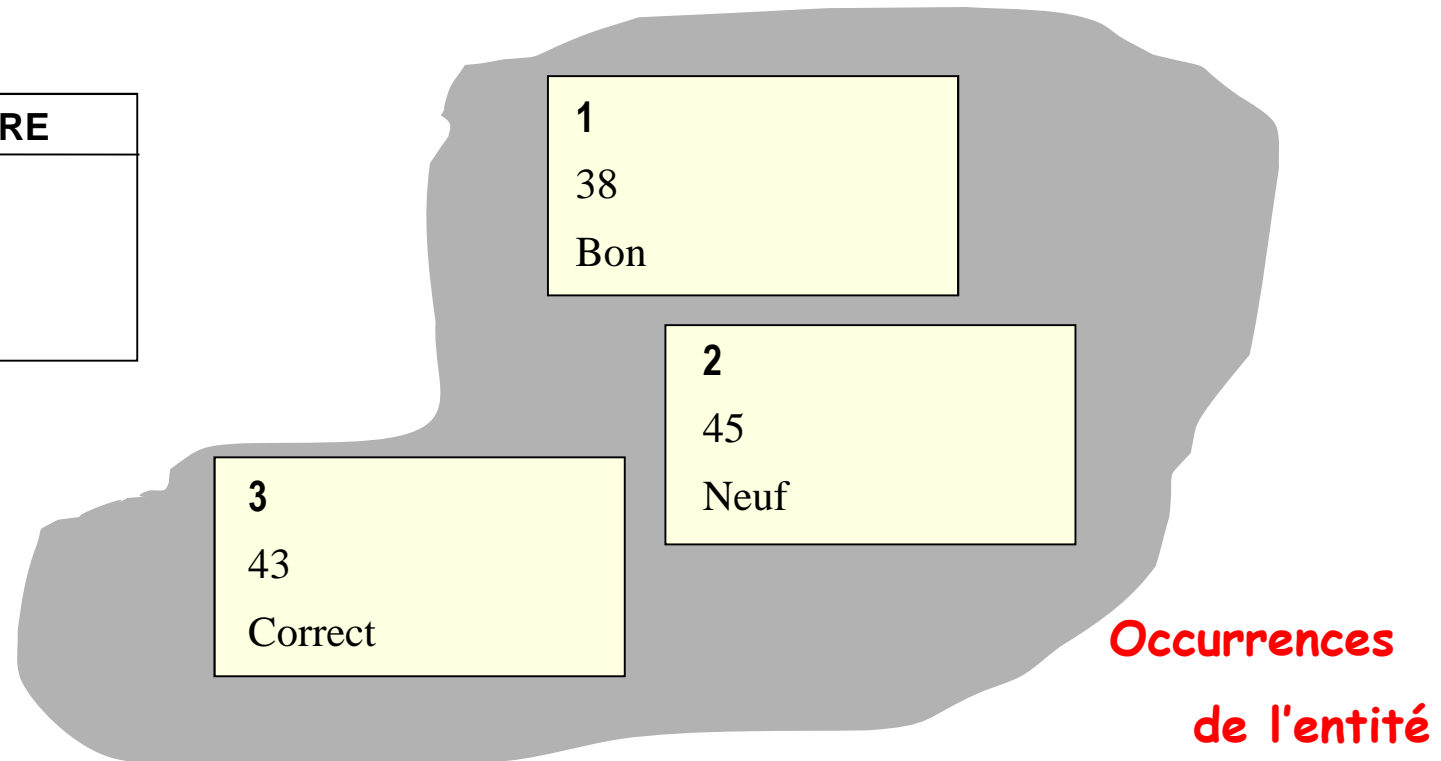
MCD : une entité est représentée par un rectangle



- Une occurrence d'entité est un objet particulier

Entité

CHAUSSURE
NumPaire
Pointure
Etat



- Un **identifiant** d'une entité est une ou plusieurs propriétés permettant de différencier chacune des occurrences d'une entité

