

Bases de données avancées

Master 1 IDSM-Kharkiv

2023-2024

Jérôme Darmont

<https://eric.univ-lyon2.fr/jdarmont/>



Actualité du cours



https://eric.univ-lyon2.fr/jdarmont/?page_id=3604



<https://eric.univ-lyon2.fr/jdarmont/?feed=rss2>



<https://social.sciences.re/@darmont> **#idsmbda**



Introduction

Définition

Base de données (BD) : Collection de données **cohérentes** et **structurées**

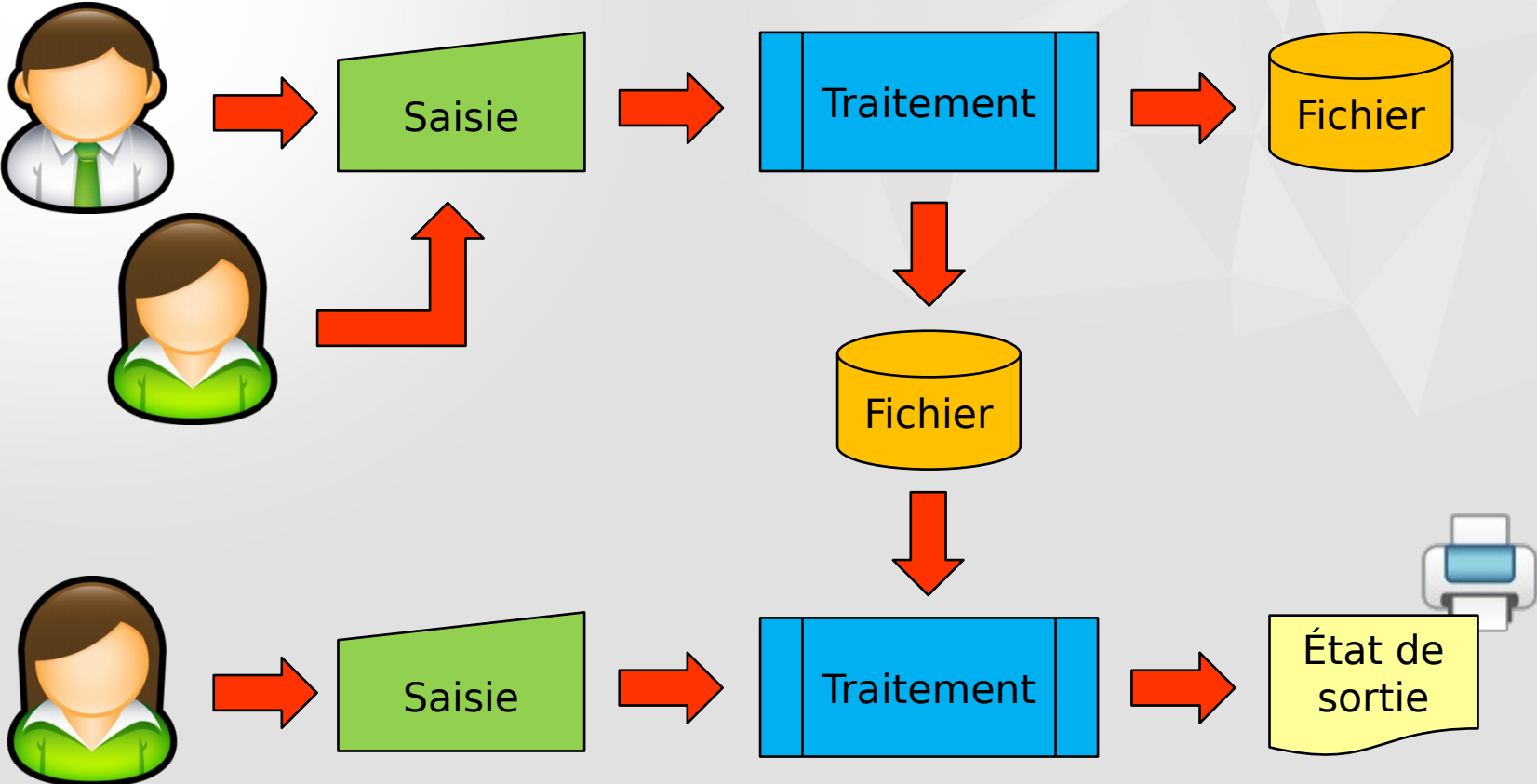


Base de données

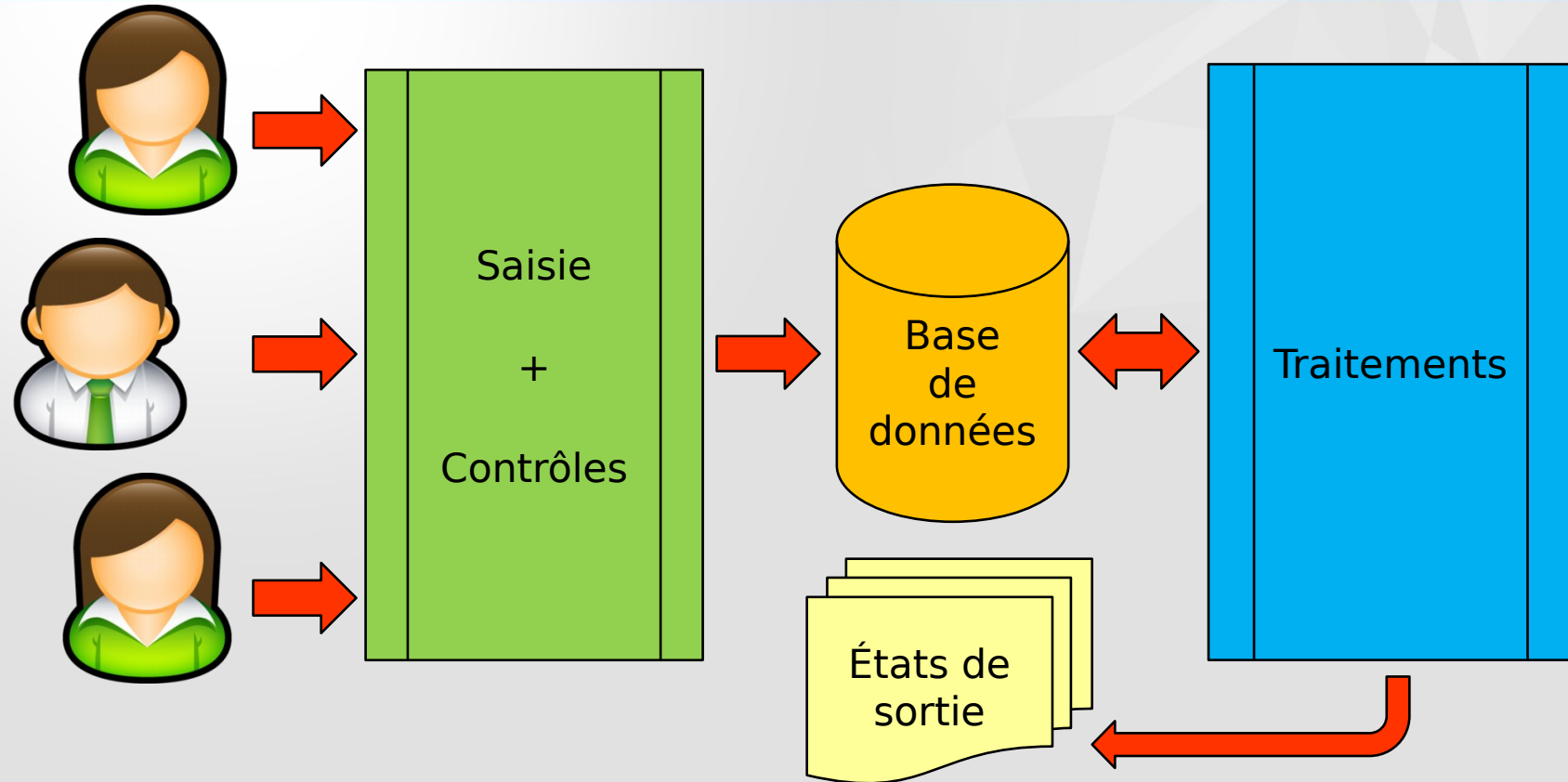


Fichiers

Organisation en fichiers



Organisation en BD



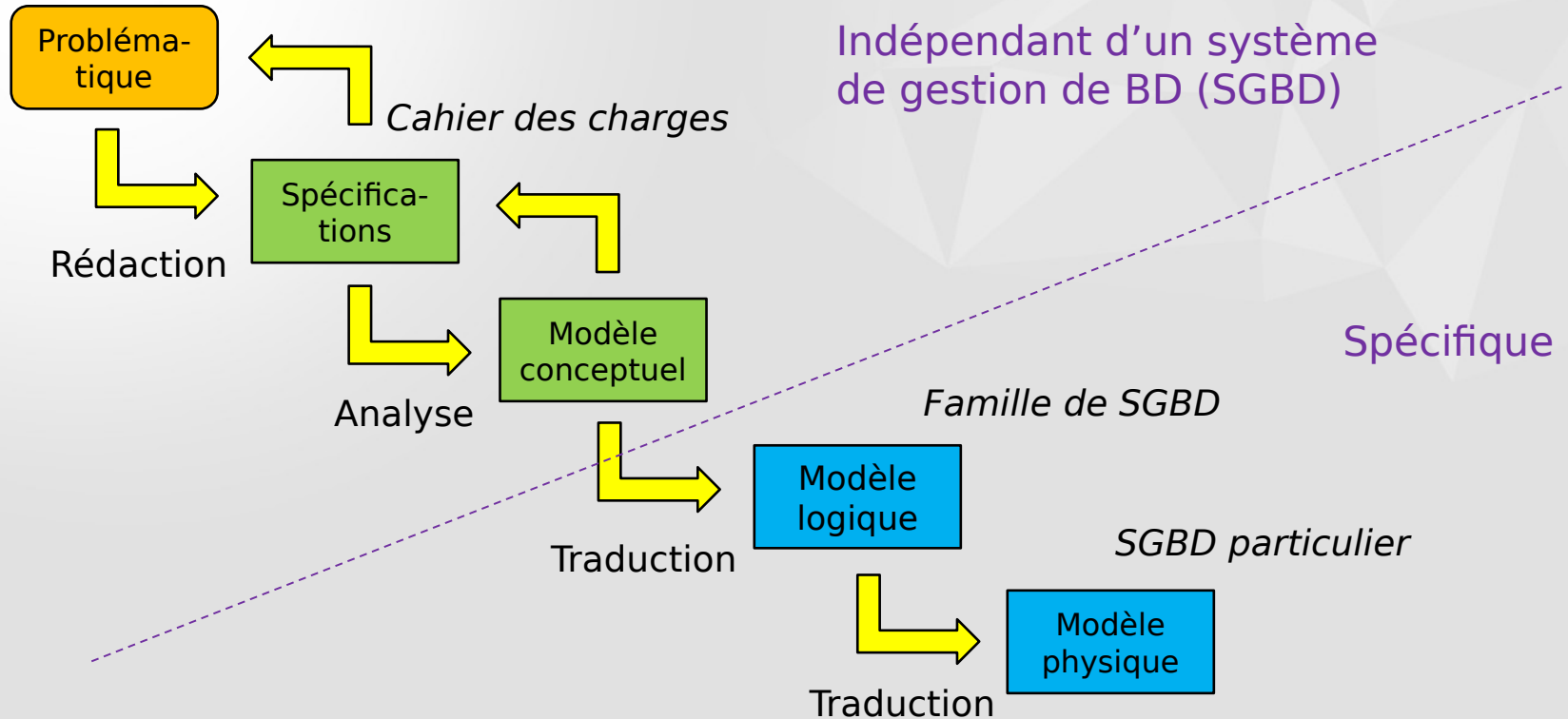
Avantages de l'organisation en BD

- ▶ Uniformisation de la saisie
- ▶ Standardisation des traitements
- ▶ Contrôle de la validité des données
- ▶ Partage de données entre plusieurs traitements

Qu'est-ce qu'un SGBD ?

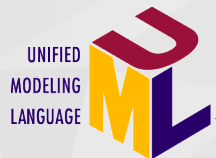
- ▶ **Système de Gestion de Bases de Données :** Logiciel(s) assurant structuration, stockage, maintenance, mise à jour et consultation des données d'une BD
- ▶ **Exemples**
 - SGBD « bureautiques » : Access, Base, Filemaker, Paradox
 - SGBD serveurs : Oracle, DB2, SQL Server, PostgreSQL, MySQL, MariaDB...

Processus de conception d'une BD



Plan du cours

▶ Partie 1 : Modélisation conceptuelle



▶ Partie 2 : Modélisation logique

Modèle relationnel

PubID	Publisher	PubAddress
03412822	Randam House	123 4th Street, New York
04773360	Wiley and Sons	45 Lincoln Blvd, Chicago
034859223	O'Reilly Press	77 Boston Ave, Cambridge
03-202088	Chia Light Books	99 Market, San Francisco

AuthorID	AuthorName	AuthorShip
145-20-238	Halle Gillisse	14-Aug-92
192-48-8965	Jon Stone	14-Mar-12
454-22-4912	Sally Hemmings	12-Sep-70
663-55-1254	Hannah Arendt	12-Mar-08

Title	AuthorID	PubID	Year
Cold Fusion for Dummies	1-34432-402-1	145-20-238	04-11-2002
Macrame and Straw Tying	1-38463-905-1	192-48-8965	04-7732003
Fluid Dynamics of Aqueducts	2-29521-409-4	454-22-4912	04-4959223
Beads, Baskets & Revolution	1-36278-293-4	663-55-1254	03-3920886

▶ Partie 3 : Interrogation et manipulation de bases de données

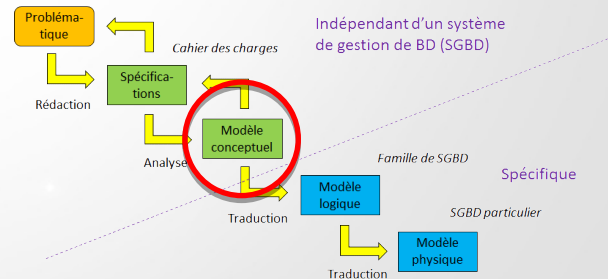


▶ Partie 4 : Programmation de bases de données



Partie 1

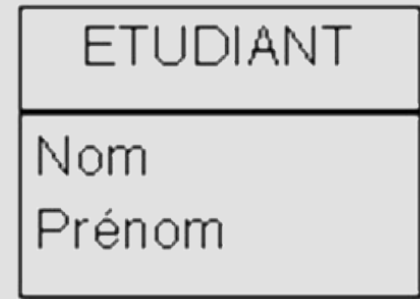
Modélisation conceptuelle



Modèle conceptuel UML

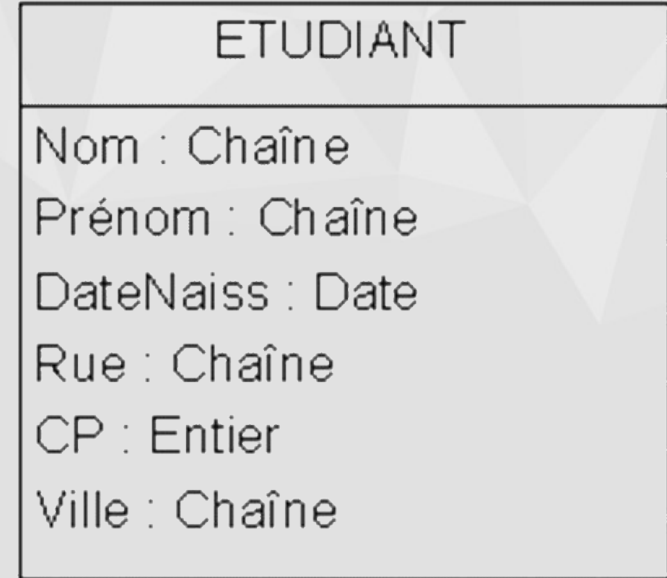
- ▶ **Standard** de l'Object Management Group
- ▶ Ensemble de **formalismes graphiques**
- ▶ **Diagramme de classes**

- ▶ **Classe** : Groupe d'entités du monde réel ayant les mêmes caractéristiques et le même comportement
ex. ETUDIANT
- ▶ **Attribut** : Propriété de la classe
ex. Nom et Prénom de l'étudiant·e
- ▶ **Représentation graphique** :



► Type d'attribut :

- Nombre entier (**Entier**)
- Nombre réel (**Réel**)
- Chaîne de caractères (**Chaîne**)
- Date (**Date**)



Instances

► Objets (individus) de la classe ETUDIANT = les étudiant-es

<u>Nom</u>	<u>Prénom</u>	<u>DateNaiss</u>	<u>Etc.</u>
Dupont	Albertine	01/06/1993	...
West	James	03/09/1994	...
Martin	Marie	05/06/1995	...
Abidi	Rachid	15/11/1995	...
Titgoutte	Justine	28/02/1996	...
Dupont	Noémie	18/09/1995	...
Dupont	Albert	23/05/1990	...

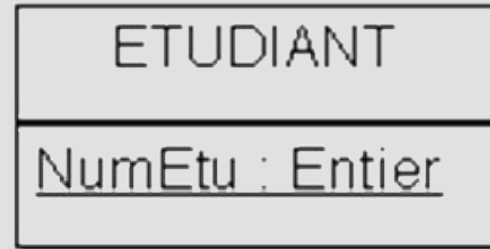
Problème : Comment distinguer les Dupont ?

Identifiant (1/2)

- ▶ Solution : Ajouter un attribut **numéro d'étudiant** !

<u>NumEtu</u>	<u>Nom</u>	<u>Prénom</u>	<u>DateNaiss</u>
1110	Dupont	Albertine	01/06/1993
2002	West	James	03/09/1994
3333	Martin	Marie	05/06/1995
4042	Durand	Rachid	05/11/1995
5552	Titgoutte	Justine	28/02/1996
6789	Dupont	Noémie	18/09/1995
7000	Dupont	Albert	23/05/1990

- ▶ Le numéro d'étudiant est un attribut **identifiant**.
- ▶ Un identifiant caractérise **de façon unique** les instances d'une classe.
- ▶ **Convention graphique :**
NB : Ne pas confondre avec les attributs de classe UML dont c'est la notation usuelle



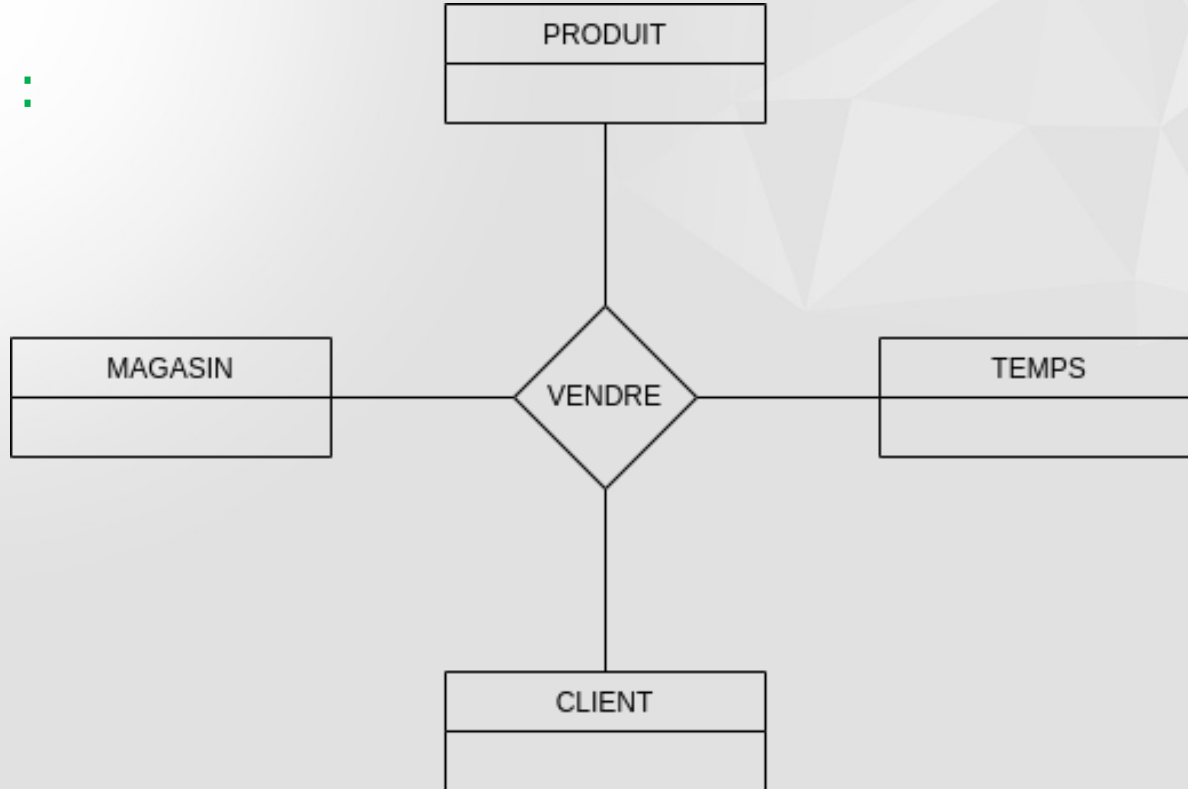
- ▶ **Association** : liaison perçue entre des classes
ex. Les étudiant·es passent des épreuves.



- ▶ Les classes ETUDIANT et EPREUVE peuvent être qualifiées de **participantes** à l'association PASSER.
- ▶ **Degré** ou **arité** d'une association : nombre de classes participantes.
En général : **associations binaires** (de degré 2).

Associations n-aires (degré $n > 2$)

Exemple :

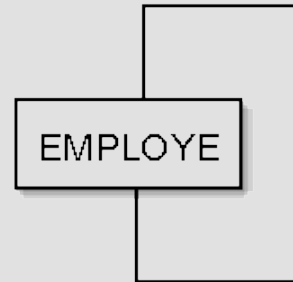


- ▶ Une classe peut être associée à elle-même, chaque instance pouvant jouer plusieurs rôles dans l'association.

ex. Employés et supérieurs hiérarchiques

+EST SUPERIEUR

- ▶ **Rôle** : fonction de chaque classe participante (+).



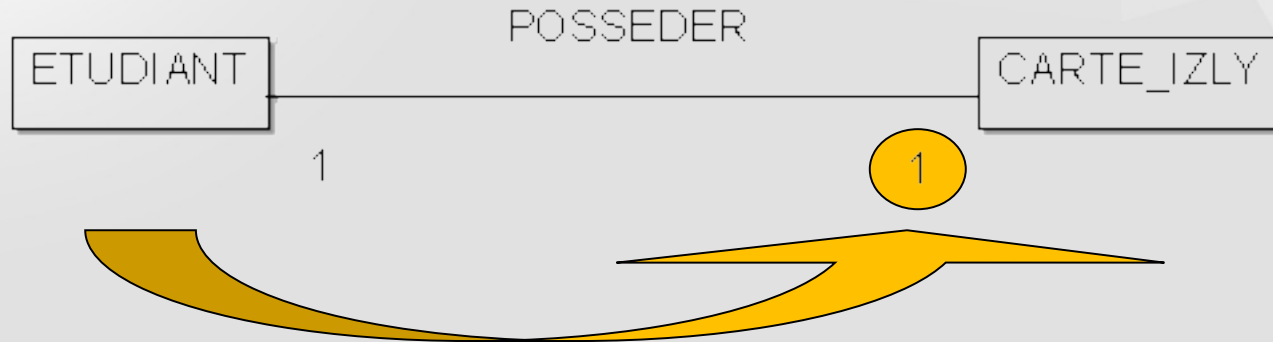
HIERARCHIE

+EST SUBALTERNE

- ▶ **Définition** : Indicateur qui montre combien d'instances de la classe considérée peuvent être liées à une instance de l'autre classe participant à l'association

- 1 Un et un seul
- 0..1 Zéro ou un
- 0..* ou * Zéro ou plus
- 1..* Un ou plus
- M..N De M à N (M, N entiers)
ex. 4..10 (de 4 à 10)

- ▶ ex. Un·e étudiant·e possède une et une seule carte Izly. Cette dernière n'est possédée que par un·e seul·e étudiant·e.



Lire « Un·e étudiant.e possède **multiplicité (1)** carte Izly ».

- ▶ ex. Une épreuve relève d'une et une seule matière.
Une matière peut donner lieu à plusieurs épreuves.



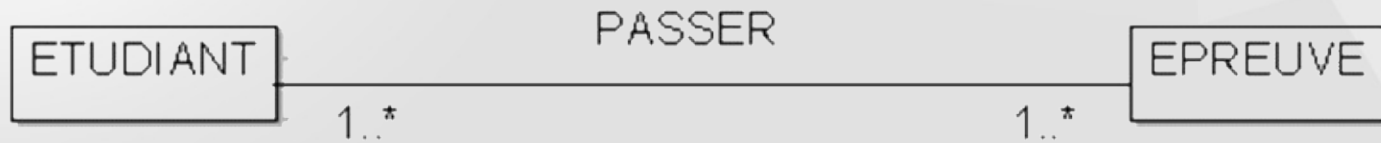
NB : La multiplicité un à plusieurs (1..*) peut aussi être zéro à plusieurs (0..* ou *).

- ▶ ex. Un·e étudiant·e peut appartenir ou non à un groupe de TD. Un groupe de TD réunit plusieurs étudiant·es.