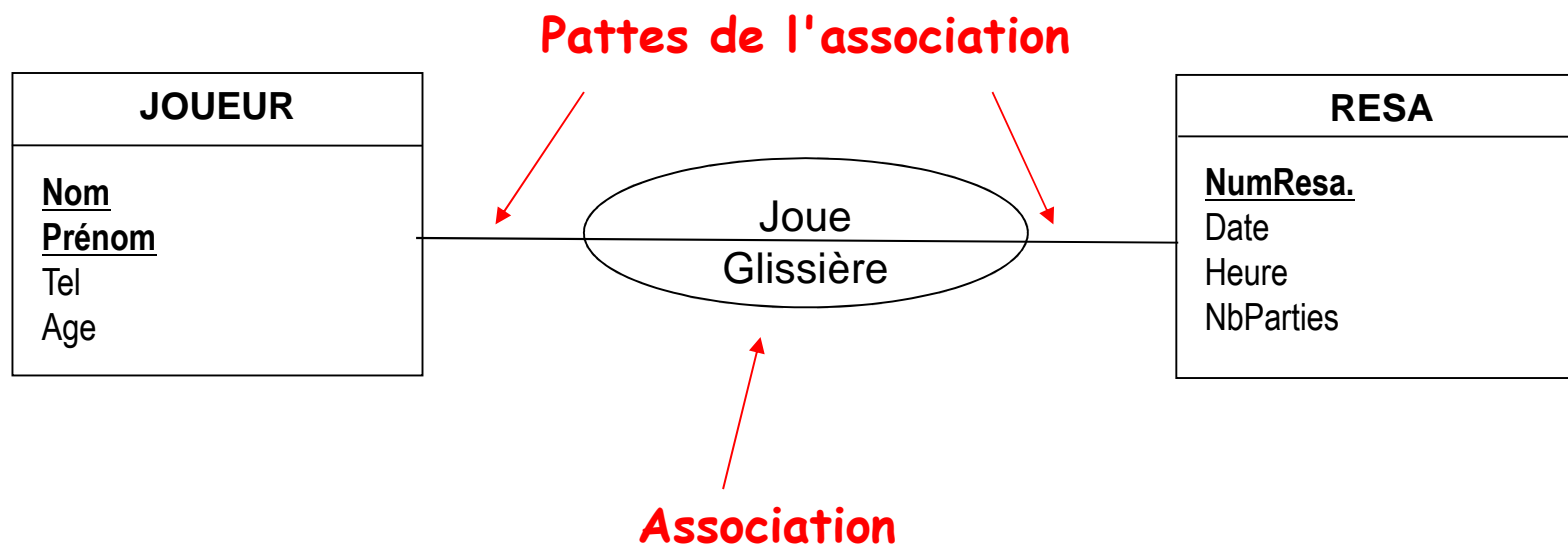
MCD : un identifiant est souligné

CHAUSSURE
<u>NumPaire</u>
Pointure
Etat

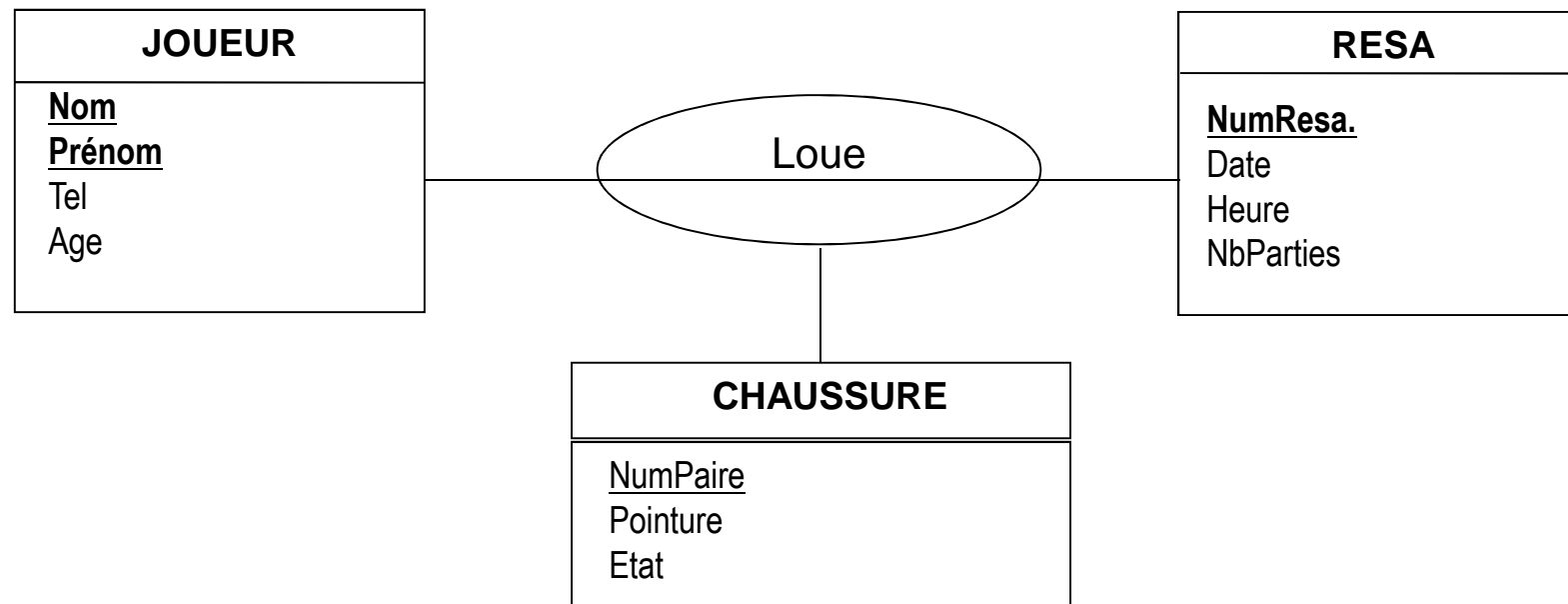
- Lorsque l'identifiant est constitué de plusieurs propriétés, toutes ces propriétés doivent être nécessaires à la différenciation des occurrences de l'entité

- Une **association** relie 2 ou plusieurs entités
- Une association est définie par :
 - ses propriétés propres (il n'y en a pas toujours)
 - les entités associées

MCD : une association est représentée par un ovale



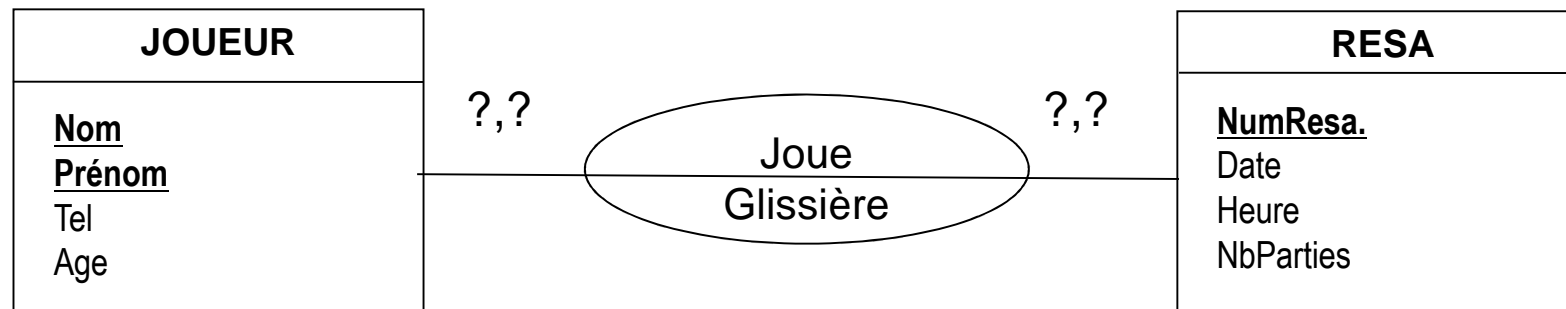
- Une **association** peut-être binaire, ternaire, ...