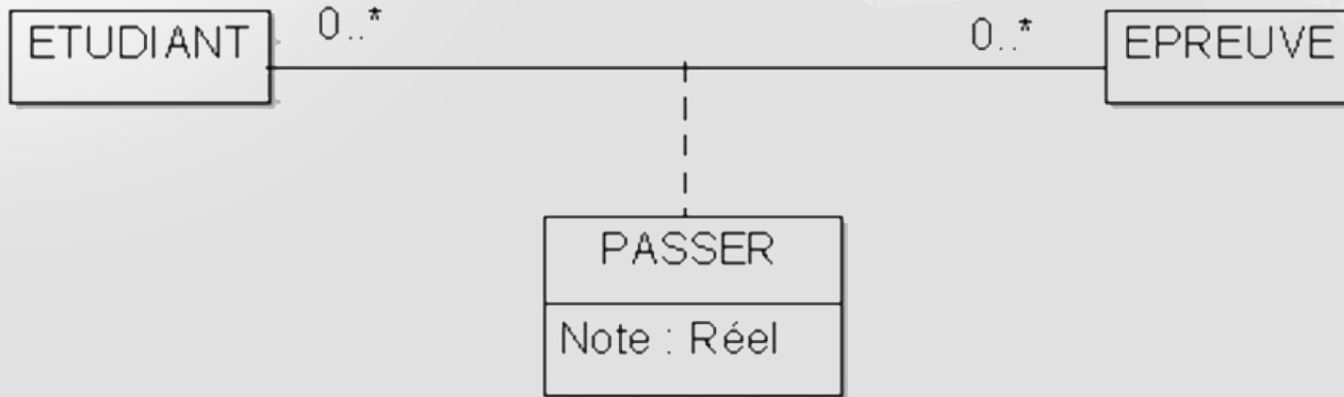
NB : La multiplicité un à plusieurs (1..*) peut aussi être zéro à plusieurs (0..* ou *).

- ▶ ex. Un·e étudiant·e peut passer plusieurs épreuves.
Une épreuve peut être passée par plusieurs étudiant·es.



NB : Les multiplicités un à plusieurs (1..*)
peuvent aussi être zéro à plusieurs (0..* ou *).

- ▶ Il est possible de caractériser une association par des attributs.
ex. Un·e étudiant·e qui passe une épreuve obtient une note.



NB : Une classe-association **est une association**, pas une classe.

Exemple : Spécifications (1/2)

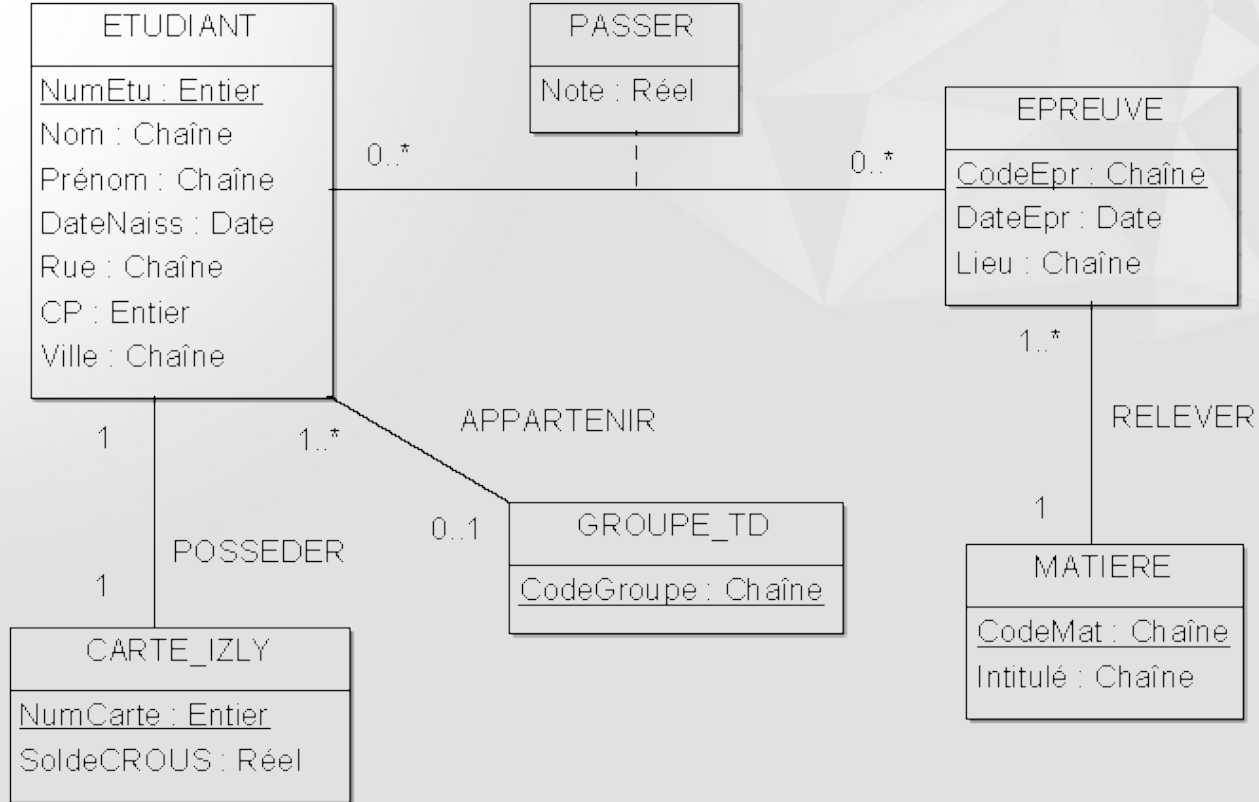
- ▶ Les étudiant·es sont caractérisé·es par un numéro unique, leur nom, prénom, date de naissance, rue, code postal et ville.
- ▶ Les étudiant·es possèdent une carte Izly caractérisée par un numéro unique et un solde d'argent utilisable au CROUS.
- ▶ Selon qu'ils ou elles sont dispensé·es ou non d'assiduité, les étudiant·es appartiennent à un groupe de TD caractérisé par un code unique.

Exemple : Spécifications (2/2)

- ▶ Les étudiant·es **passent des** épreuves et obtiennent une **note** pour chacune.
- ▶ Les **épreuves** sont caractérisées par un **code unique**, ainsi que la **date** et le **lieu** auxquels elles se déroulent.
- ▶ Chaque épreuve **relève d'une** matière unique (mais une matière donnée peut donner lieu à **plusieurs** épreuves).
- ▶ Les **matières** sont caractérisées par un **code unique** et un **intitulé**.

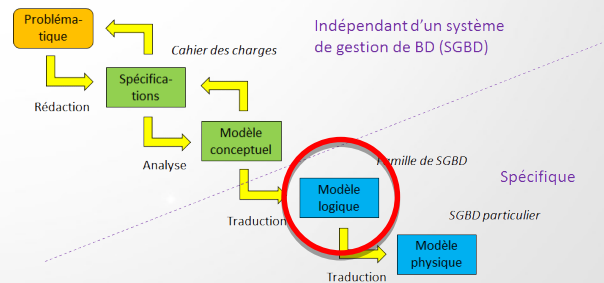
1. Identifier les **classes**
2. Identifier les **associations** entre les classes
3. Identifier les **attributs** de chaque classe et de chaque classe-association
4. Identifier et souligner l'**identifiant** de chaque classe
5. Évaluer les **multiplicités** des associations

Exemple : Diagramme de classes



Partie 2

Modélisation logique



Modèle logique relationnel

- ▶ **Modèle** associé aux SGBD relationnels
(ex. Oracle, SQL Server, DB2, PostgreSQL, MariaDB, MySQL...)
- ▶ **Objectifs du modèle relationnel**
 - Indépendance physique
 - Traitement du problème de redondance des données
 - Langages non procéduraux (faciles à utiliser)
 - Devenir un standard

Caractéristiques des SGBD relationnels

- ▶ Langages d'interrogation puissants et déclaratifs
- ▶ Accès orienté **valeur**
- ▶ Grande simplicité, absence de considérations physiques
- ▶ Description du schéma très réduite
- ▶ LDD intégré au LMD
- ▶ Grande dynamique de structure
- ▶ Optimisation de requêtes
- ▶ Utilisation interactive ou à partir d'un langage hôte

Relations et attributs

PubID	Publisher	PubAddress	AuthorID	AuthorName	AuthorCity
03-4472822	Random House	123 4th Street, New York	345-20-2820	Halle Delisse	14-Aug-92
04-7723951	Wiley and Sons	45 Lincoln Blvd, Chicago	312-40-8902	Joan Blow	14-Mar-95
03-4089223	O'Reilly Press	77 Boston Ave, Cambridge	616-25-4012	Sally Hemmings	13-Sep-70
02-202088	Ch. Light Books	68 Market, San Francisco	1503-550-1234	Hannah Arendt	13-Mar-08

PubID	AuthorID	PubID	AuthorID	Title
1-34312-402-1	345-20-2820	02-4472822	1893	Cold Fusion for Dummies
1-3882-895-1	312-40-8902	04-7723951	1895	Murama and Blow-Ting
2-39921-499-4	454-22-4012	03-4089223	1892	Fluid Dynamics of Aqueducts
1-38278-203-4	663-08-1234	02-3920888	1887	Evans, Baskets & Revolution

Modèle relationnel

- ▶ Une **relation** R est un ensemble d'**attributs** $\{A_1, A_2, \dots, A_n\}$.
ex. La relation EPREUVE est l'ensemble des attributs $\{\text{CodeEpr}, \text{DateEpr}, \text{Lieu}\}$.
- ▶ Chaque attribut A_i prend ses valeurs dans un **domaine** $\text{dom}(A_i)$.
ex. $\text{Note} \in [0, 20]$
 $\text{Lieu} \in \{\text{'Amphi Say'}, \text{'Amphi Aubrac'}, \text{'Salle D101'}, \dots\}$

PubID	Publisher	PubAddress	AuthorID	AuthorName	AuthorCity	Title	Year
03-4472822	Random House	123 4th Street, New York	345-20-2820	Halle Delassie	14-Aug-92		
04-7720921	Wiley and Sons	45 Lincoln Blvd, Chicago	302-49-8902	Jon Snow	14-Mar-15		
03-4898223	O'Reilly Press	77 Boston Ave, Cambridge	456-25-4012	Sally Hemmings	15-Sep-70		
02-2020888	City Lights Books	69 Market, San Francisco	100-50-1234	Hannah Arendt	13-Mar-08		

PubID	AuthorID	PubID	Title	Year
1-34532-402-1	345-20-2820	02-4472822	1980	Cold Fusion for Dummies
1-3882-895-1	392-88-9905	04-7720921	1985	Murders and Snow Time
2-39921-499-4	456-22-4012	03-4898223	1982	Fluid Dynamics of Aqueducts
1-92278-203-4	665-08-1234	02-3920888	1987	Ewads, Baskets & Revolution

► Notation d'une relation : $R (A_1, A_2, \dots, A_n)$

ex. EPREUVE (CodeEpr, DateEpr, Lieu)

► Un **n-uplet** t est un ensemble de valeurs $t = \langle V_1, V_2, \dots, V_n \rangle$ où $V_i \in \text{dom}(A_i)$ ou bien V_i est la valeur nulle (**NULL**).

ex. $\langle \text{'InfoS2'}, \text{'30-06-2016'}, \text{'Amphi Aubrac'} \rangle$ est un n-uplet de la relation EPREUVE.

Contraintes d'intégrité (1/2)

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-772093	Wiley and Sons	45 Lincoln Blvd, Chicago
03-489223	O'Reilly Press	77 Boston Ave, Cambridge
03-202088	Ch. Light Books	88 Market, San Francisco


AuthorID	Author Name	AuthorID
345-20-2820	Halle Gellesse	14-Aug-92
302-49-8962	Joel Elow	14-Mar-95
456-25-4012	Sally Hemmings	12-Sep-70
1503-50-1250	Hannah Arendt	12-Mar-08

Code	AuthorID	PubID	Code	Title
1-34513-492-1	345-20-2820	03-4472822	1980	Cold Fusion for Dummies
1-3882-895-1	392-88-9985	04-772093	1985	Muonons and Straw Tings
2-39921-499-4	456-25-4012	03-489223	1982	Fluid Dynamics of Aqueducts
1-38278-203-4	665-08-1250	03-392088	1987	Evads, Baskets & Revolution

Modèle relationnel

- ▶ **Clé primaire** : Ensemble d'attributs dont les valeurs permettent de distinguer les n-uplets les uns des autres.
ex. CodeEpr est clé primaire de la relation EPREUVE.
- ▶ **Clé étrangère** : Attribut qui est clé primaire d'une autre relation.
ex. Connaître la matière dont relève chaque épreuve
⇒ ajout de l'attribut CodeMat à la relation EPREUVE

Contraintes d'intégrité (2/2)



PubID	Publisher	PubAddress	AuthorID	AuthorName	AuthorAddress
03-4472822	Random House	123 4th Street, New York	345-20-2820	Halle Delassée	14-Aug-92
04-7733903	Wiley and Sons	45 Lincoln Blvd, Chicago	302-49-8980	Joan Elow	14-Mar-95
03-4892223	O'Reilly Press	77 Boston Ave, Cambridge	604-25-4012	Sally Hemmings	12-Sep-70
02-2020888	Ch. Light Books	69 Market, San Francisco	1503-550-7250	Hannah Arendt	12-Mar-00

Code	AuthorID	PubID	Code	Title
1-34812-492-1	345-20-2820	03-4472822	1980	Cold Fusion for Dummies
1-3882-895-1	302-49-8980	04-7733903	1985	Murama and Elow Ting
2-39921-499-4	454-22-4012	03-4892223	1952	Fluid Dynamics of Aqueducts
1-38278-203-4	605-08-1224	02-2020888	1967	Ewads, Baskets & Revolution

Modèle relationnel

- ▶ Notations : Clés primaires soulignées, clés étrangères postfixées par le caractère #.

ex. EPREUVE (CodeEpr, DateEpr, Lieu, CodeMat#)

- ▶ Contraintes de domaine : Les attributs doivent respecter une condition logique.

ex. Note ≥ 0 ET Note ≤ 20

Contraintes d'intégrité en pratique

PubID	Publisher	PubAddress	AuthorID	AuthorName	AuthorCountry
03-4472822	Pantheon House	123 4th Street, New York	145-20-2820	Halle Gellesse	14-Aug-92
04-7720903	Wiley and Sons	45 Lincoln Blvd, Chicago	302-49-8965	Joan Elbow	14-Mar-95
03-4898223	O'Reilly Press	77 Boston Ave, Cambridge	456-25-4012	Sally Hammings	13-Sep-70
02-2020888	Co. Light Books	68 Market, San Francisco	1503-55-7258	Harrold Avenue	13-Mar-08

Modèle relationnel

EPREUVE

MATIERE

<u>CodeEpr</u>	DateEpr	Lieu	Codemat#
ECOS101	15/01/2016	Amphi Aubrac	ECO
ECOS102	16/01/2016	Amphi Aubrac	ECO
GESS201	25/05/2016	Salle 201	GES
INFOS101	20/01/2016	Salle 101	INFO

<u>CodeMat</u>	Intitulé
ECO	Économie
GES	Gestion
INFO	Informatique

Traduction UML-relational (1/4)

- ▶ Chaque **classe** devient une **relation**.
- ▶ Les **attributs** de la classe deviennent **attributs** de la relation.
- ▶ L'**identifiant** de la classe devient **clé primaire** de la relation.

ex. ETUDIANT (NumEtu, Nom, Prénom, DateNaiss, Rue, CP, Ville)

Traduction UML-relational (2/4)

- ▶ Chaque **association 1-1** est prise en compte en incluant la clé primaire d'une des relations participante comme **clé étrangère** dans l'autre relation.

ex. CARTE_IZLY (NumCarte, SoldeCROUS)

ETUDIANT (NumEtu, Nom, Prénom, DateNaiss, Rue, CP, Ville, **NumCarte#**)

Traduction UML-relational (3/4)

- ▶ Chaque **association 1-N** est prise en compte en incluant la clé primaire de la relation dont la **multiplicité maximale est 1** comme **clé étrangère** dans l'autre relation participante.

ex. EPREUVE (CodeEpr, DateEpr, Lieu, **CodeMat#**)

MATIERE (CodeMat, Intitulé)

Traduction UML-relational (4/4)

- ▶ Chaque **association M-N** est prise en compte en créant une nouvelle relation dont la clé primaire est la **concaténation des clés primaires** des relations participantes. Les attributs de la classe-association sont insérés dans cette nouvelle relation si nécessaire.

ex. PASSER (NumEtu#, CodeEpr#, Note)

Exemple : Modèle logique relationnel

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7723903	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4898223	O'Reilly Press	77 Boston Ave, Cambridge
03-2020288	Co. Light Books	88 Market, San Francisco

AuthorID	Author Name	AuthorCity
345-20-2820	Halle Delisse	14-Aug-92
302-49-8960	Jon Elmer	14-Mar-92
456-25-4012	Sally Hemmings	13-Sep-70
1503-59-7250	Harrold Jenett	13-Mar-00

Code	AuthorID	PubID	Title
1-34432-492-1	345-20-2820	03-4472822	1993 Cold Fusion for Dummies
1-3882-895-1	392-88-9980	04-7723903	1985 Microsoft and Other Things
2-39921-499-4	456-25-4012	03-4898223	1992 Fluid Dynamics of Aquabots
1-92278-203-4	665-98-1254	03-3920886	1987 Ewads, Baskets & Revolution

Modèle relationnel

CARTE_IZLY (NumCarte, SoldeCROUS)

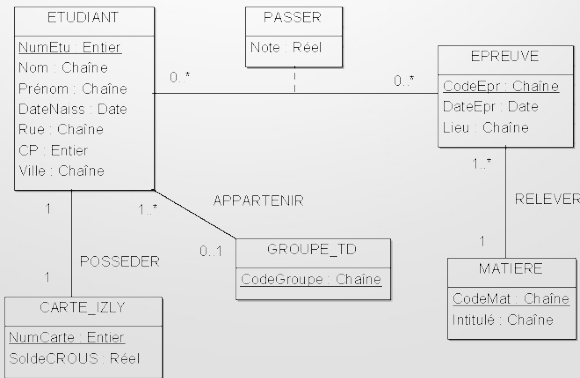
GROUPE_TD (CodeGroupe)

ETUDIANT (NumEtu, Nom, Prénom, DateNaiss, Rue, CP, Ville, NumCarte#, CodeGroupe#)

MATIERE (CodeMat, Intitulé)

EPREUVE (CodeEpr, DateEpr, Lieu, CodeMat#)

PASSER (NumEtu#, CodeEpr#, Note)



Traduction d'une association M-N

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7729903	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4898223	O'Reilly Press	77 Boston Ave, Cambridge
02-202088	Ch. Light Books	88 Market, San Francisco

AuthorID	AuthorName	AuthorAddress
345-20-2820	Halle Gellesse	14-Aug-92
302-49-8965	Jon Elmer	14-Mar-16
456-25-4012	Sally Hammings	13-Sep-70
1603-59-729	Heleno Azevedo	13-Mar-08

Book	AuthorID	PubID	Book Title	Year
1-34833-492-1	345-20-2820	03-4472822	1990 Cold Fusion for Dummies	
1-3982-895-1	392-48-9985	04-7729903	1985 Mathematics and Other Things	
2-39931-499-4	456-25-4012	03-4898223	1992 Fluid Dynamics of Aqueducts	
1-98278-203-4	665-08-1254	03-3920886	1987 Ewads, Baskets & Revolution	

Modèle relationnel

ETUDIANT

NumEtu	Nom	Prénom
1110	Dupont	Albertine
2002	West	James

PASSER (table « pont »)

NumEtu#	CodeEpr#	Note
1110	INFOS101	15,5
2002	ECOS101	8,5
2002	ECOS102	13
1110	GESS201	14
2002	GESS201	14,5

EPREUVE

CodeEpr	DateEpr	Lieu
ECOS101	15/01/2016	Aubrac
ECOS102	16/01/2016	Aubrac
GESS201	25/05/2016	D201
INFOS101	20/01/2016	D101

Problème de la redondance

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7723903	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4898223	O'Reilly Press	77 Boston Ave, Cambridge
03-202088	Ch. Light Books	88 Market, San Francisco

AuthorID	Author Name	AuthorCity
345-20-2820	Halle Gellesse	14-Aug-92
392-49-8980	Jon Blow	14-Mar-95
456-25-4012	Sally Hammings	13-Sep-70
150-50-7250	Harrold Averett	13-Mar-00

Title	AuthorID	PubID	Price	Title
1-34333-492-1	345-20-2820	03-4472822	1990	Cold Fusion for Dummies
1-3882-895-1	392-49-8980	04-7723903	1895	Muramasa and Blow-Ting
2-39921-499-4	456-25-4012	03-4898223	1982	Fluid Dynamics of Aqueducts
1-38278-213-4	665-08-1224	03-3920886	1987	Evans, Buckets & Revolutions

Modèle relationnel

- ▶ Lorsque l'on effectue directement une modélisation logique ex. Soit la relation PASSER_EPREUVE.

<u>NumEtu</u>	<u>Note</u>	<u>CodeEpr</u>	<u>Lieu</u>
1110	15,5	INFOS101	Amphi Aubrac
1110	14,0	ECOS101	Amphi Aubrac
2002	13,0	ECOS102	Salle D201
3333	10,5	INFOS101	Amphi Aubrac

Cette relation présente différentes anomalies.

Anomalies liées à la redondance

PubID	Publisher	PubAddress
034473822	Random House	121 4th Street, New York
047739031	Wiley and Sons	45 Lincoln Blvd, Chicago
034850223	O'Reilly Press	77 Boston Ave, Cambridge
023020808	Ch. Light Books	99 Market, San Francisco

AuthorID	J. last Name	AuthorCountry
1340-20-2030	Halle Gelasca	14-Aug-92
262-24-8905	Joel Blou	14-Mar-16
454-24-4012	Baily Hammings	12-Sep-70
160-58-1254	Harrah Jenett	12-Mar-08

Course	AuthorID	BookID	Book Title	Year
1-34433-432-1	345-20-2030	03-4472822	1993	Cold Fusion for Dummies
1-3842-895-1	392-48-9985	04-7733063	1995	Microware and Straw-Twig
2-39521-489-4	454-22-4012	03-4893223	1952	Fluid Dynamics of Aqueducts
1-38278-293-4	463-05-1254	02-3920808	1987	Reads, Baskets & Revolution

Modèle relationnel

- ▶ Anomalies de modification : Si l'on souhaite mettre à jour le lieu d'une épreuve, il faut le faire pour tous les n-uplets concernés.
- ▶ Anomalies d'insertion : Pour ajouter une nouvelle épreuve, il faut obligatoirement fournir des valeurs pour **NumEtu** et **Note**.
- ▶ Anomalies de suppression
ex. La suppression de l'étudiant n° **2002** fait perdre toutes les informations concernant l'épreuve **ECOS102**.

Éviter la redondance

► Pourquoi ?

- Suppression des problèmes de mise à jour
- Minimisation de l'espace de stockage

► Comment ?

- **Ne pas** travailler directement au niveau logique ou physique
- Adopter le processus : modèles conceptuel → logique → physique
- Dans le modèle conceptuel, ne spécifier que des attributs **non décomposables** (première forme normale).
ex. Une adresse doit être décomposée en rue, code postal, ville...
- C'est tout !



Partie 3

Interrogation et manipulation de bases de données

Qu'est-ce que l'algèbre relationnelle ?

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7723953	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4898223	O'Reilly Press	77 Boston Ave, Cambridge
02-202088	Ch. Light Books	88 Market, San Francisco

AuthorID	Author name	Publication
345-20-2820	Halle Gellesse	14-Aug-92
302-49-8962	Joan Elow	14-Mar-92
456-25-4012	Sally Hemmings	12-Sep-70
150-50-7250	Harrold Arentz	12-Mar-00

PubID	PubID	PubID	PubID	PubID
1-34533-492-1	345-20-2820	03-4472822	1980	Cold Fusion for Dummies
1-3882-895-1	392-88-9985	04-7723953	1985	Murama and Elow Ting
2-39921-499-4	456-22-4012	03-4898223	1982	Fluid Dynamics of Aquabubbs
1-38278-203-4	665-08-1224	03-3920886	1987	Evads, Baskets & Revolution

Modèle relationnel

- ▶ Ensemble d'opérateurs qui s'appliquent aux relations
- ▶ **Résultat** : nouvelle relation qui peut à son tour être manipulée
- ⇒ L'algèbre relationnelle permet d'effectuer des recherches dans les relations.

Opérateurs ensemblistes (1/5)

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7723901	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4898223	O'Reilly Press	77 Boston Ave, Cambridge
03-2020888	Co. Light Books	68 Market, San Francisco

AuthorID	Author Name	AuthorAddress
145-20-2820	Halle Delisse	14-Aug-92
102-49-8980	Joel Elow	14-Mar-92
456-25-4012	Sally Hammings	13-Sep-70
100-50-7250	Harrold Avenel	13-Mar-00

Event	AuthorID	PubID	Event	Title
1-34532-492-1	145-20-2820	03-4472822	1990	Cold Fusion for Dummies
1-3882-895-1	102-49-8980	04-7723901	1985	Murphy and Elow-Ting
2-39921-499-4	456-25-4012	03-4898223	1982	Fluid Dynamics of Aqueducts
1-92278-203-4	683-08-1224	03-3920888	1987	Evans, Baskets & Revolution

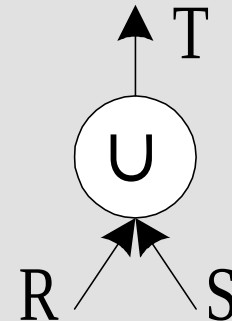
Modèle relationnel

- ▶ Union : $T = R \cup S$ (notation algébrique)
ou $T = \text{UNION}(R, S)$ (notation fonctionnelle)

R et S doivent avoir même schéma.

ex. R et S sont les relations ETUDIANT de deux formations (ex. anciens M1 Finance et Eco-Société) fusionnées pour constituer une liste d'émargement commune.