- Une association peut-être réflexive (relie une entité avec elle-même)
- L'identifiant d'une association est formée des identifiants des entités qu'elle relie

- Mesure la participation des occurrences d'une entité à l'association \Rightarrow couple de 2 valeurs qui sont les nombres minimum et maximum d'occurrences de l'association pouvant exister pour **une** occurrence de l'entité

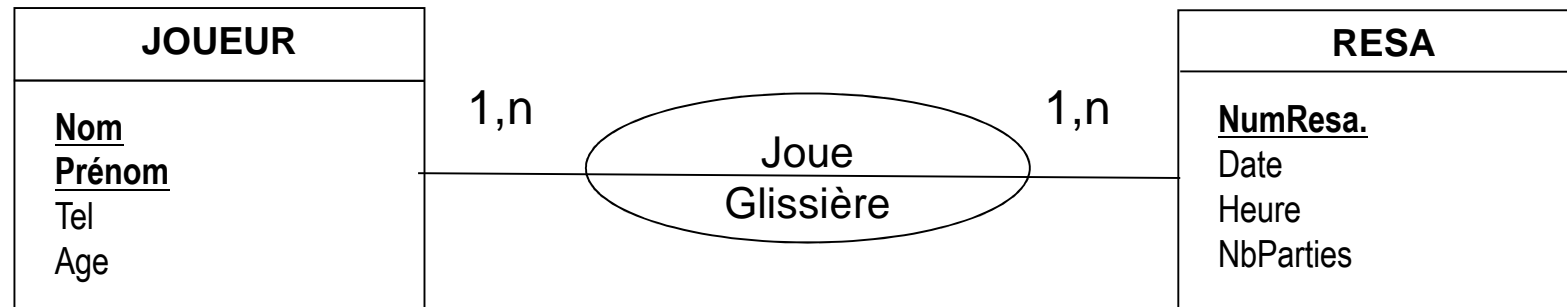
MCD : le couple de cardinalités se rajoute sur chaque patte



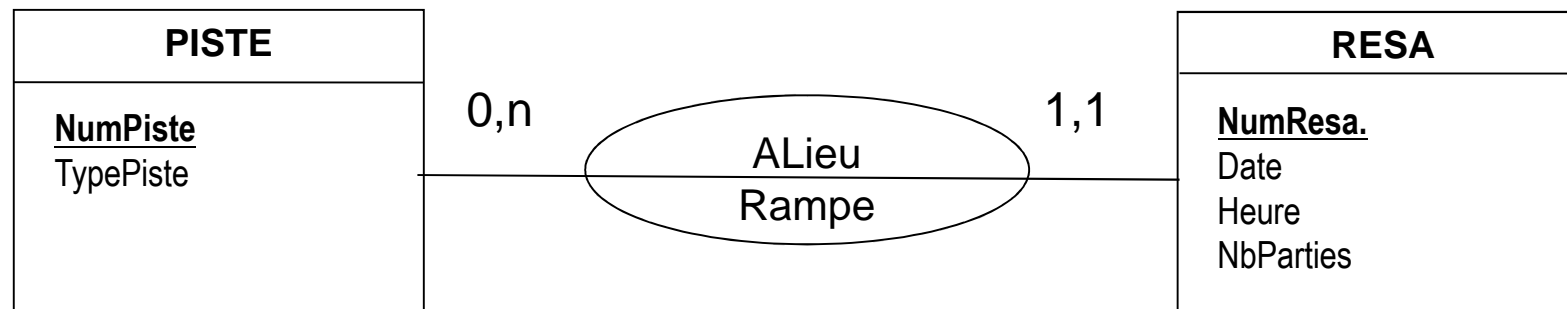
- La cardinalité minimale peut être 0, 1 ou n.
La cardinalité maximale peut être 1 ou n.