Notation graphique :



Opérateurs ensemblistes (2/5)

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7723951	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4892223	O'Reilly Press	77 Boston Ave, Cambridge
02-2020888	Ch. Light Books	88 Market, San Francisco

AuthorID	Author Name	AuthorCity
345-20-2820	Halle Delacoste	14-Aug-92
302-48-8982	Joan Elbow	14-Mar-92
456-25-4012	Sally Hemmings	12-Sep-70
150-50-7254	Harrold Averett	12-Mar-00

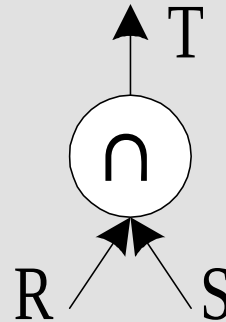
Course	AuthorID	PubID	Enroll	Title
1-34531-402-1	345-20-2820	03-4472822	1992	Cold Fusion for Dummies
1-3882-895-1	302-48-8982	04-7723951	1985	Mathematics and Snow Timing
2-35921-499-4	456-25-4012	03-4892223	1982	Fluid Dynamics of Aqueducts
1-38278-203-4	150-50-7254	02-2020888	1987	Evans, Buckets & Revolutions

► Intersection : $T = R \cap S$
ou $T = \text{INTERSECT} (R, S)$

R et S doivent avoir même schéma.

ex. Permet de trouver les étudiant·es commun·es à deux formations.

Notation graphique :



Opérateurs ensemblistes (3/5)

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Stree, New York
04-7723951	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4898223	O'Reilly Press	77 Boston Ave, Cambridge
03-2020888	Ch. Light Books	88 Market, San Francisco

AuthorID	Author name	Publication
345-20-2820	Halle Delassée	14-Aug-92
302-49-8965	Joan Elbow	14-Mar-95
456-25-4012	Sally Hammings	12-Sep-70
1503-59-7254	Hannah Arendt	12-Mar-00

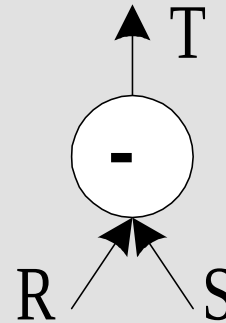
PubID	AuthorID	PubID	PubID	Title
1-34432-492-1	345-20-2820	03-4472822	1980	Cold Fusion for Dummies
1-3882-895-1	392-88-9985	04-7723951	1985	Murder and Blow-Ting
2-39921-499-4	456-25-4012	03-4898223	1982	Fluid Dynamics of Aqueducts
1-38278-203-4	665-98-1254	03-3920888	1987	Evans, Baskets & Revolution

- ▶ Différence : $T = R - S$
ou $T = \text{MINUS}(R, S)$

R et S doivent avoir même schéma.

ex. Permet de retirer les étudiant·es de la relation S existant dans la relation R.

Notation graphique :



Opérateurs ensemblistes (4/5)

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7723903	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4892223	O'Reilly Press	77 Boston Ave, Cambridge
03-2020888	Ch. Light Books	88 Market, San Francisco

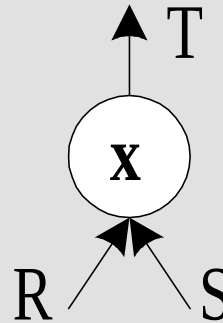
AuthorID	Author Name	Publication
345-20-2820	Halle Delassée	14-Aug-92
302-49-8962	Joan Blow	14-Mar-95
456-25-4012	Sally Hammonds	15-Sep-70
150-50-7254	Hannah Arendt	13-Mar-08

Event	AuthorID	PubID	Event	Title
1-34532-492-1	345-20-2820	03-4472822	1980	Cold Fusion for Dummies
1-3882-895-1	392-88-9985	04-7723903	1985	Murders and Blow-Ting
2-39921-499-4	456-25-4012	03-4892223	1982	Fluid Dynamics of Aqueducts
1-38278-203-4	665-98-1254	03-3920888	1987	Evans, Baskets & Revolution

- ▶ **Produit cartésien** : $T = R \times S$
ou $T = \text{PRODUCT}(R, S)$

Associe chaque n-uplet de R à chaque n-uplet de S.

Notation graphique :



Produit cartésien

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7723951	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4498223	O'Reilly Press	77 Boston Ave, Cambridge
03-2020888	Ch. Light Books	99 Market, San Francisco

AuthorID	Author Name	AuthorAddress
345-20-2920	Halle Gellesse	14-Aug-92
302-49-8982	Joan Elow	14-Mar-95
456-25-4012	Sally Hemmings	12-Sep-70
100-50-1254	Hannah Arendt	12-Mar-00

Code	ProjectID	ProjID	Code	Title
1-34313-402-1	345-20-2920	03-4472822	1980	Cold Fusion for Dummies
1-3882-895-1	392-88-9985	04-7723951	1985	Murphy and Brown Thing
2-35921-499-4	456-22-4012	03-4498223	1982	Fluid Dynamics of Aqueducts
1-38278-203-4	665-08-1254	03-3920888	1987	Evans, Baskett & Revolution

Modèle relationnel

ex.

<u>NumEtu</u>	<u>Nom</u>
101	E1
102	E2

X

<u>CodeEprLieu</u>	
INFO1	Aubrac
ECO1	Aubrac
ECO2	D201

=

<u>NumEtu</u>	<u>Nom</u>	<u>CodeEprLieu</u>	
101	E1	INFO1	Aubrac
102	E2	INFO1	Aubrac
101	E1	ECO1	Aubrac
102	E2	ECO1	Aubrac
101	E1	ECO2	D201
102	E2	ECO2	D201

Opérateurs ensemblistes (5/5)

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7723901	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4898223	O'Reilly Press	77 Boston Ave, Cambridge
03-2020888	Ch. Light Books	88 Market, San Francisco

AuthorID	AuthorName	Publication
345-20-2820	Halle Delisse	14-Aug-92
345-40-8900	Joan Elow	14-Mar-92
456-25-4012	Sally Hemmings	12-Sep-70
1503-50-1254	Harrold Averett	12-Mar-00

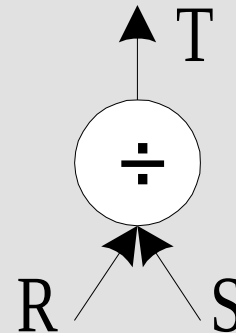
PubID	AuthorID	PubID	AuthorID	Title
1-34513-401-1	345-20-2820	03-4472822	1980	Cold Fusion for Dummies
1-34862-895-1	392-80-8900	04-7723901	1985	Murder and Blow-Ting
2-39821-499-4	456-22-4012	03-4898223	1982	Fluid Dynamics of Aqueducts
1-38278-213-4	665-08-1254	03-3920888	1987	Evads, Baskets & Revolution

► **Division** : $T = R \div S$
ou $T = \text{DIVISION}(R, S)$

$R(A_1, A_2, \dots, A_n) S(A_{p+1}, \dots, A_n)$

$T(A_1, A_2, \dots, A_p)$ contient tous les n-uplets tels que leur concaténation à **chacun** des n-uplets de S donne toujours un n-uplet de R.

Notation graphique :



Division

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7723903	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4898223	O'Reilly Press	77 Boston Ave, Cambridge
03-2020888	Ch. Light Books	88 Market, San Francisco

AuthorID	Author Name	AuthorEntry
345-20-2820	Halle Gellesse	14-Aug-92
302-49-8965	Jon Elmer	14-Mar-95
456-25-4012	Sally Hemmings	15-Sep-70
1503-50-1254	Harrold Averell	13-Mar-00

Code	AuthorID	PubID	Entry	Title
1-34532-482-1	345-20-2820	03-4472822	1993	Cold Fusion for Dummies
1-38862-896-1	302-49-8965	04-7723903	1995	Murphy and Brown Thing
2-39921-499-4	456-25-4012	03-4898223	1992	Fluid Dynamics of Aqueducts
1-38278-203-4	665-08-1254	03-3920888	1987	Evans, Buckets & Revolution

Modèle relationnel

ex.

<u>NumEtu</u>	<u>CodeEpr</u>	<u>Note</u>
101	INFO1	11
101	ECO1	15
101	ECO2	12
102	ECO1	9
103	INFO1	11
103	ECO2	12

÷

<u>CodeEpr</u>	<u>Note</u>
INFO1	11
ECO2	12

=

<u>NumEtu</u>
101
103

Opérateurs spécifiques (1/3)

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7723951	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4892223	O'Reilly Press	77 Boston Ave, Cambridge
03-2020888	Ch. Light Books	88 Market, San Francisco

AuthorID	AuthorName	AuthorSalary
345-20-2820	Halle Delacoste	14-Aug-92
302-48-8980	Joan Elbow	14-Mar-92
456-25-4012	Sally Hammings	12-Sep-70
150-30-1234	Harrold Avenel	12-Mar-80

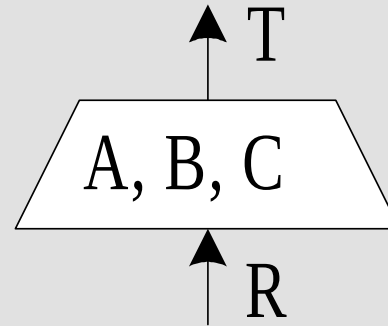
ProjID	ProjDesc	ProjStart	ProjEnd	
1-34532-482-1	345-20-2820	03-4472822	1993	Cold Fusion for Dummies
1-3882-895-1	392-88-9980	04-7723951	1985	Murama and Elbow Time
2-39921-499-4	456-22-4012	03-4892223	1992	Fluid Dynamics of Aqueducts
1-38278-203-4	665-08-1234	03-3920888	1987	Evans, Baskette & Revolution

Modèle relationnel

- Projection : $T = \pi \langle A, B, C \rangle (R)$
ou $T = \text{PROJECT } (R / A, B, C)$

T ne contient que les attributs A, B et C de R.
ex. Noms et prénoms des étudiant·es.

Notation graphique :



Opérateurs spécifiques (2/3)

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7723951	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4898223	O'Reilly Press	77 Boston Ave, Cambridge
03-2020888	Ch. Light Books	88 Market, San Francisco

AuthorID	Author Name	AuthorCity
345-20-2820	Halle Delisse	14-Aug-92
302-48-8982	Joel Elbow	14-Mar-92
456-25-4012	Sally Hammings	12-Sep-70
150-50-7254	Ed Harcourt-Alexand	12-Mar-00

Cross	AuthorID	PubID	Event	Title
1-34532-482-1	345-20-2820	03-4472822	1993	Cold Fusion for Dummies
1-38482-898-1	392-88-9985	04-7723951	1995	Muram and Elbow Ties
2-39821-489-4	456-22-4012	03-4898223	1992	Fluid Dynamics of Aqueducts
1-38278-283-4	665-08-1254	03-3920888	1987	Elbows, Buckets & Revolutions

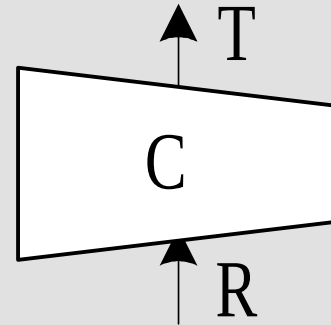
Modèle relationnel

- Restriction : $T = \sigma \langle C \rangle (R)$
ou $T = \text{RESTRICT} (R / C)$

T ne contient que les attributs de R qui satisfont la condition C.

ex. C = Étudiant·es qui habitent à Lyon.

Notation graphique :



Opérateurs spécifiques (3/3)

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7726903	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4898223	O'Reilly Press	77 Boston Ave, Cambridge
02-2020888	Ch. Light Books	88 Market, San Francisco

AuthorID	Author name	AuthorAddress
345-20-2820	Halle Delisse	14-Aug-92
302-48-8982	Joel Elbow	14-Mar-92
456-25-4012	Sally Hammings	12-Sep-70
1503-50-7254	Harrold Arentz	12-Mar-00

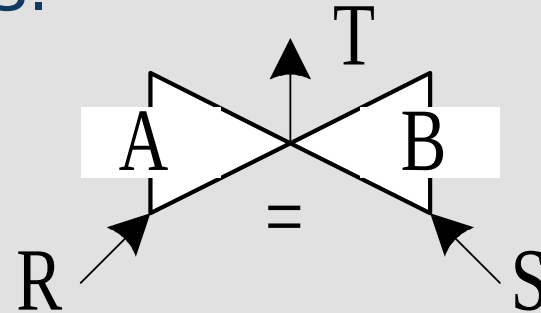
Title	Year	PubID	AuthorID	Price
1-34432-482-1	145-20-2910	02-4472822	1980	Cold Fusion for Dummies
1-3882-895-1	192-80-8985	04-7726903	1985	Murama and Elbow Time
2-39921-489-4	456-22-4012	03-4898223	1982	Fluid Dynamics of Aquabuts
1-38278-203-4	665-08-1254	02-2020888	1987	Evans, Baskets & Revolution

Modèle relationnel

- ▶ Jointure naturelle : $T = R \bowtie S$
ou $T = \text{JOIN} (R, S)$

Produit cartésien $R \times S$ et restriction $A = B$ sur les attributs $A \in R$ et $B \in S$.