Cardinalité des associations

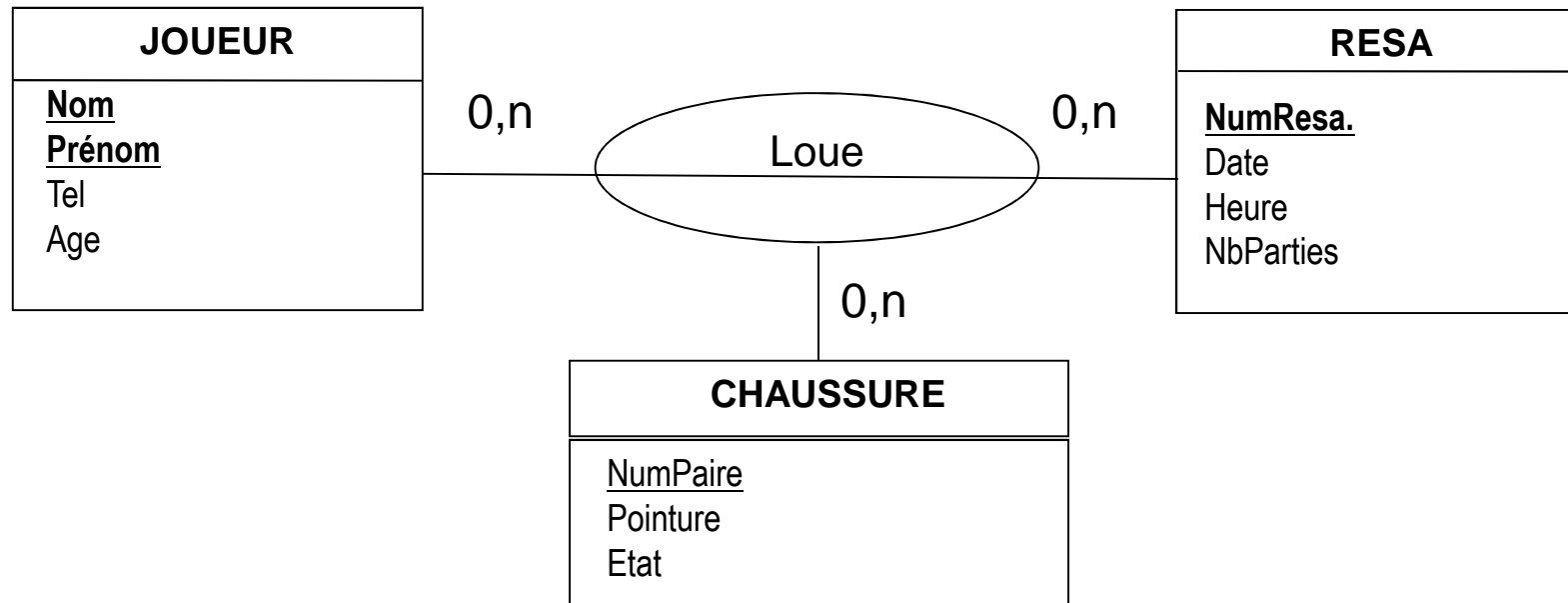
83



*RG : Une réservation concerne plusieurs joueurs, mais au moins un.
RG : Un joueur peut réserver plusieurs fois.*



*RG : Il existe différents types de pistes (normale, luxe, mini-piste)
RG : Une piste est affectée à chaque réservation, avec éventuellement une rampe de lancement (pour enfants ou personne handicapée)*



RG : Un joueur peut louer une paire de chaussures pour une réservation

- Une **contrainte d'intégrité** (CI) exprime une règle sur une propriété
- Il existe des contraintes de valeurs et des contraintes fonctionnelles (*dépendances entre les entités*)
- Les contraintes de valeurs ne se notent pas sur le MCD
- Les contraintes fonctionnelles (CIF) se notent sur le MCD et auront une influence sur le schéma relationnel

➤ Il existe différents types de contraintes de valeurs :

- domaines de valeurs

Exemple : l'état d'une chaussure est 'Neuf', 'Bon' ou 'Correct'

- contraintes statiques (*Les valeurs d'une propriété dépendent d'une ou plusieurs autres valeurs ou propriétés*)

Exemple : une pointure est entre 25 et 48

une piste de luxe est interdite aux enfants

- contraintes dynamiques (*lors d'un changement de la BD*)

Exemple : l'état d'une chaussure ne peut que se dégrader

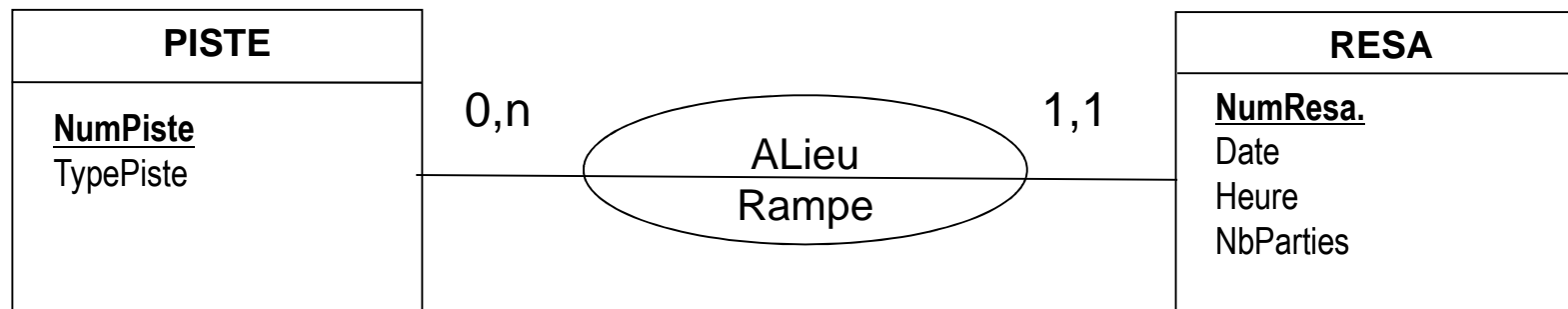
Contrainte d'Intégrité Fonctionnelle (CIF)