Notation graphique :



Exemple de requête (1/4)

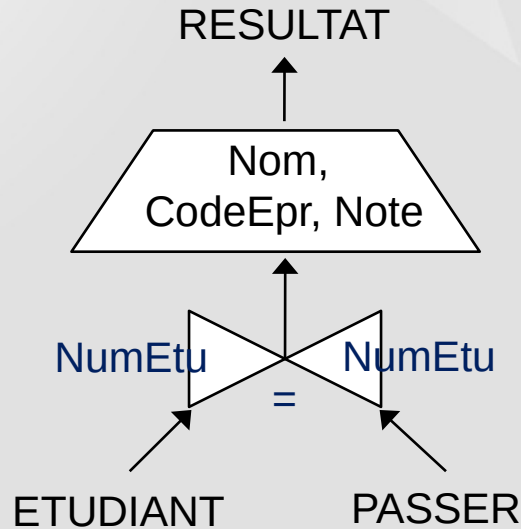
PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7723951	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4698223	O'Reilly Press	77 Boston Ave, Cambridge
03-2020888	Ch. Light Books	69 Market, San Francisco

AuthorID	Author Name	Publication
345-20-2820	Halle Delassie	14-Aug-92
302-49-8962	Joan Elbow	14-Mar-95
456-25-4012	Sally Hemmings	15-Sep-70
1503-50-1254	Hannah Arendt	13-Mar-00

PubID	AuthorID	PubID	PubID	Title
1-34432-492-1	345-20-2820	03-4472822	1980	Cold Fusion for Dummies
1-3982-895-1	392-88-9985	04-7723951	1985	Murphy and Elbow Ties
2-35931-499-4	456-22-4012	03-4698223	1982	Fluid Dynamics of Aqueducts
1-38278-203-4	665-08-1254	03-3920888	1987	Evans, Baskett & Revolution

Modèle relationnel

Notes des étudiant-es en précisant leurs noms
(et pas seulement leurs numéros)



Exemple de requête (2/4)

PubID	Publisher	PubAddress
03-4472522	Random House	123 4th Street, New York
04-7723951	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4892223	O'Reilly Press	77 Boston Ave, Cambridge
03-2920888	Ch. Light Books	88 Market, San Francisco

AuthorID	AuthorName	AuthorCity
345-20-2920	Halle Delasse	14-Aug-92
302-49-8965	Jon Blow	14-Mar-95
456-25-4012	Sally Hemmings	13-Sep-70
150-55-7254	Harrold Arentz	13-Mar-00

Book	AuthorID	PubID	Price	Title
1-34513-492-1	345-20-2920	03-4472522	1990	Cold Fusion for Dummies
1-3882-895-1	392-88-9985	04-7723951	1895	Mathematics and Blow-Tung
2-39921-499-4	456-25-4012	03-4892223	1982	Fluid Dynamics of Aqueducts
1-38278-293-4	665-08-1224	03-3920888	1987	Ewads, Baskets & Revolution

Modèle relationnel

Décomposition des opérations

ETUDIANT

<u>NumEtu</u>	<u>Nom</u>
101	E1
102	E2
103	E3

X

PASSER

<u>NumEtu</u>	<u>CodeEpr</u>	<u>Note</u>
101	INFO1	10
103	INFO1	15
103	ECO1	12

Exemple de requête (3/4)

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7723951	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4898223	O'Reilly Press	77 Boston Ave, Cambridge
03-2020888	Ch. Light Books	89 Market, San Francisco

AuthorID	Author Name	AuthorCity
345-20-2920	Halle Gellesse	14-Aug-92
302-49-8962	John Elbow	14-Mar-95
456-25-4012	Sally Hemmings	12-Sep-70
1003-950-7254	Harrold Averett	12-Mar-00

BookID	AuthorID	PubID	Book Title
1-34432-492-1	345-20-2920	03-4472822	1980 Cold Fusion for Dummies
1-3882-895-1	392-88-9985	04-7723951	1985 Molecules and How They
2-39921-499-4	456-25-4012	03-4898223	1992 Fluid Dynamics of Aqueducts
1-98278-203-4	665-08-1254	03-3920888	1987 Ewads, Baskets & Revolution

Modèle relationnel

==

<u>E.NumEtu</u>	<u>Nom</u>	<u>P.NumEtu</u>	<u>CodeEpr</u>	<u>Note</u>
101	E1	101	INFO1	10
102	E2	101	INFO1	10
103	E3	101	INFO1	10
101	E1	103	INFO1	15
102	E2	103	INFO1	15
103	E3	103	INFO1	15
101	E1	103	ECO1	12
102	E2	103	ECO1	12
103	E3	103	ECO1	12

Exemple de requête (4/4)

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-772091	Wiley and Sons	45 Lincoln Blvd, Chicago
03-489223	O'Reilly Press	77 Boston Ave, Cambridge
03-202088	Ch. Light Books	88 Market, San Francisco

AuthorID	Author Name	AuthorCity
345-20-2920	Halle Delisse	14-Aug-92
302-49-8960	Jon Elow	14-Mar-92
454-25-4012	Sally Hemmings	12-Sep-70
1503-50-7250	Heleno Azeved	12-Mar-00

Book	AuthorID	PubID	Price	Title
1-34513-401-1	345-20-2920	03-4472822	19.95	Cold Fusion for Dummies
1-3882-895-1	302-49-8960	04-772091	19.95	Macroe and Elow Temp
2-39921-499-4	454-25-4012	03-489223	19.92	Fluid Dynamics of Aqueducts
1-39278-203-4	665-08-1254	03-392088	19.97	Evans, Buckets & Revolution

ETUDIANT \triangleright \triangleleft PASSER

<u>E.NumEtu</u>	<u>Nom</u>	<u>P.NumEtu</u>	<u>CodeEpr</u>	<u>Note</u>
101	E1	101	INFO1	10
103	E3	103	INFO1	15
103	E3	103	ECO1	12

π \langle Nom, CodeEpr, Note \rangle (ETUDIANT \triangleright \triangleleft PASSER)

<u>Nom</u>	<u>CodeEpr</u>	<u>Note</u>
E1	INFO1	10
E3	INFO1	15
E3	ECO1	12

(Projection sur les attributs
Nom, CodeEpr et Note)

Classification des SGBD relationnels

- ▶ Niveau 1 : Systèmes non relationnels.
Supportent uniquement la structure tabulaire.
- ▶ Niveau 2 : Systèmes relationnellement minimaux.
Permettent les opérations de restriction, projection et jointure.
- ▶ Niveau 3 : Systèmes relationnellement complets.
Toutes les opérations de l'algèbre relationnelle.
- ▶ Niveau 4 : Systèmes relationnellement pleins.
Permettent la définition des contraintes d'intégrité.

Qu'est-ce que SQL ?

- ▶ Structured Query Language
- ▶ Issu de SEQUEL (Structured English as a Query Language)
- ▶ Permet la **LDD** définition, la **LMD** manipulation et le **LCD** contrôle d'une base de données relationnelle.
- ▶ SQL se base sur l'algèbre relationnelle.
- ▶ **Standard** depuis 1986.

- ▶ `NUMBER(n)` : nombre entier à n chiffres
- ▶ `NUMBER(n, m)` : nombre réel à n chiffres au total (virgule comprise) et m chiffres après la virgule
- ▶ `VARCHAR(n)` : chaîne de caractères de taille n
- ▶ `DATE` : date au format 'JJ-MM-AAAA'

- ▶ Mot clé **CONSTRAINT**
- ▶ Identification par un nom de contrainte
- ▶ Clé primaire :
PRIMARY KEY (clé)
- ▶ Clé étrangère :
FOREIGN KEY (clé) REFERENCES table(attribut)
- ▶ Contrainte de domaine :
CHECK (condition)

ex.

```
CREATE TABLE Etudiant (  
    NumEtu NUMBER(8),  
    Nom VARCHAR(255),  
    Prenom VARCHAR(255),  
    DateNaiss DATE,  
    Rue VARCHAR(255),  
    CP NUMBER(5),  
    Ville VARCHAR(255),  
  
    CONSTRAINT EtuClePri PRIMARY KEY (NumEtu) )
```

Définition des données (2/2)

ex.

```
CREATE TABLE Passer ( NumEtu NUMBER(8),  
                      CodeEpr VARCHAR(10),  
                      Note NUMBER(5, 2),  
  
CONSTRAINT PassClePri PRIMARY KEY (NumEtu, CodeEpr),  
  
CONSTRAINT PassCleEtrEtu FOREIGN KEY (NumEtu)  
REFERENCES Etudiant (NumEtu),  
  
CONSTRAINT PassCleEtrEpr FOREIGN KEY (CodeEpr)  
REFERENCES Epreuve (CodeEpr),  
  
CONSTRAINT NoteValide CHECK (Note >= 0 AND Note <= 20) )
```

▶ Ajout d'attributs

ALTER TABLE nom_table ADD (attribut TYPE, ...)

ex. ALTER TABLE Etudiant ADD (tel NUMBER(8))

▶ Modifications d'attributs

ALTER TABLE nom_table MODIFY (attribut TYPE, ...)

ex. ALTER TABLE Etudiant MODIFY (tel NUMBER(10))

▶ Suppression d'attributs

ALTER TABLE nom_table DROP COLUMN attribut, ...

ex. ALTER TABLE Etudiant DROP COLUMN tel

▶ Ajout de contrainte

```
ALTER TABLE nom_table  
ADD CONSTRAINT nom_contrainte définition_contrainte
```

ex. ALTER TABLE Epreuve
ADD CONSTRAINT LieuValide CHECK (Lieu IN ('Say', 'Aubrac'))

▶ Suppression de contrainte

```
ALTER TABLE nom_table DROP CONSTRAINT nom_contrainte
```

ex. ALTER TABLE Epreuve
DROP CONSTRAINT LieuValide



- ▶ **Définition** : Structure de données physique permettant d'accélérer les accès aux données
- ▶ **Exemple** : `CREATE INDEX IdxNomEtu ON Etudiant (Nom)`
- ▶ **NB** : La clé primaire d'une relation est automatiquement indexée.

- ▶ **Définition** : Une vue est une table **virtuelle** calculée à partir d'autres tables grâce à une requête.
- ▶ **Création d'une vue**

```
CREATE VIEW nom_vue AS requête
```

```
ex. CREATE VIEW lesNoms AS  
    SELECT Nom, Prenom FROM Etudiant
```

- ▶ Simplification de l'accès aux données en masquant les opérations de jointure

ex. `CREATE VIEW notesParEtudiant AS
SELECT E.NumEtu, Nom, Prenom, NumEpr, Note
FROM Etudiant E, Passer P
WHERE E.NumEtu = P.NumEtu`

`SELECT NumEtu, Nom FROM notesParEtudiant
WHERE Note > 10`

- ▶ Sauvegarde indirecte de requêtes complexes
- ▶ Présentation de mêmes données sous différentes formes adaptées aux différents usagers particuliers
- ▶ Support de l'indépendance logique
ex. Si la table Etudiant est remaniée, la vue notesParEtudiant doit être refaite, mais les requêtes qui utilisent cette vue n'ont pas à être remaniées.
- ▶ Renforcement de la sécurité des données par masquage des lignes et des colonnes sensibles aux usagers non habilités

- ▶ Le mot clé DISTINCT doit être **absent** de la requête.
- ▶ La clause FROM doit faire référence à **une seule table**.
- ▶ La clause SELECT doit faire référence directement aux attributs de la table concernée (**pas d'attribut dérivé**).
- ▶ Les clauses GROUP BY et HAVING sont **interdites**.

- ▶ ALL_TABLES (OWNER, TABLE_NAME, ...)
- ▶ ALL_VIEWS (OWNER, VIEW_NAME, ...)
- ▶ ALL_CONSTRAINTS (OWNER, TABLE_NAME, CONSTRAINT_NAME, CONSTRAINT_TYPE, SEARCH_CONDITION, ...)
- ▶ ALL_CONS_COLUMNS (OWNER, TABLE_NAME, CONSTRAINT_NAME, COLUMN_NAME, ...)
- ▶ USER_CATALOG (TABLE_NAME, TABLE_TYPE)
- ▶ USER_TAB_COLUMNS (TABLE_NAME, COLUMN_NAME, ...)
- ▶ USER_IND_COLUMNS (INDEX_NAME, TABLE_NAME, COLUMN_NAME, ...)
- ▶ USER_CONSTRAINTS (TABLE_NAME, CONSTRAINT_NAME, CONSTRAINT_TYPE, SEARCH_CONDITION, ...)