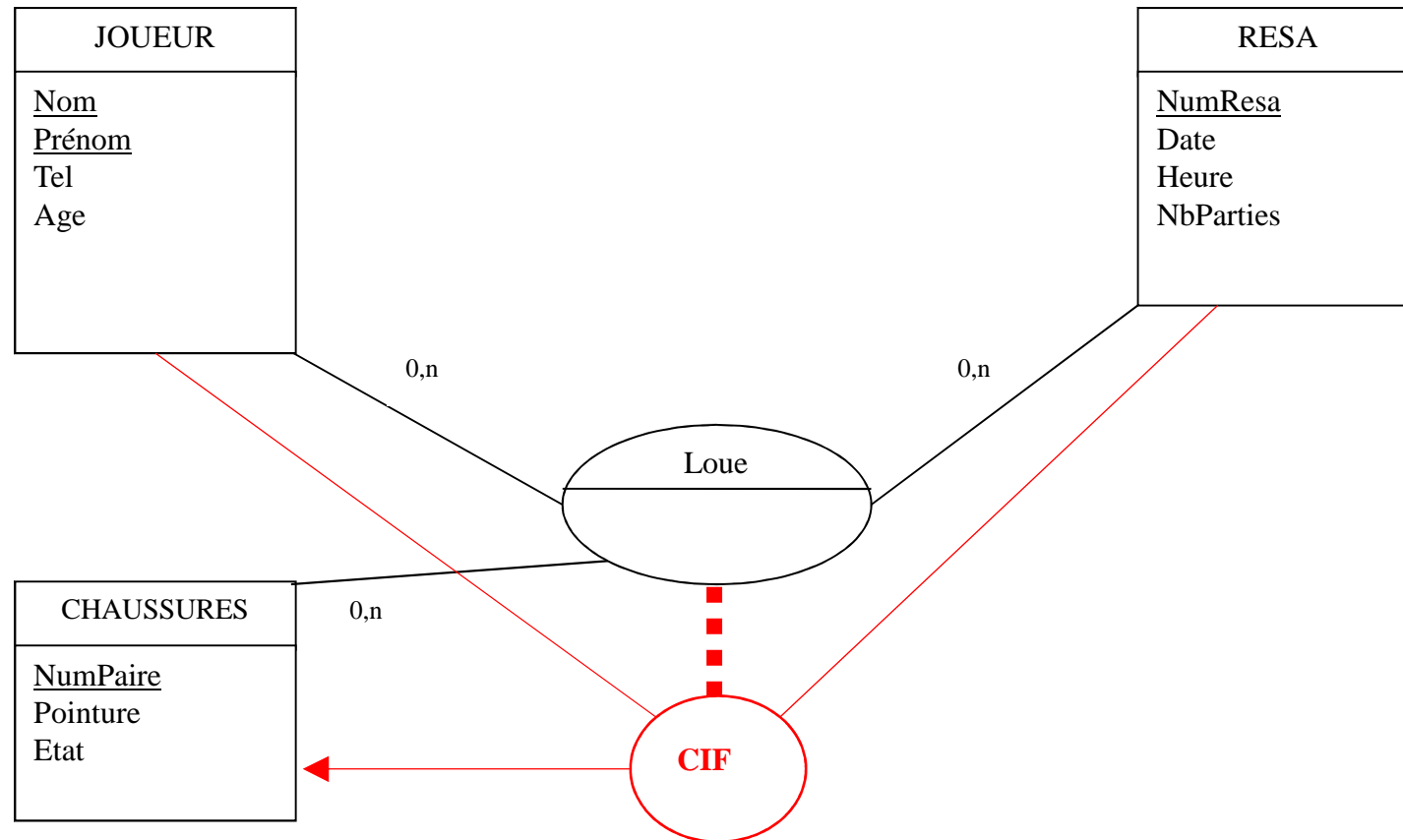
➤ Une CIF est définie sur une association

➤ Elle exprime une règle de gestion :

une entité participant à une association est totalement déterminée par la connaissance d'une ou plusieurs autres entités participant à cette même association

➤ CIF implicite : cardinalité 1,1

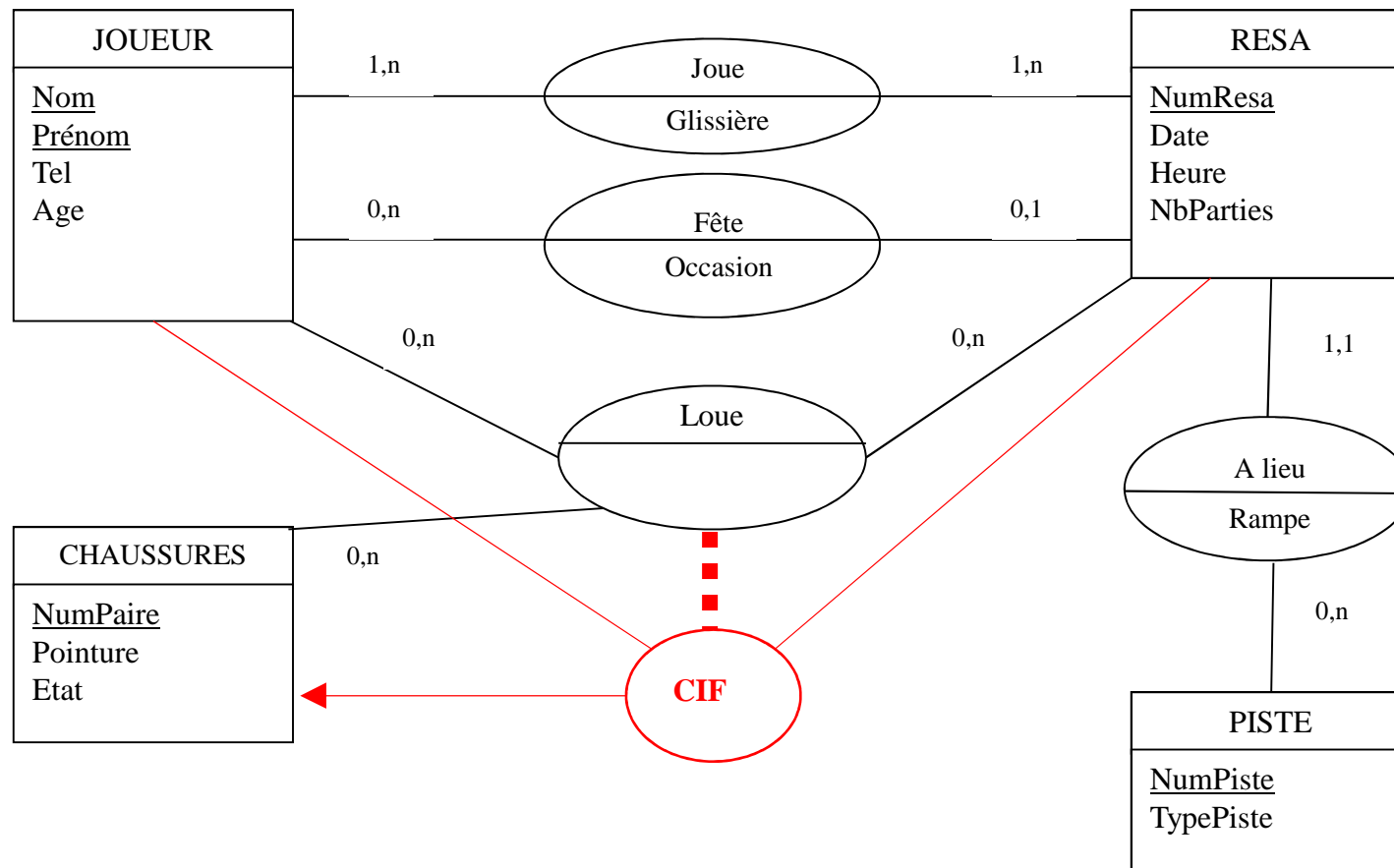




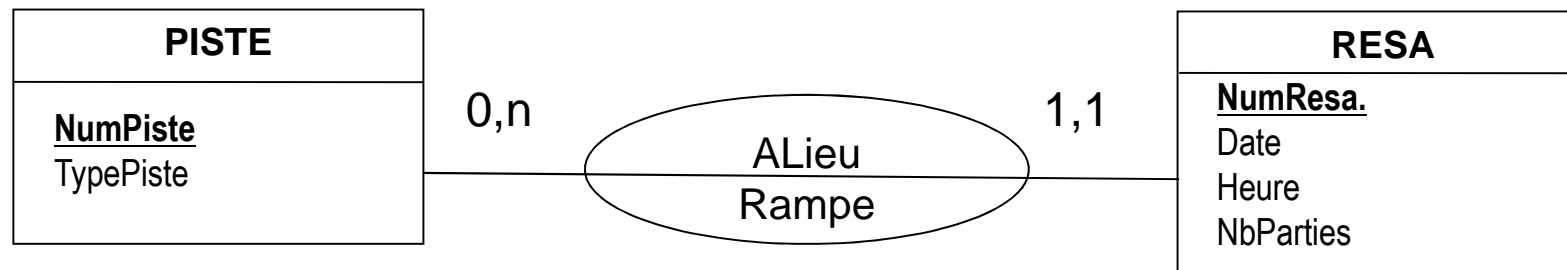
Lors d'une réservation, un joueur ne loue qu'une seule paire de chaussures

MCD : un exemple complet

89



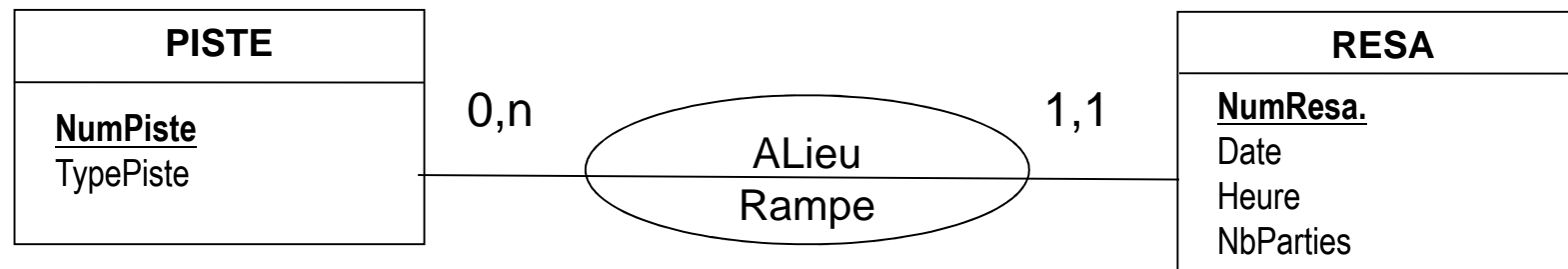
- **R0** : Chaque entité **devient une relation** dont :
- les attributs sont les propriétés de l'entité
 - la clé est l'identifiant de l'entité



PISTE (NumPiste, TypePiste)

RESA (NumResa, Date, Heure, NbParties)