- ▶ Tables qui contiennent un attribut *Intitulé*

```
SELECT TABLE_NAME FROM USER_TAB_COLUMNS  
WHERE COLUMN_NAME = 'INTITULÉ'
```

- ▶ Attributs de la table *Client*

```
SELECT COLUMN_NAME FROM USER_TAB_COLUMNS  
WHERE TABLE_NAME = 'CLIENT'
```

- ▶ Contraintes des tables de l'utilisateur courant

```
SELECT TABLE_NAME, CONSTRAINT_NAME  
FROM ALL_CONSTRAINTS  
WHERE OWNER = USER
```

- ▶ Ajout d'un n-uplet

ex. `INSERT INTO Matiere
VALUES ('BDM1MBFA', 'Bases de données')`

- ▶ Modification de la valeur d'un attribut

ex. `UPDATE Etudiant SET Nom='Tatiana'
WHERE NumEtu = 333333`

ex. `UPDATE Passer SET Note = Note + 1`

- ▶ Suppression de n-uplets

ex. `DELETE FROM Etudiant
WHERE Ville = 'Lyon'`

ex. `DELETE FROM Epreuve`

▶ Par l'exemple, sur la base ETUDIANTS

CARTE_IZLY (NumCarte, SoldeCROUS)

GROUPE_TD (CodeGroupe)

ETUDIANT (NumEtu, Nom, Prénom, DateNaiss, Rue, CP, Ville,
NumCarte#, CodeGroupe#)

MATIERE (CodeMat, Intitulé)

EPREUVE (CodeEpr, DateEpr, Lieu, CodeMat#)

PASSER (NumEtu#, CodeEpr#, Note)

Note : Les symboles [] indiquent une clause optionnelle d'une requête dans les transparents suivants.

- ▶ Tous les n-uplets d'une table : étoile ()
ex. `SELECT * FROM Etudiant`
- ▶ Tri du résultat
ex. Par ordre alphabétique [inverse] de nom
`SELECT * FROM Etudiant
ORDER BY Nom [DESC]`
- ▶ Champs calculés
ex. Transformation de notes sur 20 en notes sur 10
`SELECT Note / 2 FROM Passer`

▶ Projection

ex. Noms et Prénoms des étudiant·es, uniquement (pas les autres attributs)

```
SELECT Nom, Prénom FROM Etudiant
```

▶ Suppression des doublons

ex. SELECT DISTINCT Nom FROM Etudiant

▶ Restriction

ex. Étudiant·es qui habitent à Lyon

```
SELECT * FROM Etudiant  
WHERE Ville = 'Lyon'
```

ex. Épreuves se déroulant après le 01/01/2016

```
SELECT * FROM Epreuve  
WHERE DateEpr >= '01-01-2016'
```

ex. Notes comprises entre 10 et 20

```
SELECT * FROM Passer  
WHERE Note BETWEEN 10 AND 20
```

ex. Notes indéterminées (sans valeur)

```
SELECT * FROM Passer  
WHERE Note IS NULL
```

ex. Étudiant-es habitant une ville dont le nom se termine par sur-Saône

```
SELECT * FROM Etudiant  
WHERE Ville LIKE '%sur-Saône'
```

'sur-Saône%' ⇒ commence par *sur-Saône*

'%sur%' ⇒ contient le mot *sur*

ex. Prénoms des étudiant·es dont le nom est Dupont, Durand ou Martin

```
SELECT Prénom FROM Etudiant  
WHERE Nom IN ('Dupont', 'Durand', 'Martin')
```

NB : Possibilité d'utiliser la négation pour tous ces prédicats
⇒ **NOT BETWEEN**, **NOT NULL**, **NOT LIKE**, **NOT IN**.

- ▶ **ET.** ex. Épreuves se déroulant le 15/01/2016 en salle D201
SELECT * FROM Epreuve
WHERE DateEpr = '15-01-2016' **AND** Lieu = 'D201'
- ▶ **OU.** ex. Étudiant·es né·es avant 1990 ou habitant hors Lyon
SELECT * FROM Etudiant
WHERE DateNaiss < '01-01-1990' **OR** Ville <> 'Lyon'
- ▶ **Combinaisons.** ex. Étudiant·es né·es après 1990 et habitant Lyon ou Vienne
SELECT * FROM Etudiant
WHERE DateNaiss > '31-12-1990'
AND (Ville = 'Lyon' **OR** Ville = 'Vienne')

- ▶ Elles opèrent sur un **ensemble de valeurs** et les agrègent.
- ▶ **AVG()**, **VARIANCE()**, **STDDEV()** : moyenne, variance et écart-type des valeurs
- ▶ **SUM()** : somme des valeurs
- ▶ **MIN()**, **MAX()** : valeur minimum, valeur maximum
- ▶ **COUNT()** : nombre de valeurs
- ▶ **ex. Moyenne des notes**

```
SELECT AVG(Note) FROM Passer
```

ex. Nombre total de notes

```
SELECT COUNT(*) FROM Passer  
SELECT COUNT(NumEtu) FROM Passer
```

ex. Nombre d'étudiant·es noté·es

```
SELECT COUNT(DISTINCT NumEtu) FROM Passer
```


Table PASSER

<u>NumEtu</u>	<u>CodeEpr</u>	<u>Note</u>
101	INFO1	10
103	INFO1	15
103	ECO1	12

COUNT(NumEtu) \Rightarrow Résultat = 3

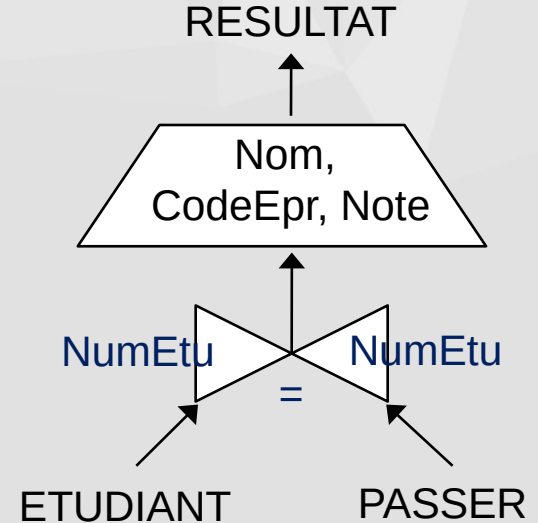
COUNT(DISTINCT NumEtu) \Rightarrow Résultat = 2

ex. Liste des notes avec le nom des étudiant-es

```
SELECT Nom, CodeEpr, Note
```

```
FROM Etudiant, Passer
```

```
WHERE Etudiant.NumEtu = Passer.NumEtu
```



ex. Idem avec le numéro d'étudiant en plus

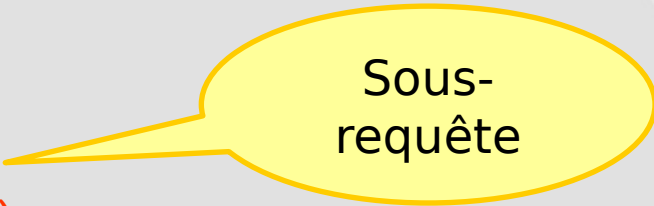
```
SELECT E.NumEtu, Nom, CodeEpr, Note  
FROM Etudiant E, Passer P  
WHERE E.NumEtu = P.NumEtu  
ORDER BY Nom, Note DESC
```

NB : Utilisation d'**alias** (E et P) pour alléger l'écriture
+ tri par nom (croissant) et note (décroissante).

Jointure exprimée avec le prédicat IN

ex. Notes des épreuves passées le 23 septembre 2016

```
SELECT Note FROM Passer
WHERE CodeEpr IN (
    SELECT CodeEpr FROM Epreuve
    WHERE DateEpr = '23-09-2016' )
```



Sous-
requête

NB : Il est possible d'imbriquer des requêtes.

▶ Prédicats EXISTS / NOT EXISTS

ex. Étudiant·es qui ont passé au moins une épreuve
[n'ont passé aucune épreuve]

```
SELECT * FROM Etudiant E
WHERE [NOT] EXISTS (
    SELECT * FROM Passer P
    WHERE E.NumEtu = P.NumEtu )
```

▶ Prédicats ALL / ANY

ex. Numéros des étudiant·es qui ont obtenu au moins une note supérieure à chacune [à au moins une] des notes obtenues par l'étudiant·e n° 1000.

```
SELECT DISTINCT NumEtu FROM Passer
WHERE Note > ALL [ANY] (
    SELECT Note FROM Passer
    WHERE NumEtu = 1000 )
```

ex. Moyenne de chaque étudiant·e

```
SELECT NumEtu, AVG(Note)
FROM Passer
GROUP BY NumEtu
```

ex. Nombre de notes par étudiant·e

```
SELECT NumEtu, COUNT(*)
FROM Passer
GROUP BY NumEtu
```

ex. Note moyenne pour les étudiant-es ayant passé moins de 5 épreuves

```
SELECT NumEtu, AVG(Note)
FROM Passer
GROUP BY NumEtu
HAVING COUNT(*) < 5
```

Attention : La clause HAVING ne s'utilise qu'avec GROUP BY.

NB : HAVING : évaluation de condition sur un résultat de groupement (*a posteriori*)

≠ WHERE : évaluation de condition *a priori*

- ▶ **Ex.** Numéro des étudiant·es qui ont passé toutes les épreuves
- ▶ **NB :** Il n'existe **pas** d'opérateur de division en SQL !
- ▶ **Deux stratégies :**
 - Étudiant·es tels qu'il n'existe pas d'épreuve tel qu'il n'existe pas de « passage » pour cet étudiant·e et cette épreuve.
 - Étudiant·es qui ont passé un nombre distinct d'épreuves égal au nombre total d'épreuves.

```
SELECT NumEtu
FROM Etudiant Et
WHERE NOT EXISTS (
    SELECT *
    FROM Epreuve Ep
    WHERE NOT EXISTS (
        SELECT *
        FROM Passer P
        WHERE Et.NumEtu = P.NumEtu
        AND P.CodeEpr = Ep.CodeEpr ) )
```

```
▶ SELECT NumEtu FROM Etudiant E
   WHERE ( SELECT COUNT(CodeEpr)
            FROM Passer P
            WHERE E.NumEtu = P.NumEtu )
   = ( SELECT COUNT(*) FROM Epreuve )
```

ou

```
▶ SELECT NumEtu FROM Passer
   GROUP BY NumEtu
   HAVING COUNT(CodeEpr) =
      ( SELECT COUNT(*) FROM Epreuve )
```

INTERSECT, MINUS, UNION

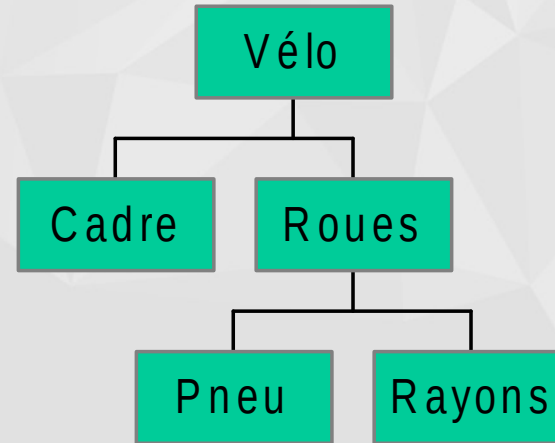
ex. Code des épreuves ayant soit lieu dans l'Amphi Aubrac, soit ayant été passées par l'étudiant-e n° 102

```
SELECT CodeEpr FROM Epreuve  
WHERE Lieu = 'Amphi Aubrac'
```

UNION

```
SELECT CodeEpr FROM Passer  
WHERE NumEtu = 102
```

Exemple de hiérarchie (nomenclature) :



Relation associée :

ELEMENT (No_Elt, Dési, Parent#)

0	Vélo	NULL
1	Cadre	0
2	Roue1	0
3	Roue2	0
4	Pneu1	2
5	Pneu2	3

6	Rayon11	2
7	Rayon12	2
8	Rayon13	2
9	Rayon21	3

ex. Structure hiérarchique des éléments à partir de la racine

```
SELECT Dési FROM Element  
CONNECT BY Parent = PRIOR No_Elt  
START WITH Parent IS NULL;
```

Racine de la
hiérarchie

Ordre de parcours de
la hiérarchie

ex. Idem avec indication du niveau dans la hiérarchie

```
SELECT LEVEL, Dési FROM Element  
CONNECT BY Parent = PRIOR No_Elt  
START WITH Parent IS NULL;
```

Résultat :

1 Velo	3 Rayon12
2 Cadre	3 Rayon13
2 Roue1	2 Roue2
3 Pneu1	3 Pneu2
3 Rayon11	3 Rayon21

ex. Idem avec élagage d'une branche de la hiérarchie

```
SELECT LEVEL, Dési FROM Element  
CONNECT BY Parent = PRIOR No_Elt AND Dési <> 'Roue2'  
START WITH Parent IS NULL;
```