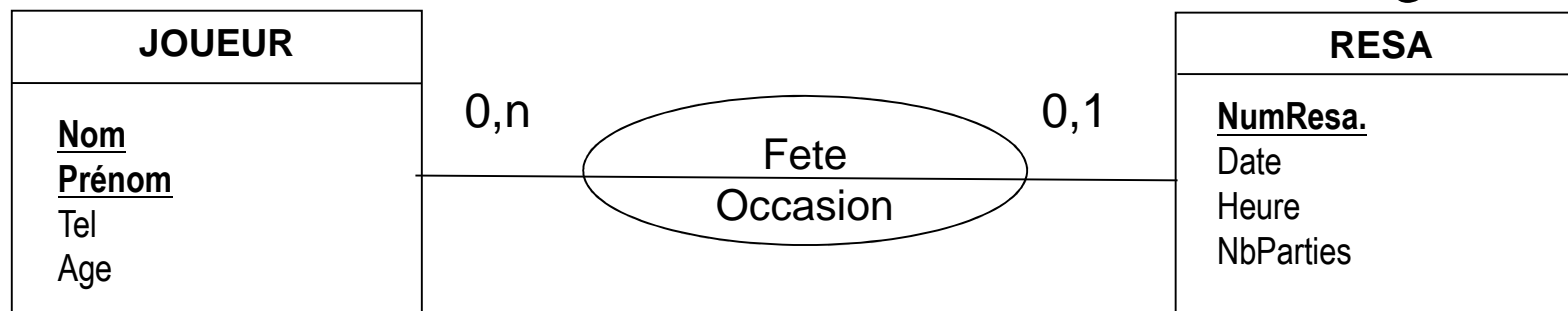
- **R1** : Toute association dont l'une des pattes a pour cardinalité 1,1 **détermine** dans la relation représentant l'entité associée à cette patte :
- des clés étrangères (identifiants des entités associées)
 - des propriétés (les propriétés propres de l'association)



PISTE (NumPiste, TypePiste)

RESA (NumResa, Date, Heure, NbParties, **#NumPiste**, Rampe)

- **R2** : Toute association dont l'une des pattes a pour cardinalité 0,1 **devient une nouvelle relation** dont :
- les attributs sont les propriétés de l'association et les identifiants des entités associées
 - la clé est l'identifiant de l'entité associée à la patte de cardinalité 0,1
 - les identifiants des entités associées sont clés étrangères.



JOUEUR (Nom, Prénom, Tel, Age)

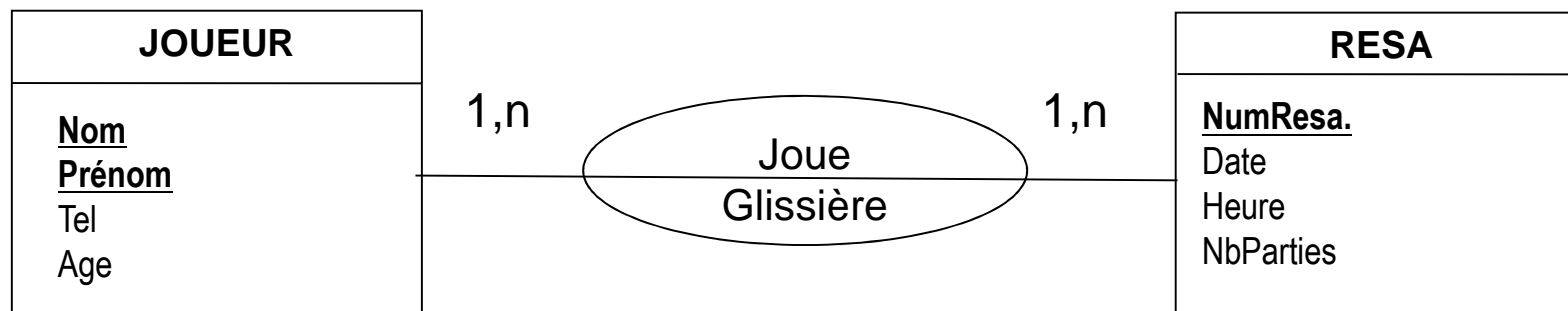
RESA (NumResa., Date, Heure, NbParties, #NumPiste, Rampe)

FETE (**#Nom**, **#Prénom**, **#NumResa.**, **Occasion**)

Dans le cas où les règles 1 et 2 ne s'appliquent pas

➤ **R3** : Toute association dont les pattes sont multivaluées devient une nouvelle relation dont :

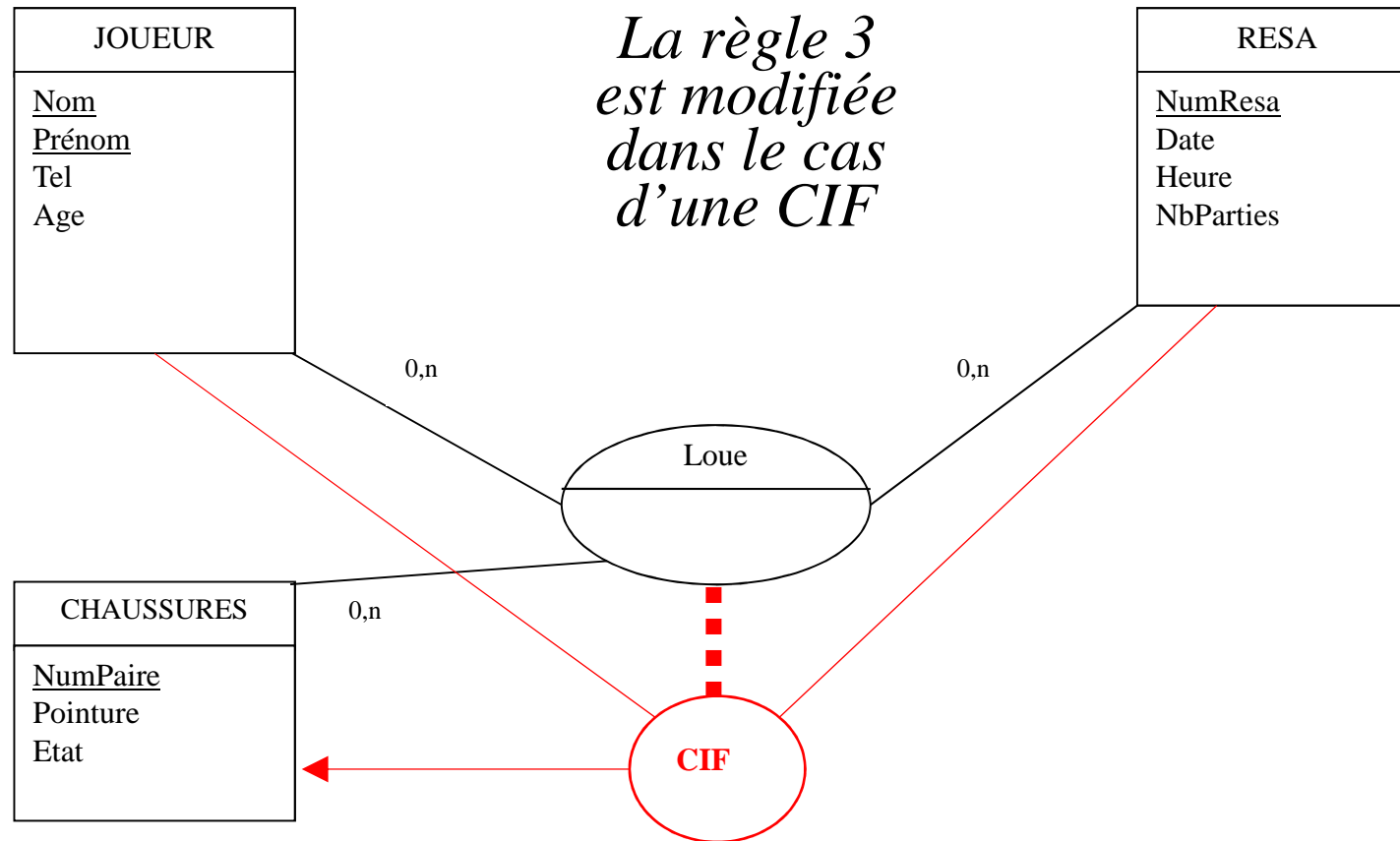
- les attributs sont les propriétés de l'association et les identifiants des entités associées
- la clé est formée des identifiants des entités associées qui sont aussi clés étrangères



JOUEUR (Nom, Prénom, Tel, Age)

RESA (NumResa., Date, Heure, NbParties, #NumPiste, Rampe)

JOUE (#Nom, #Prénom, #NumResa, Glissière)



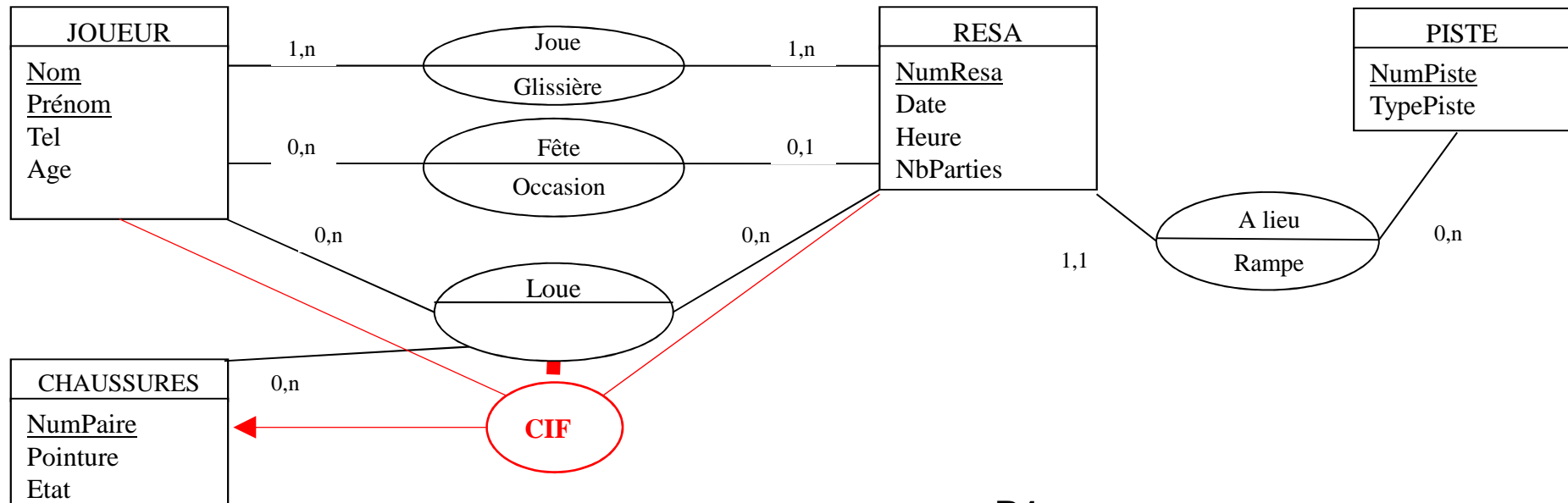
LOUE (#Nom, #Prénom, #NumResa, #NumPaire)

⇒

LOUE (#Nom, #Prénom, #NumResa, #**NumPaire**)

Exemple complet

95



RO :

JOUEUR (Nom, Prénom, Tel, Age)

RESA (NumResa, Date, Heure, NbParties , #NumPiste, Rampe)

PISTE (NumPiste, TypePiste)

CHAUSSURES (NumPaire, Pointure, Etat)

R2 :

FETE (#Nom, #Prenom, #NumResa, Occasion)

R3 :

JOUE (#Nom, #Prénom, #NumResa, Glissiere)

LOUE (#Nom, #Prénom, #NumResa, #NumPaire)

R1 :