Résultat :

1 Velo	3 Rayon12
2 Cadre	3 Rayon13
2 Roue1	2 Roue2
3 Pneu1	
3 Rayon11	

ex. Nombre d'éléments dans chaque niveau

Il est possible d'utiliser le groupement.

```
SELECT LEVEL, COUNT(No_Elt)
FROM Element
CONNECT BY Parent = PRIOR No_Elt
START WITH Parent IS NULL
GROUP BY LEVEL;
```

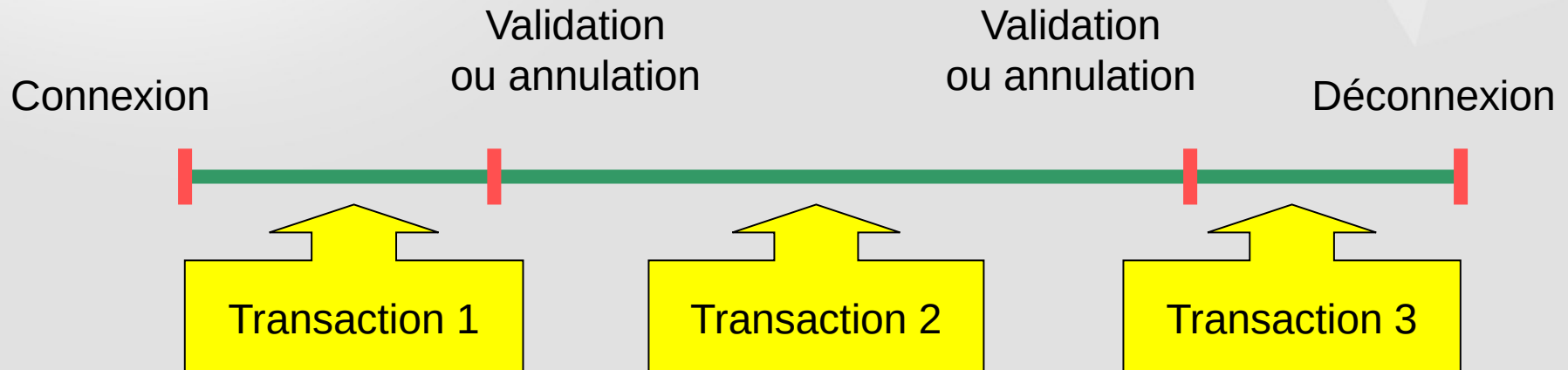
- `ABS(n)` : Valeur absolue de n
- `CEIL(n)` : Plus petit entier $\geq n$
- `FLOOR(n)` : Plus grand entier $\leq n$
- `MOD(m, n)` : Reste de m/n
- `POWER(m, n)` : m^n
- `SIGN(n)` : Signe de n
- `SQRT(n)` : Racine carrée de n
- `ROUND(n, m)` : Arrondi à 10^{-m}
- `TRUNC(n, m)` : Troncature à 10^{-m}
- `CHR(n)` : Caractère ASCII n° n
- `INITCAP(ch)` : 1^{re} lettre en maj.
- `LOWER(ch)` : c en minuscules
- `UPPER(ch)` : c en majuscules
- `LTRIM(ch, n)` : Troncature à gauche
- `RTRIM(ch, n)` : Troncature à droite
- `REPLACE(ch, car)` : Remplacement de caractère
- `SUBSTR(ch, pos, lg)` : Extraction de chaîne
- `SOUNDEX(ch)` : Représentation phonétique de ch
- `LPAD(ch, lg, car)` : Compléter à gauche
- `RPAD(ch, lg, car)` : Compléter à droite

- `ASCII(ch)` : Valeur ASCII de ch
- `INSTR(ch, ssch)` : Recherche de ssch dans ch
- `LENGTH(ch)` : Longueur de ch
- `ADD_MONTHS(dte, n)` : Ajout de n mois à dte
- `LAST_DAY(dte)` : Dernier jour du mois
- `MONTHS_BETWEEN(dt1, dt2)` : Nombre de mois entre dt1 et dt2
- `NEXT_DAY(dte)` : Date du lendemain
- `SYSDATE` : Date/heure système
- `TO_NUMBER(ch)` : Conversion de ch en nombre
- `TO_CHAR(x)` : Conversion de x en chaîne
- `TO_DATE(ch)` : Conversion de ch en date
- `NVL(x, val)` : Remplace par val si x a la valeur NULL
- `GREATEST(n1, n2...)` : + grand
- `LEAST (n1, n2...)` : + petit
- `UID` : Identifiant numérique de l'utilisateur
- `USER` : Nom de l'utilisateur



- ▶ `SELECT UID, USER FROM DUAL;`
- ▶ `SELECT GREATEST(1, 2, 3) FROM DUAL;`
- ▶ `SELECT Nom, Prenom,
FLOOR(MONTHS_BETWEEN(SYSDATE, DateNaiss) / 12) Age
FROM Etudiant;`
- ▶ `UPDATE Passer SET Note = NVL(Note, 10);`

- ▶ **Transaction** : ensemble de mises à jour des données (⇒ modifications structurelles)
- ▶ **Début de transaction** : début de la session de travail ou fin de la transaction précédente



► Validation (et fin) d'une transaction :

COMMIT

► Annulation (et fin) d'une transaction :

ROLLBACK

▶ Création

- ex. `CREATE USER moi_meme IDENTIFIED BY mon_mot_de_passe`

▶ Suppression

- ex. `DROP USER moi_meme CASCADE`

▶ Modification

- ex. `ALTER USER moi_meme IDENTIFIED BY aaaaa`

- ▶ Droit d'effectuer une action sur les objets de l'utilisateur seulement
 - ex. CREATE TABLE
ALTER INDEX
DROP VIEW
- ▶ Droit d'effectuer une action dans tous les schémas de la base de données
 - ex. CREATE ANY TABLE
ALTER ANY INDEX
DROP ANY VIEW

Privilège	Signification	Tables	Vues
ALTER	Destruction	X	
DELETE	Suppression	X	X
INDEX	Construction	X	
INSERT	Insertion	X	X
REFERENCES	Clé étrangère	X	
SELECT	Lecture	X	X
UPDATE	Mise à jour	X	X
ALL	Tous	X	X

▶ Rôles prédéfinis

- **CONNECT** : droit de création de tables, vues, synonymes, etc.
- **RESOURCE** : droit de création de procédures stockées, déclencheurs, etc.
- **DBA** : administrateur de la BD

▶ Création de nouveaux rôles

- ex. **CREATE ROLE role1**

▶ Transmission de privilèges

```
GRANT privilège ON table|vue  
TO user|PUBLIC [WITH GRANT OPTION]
```

▶ Privilèges sur des objets

- ex. GRANT SELECT ON ma_table TO toto
- ex. GRANT SELECT ON ma_table TO PUBLIC
- ex. GRANT SELECT ON ma_table TO role1

▶ Privilèges globaux et rôles

- ex. GRANT CREATE ANY TABLE TO toto
- ex. GRANT CONNECT, RESOURCE TO toto
- ex. GRANT role1 TO toto

▶ Suppression de privilèges

```
REVOKE privilège ON table|vue FROM user|PUBLIC
```

▶ Privilèges sur des objets

- ex. REVOKE SELECT ON ma_table FROM toto
- ex. REVOKE SELECT ON ma_table FROM PUBLIC
- ex. REVOKE SELECT ON ma_table FROM role1

▶ Privilèges globaux et rôles

- ex. REVOKE CREATE ANY TABLE FROM toto
- ex. REVOKE CONNECT, RESOURCE FROM toto
- ex. REVOKE role1 FROM toto

Pour approfondir SQL en ligne...



<https://eric.univ-lyon2.fr/jdarmont/tutoriel-sql/>



Bob Watkins
a aimé votre Tweet.



Jérôme Darmont @darmont_lyon2

@OracleDatabase
PL/SQL

These are words
That go together well
#PLSQLHaiku

Partie 4

Programmation de bases de données

Requêtes SQL dans un programme

- ▶ **SQL encapsulé** : Requêtes SQL incorporées dans le code source (PL/SQL, T-SQL, PL/pgSQL, Pro*C...)
- ▶ **API** : Requêtes SQL via des fonctions du langage (Java Persistence API, PHP Data Objects...)
- ▶ **Interfaces de niveau appel** : intergiciel entre le langage et le SGBD (ODBC, JDBC, ADO...)
- ▶ **Procédures stockées** : Fonctions SQL stockées dans la base de données et exécutées par le SGBD (écrites en PL/SQL, T-SQL, PL/pgSQL)

C
U
R
S
E
U
R
S

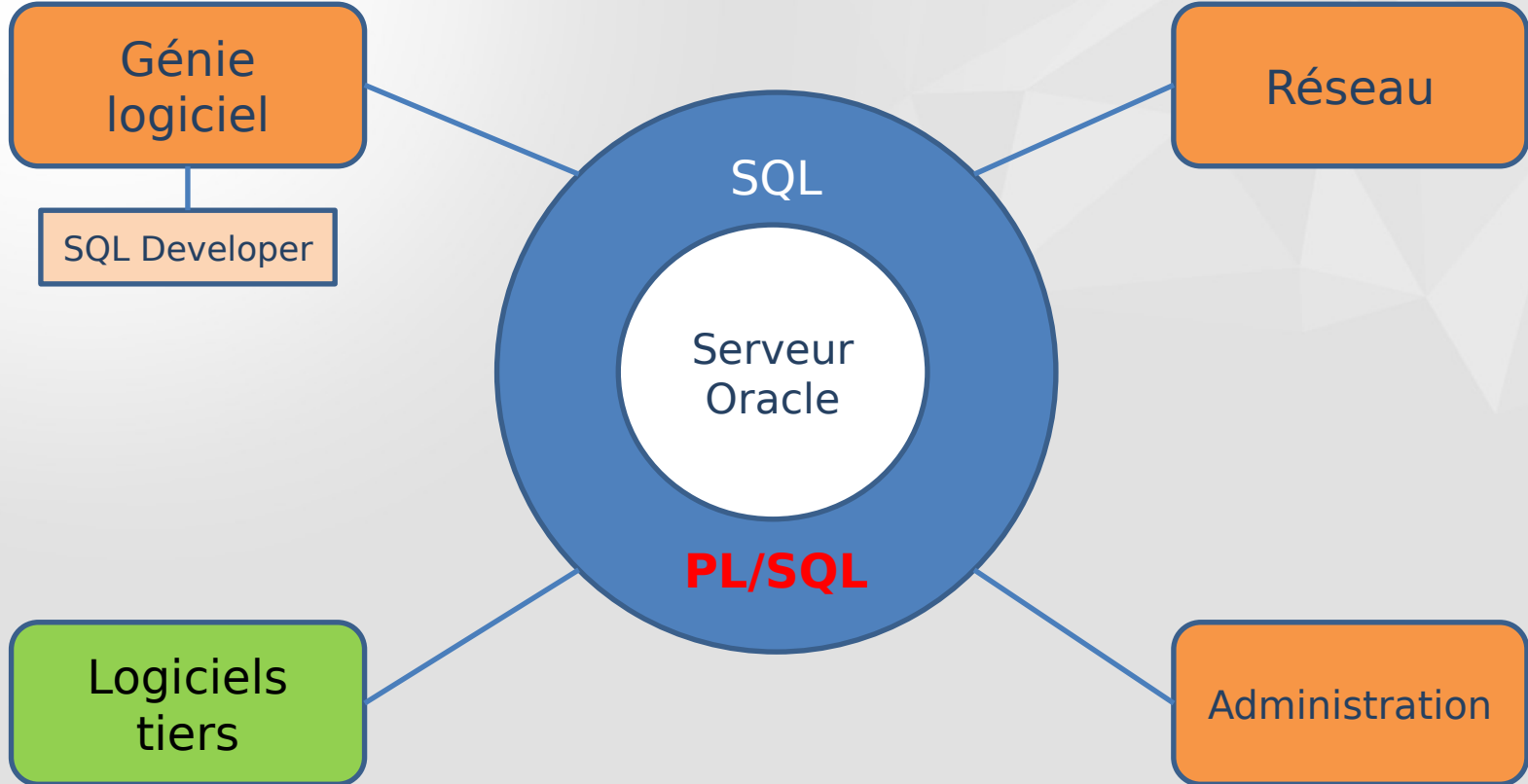
Caractéristiques du langage PL/SQL (1/2)

- ▶ Langage de 4^e génération (L4G = L3G + syntaxe type SQL)
- ▶ Conçu comme une extension de SQL
- ▶ Déclaration de variables et de constantes
- ▶ Types abstraits (collections, enregistrements, objets)
- ▶ Modularité (sous-programmes, paquetages)
- ▶ Gestion des erreurs (Gestion des erreurs)
- ▶ Interaction étroite avec Oracle/SQL (types identiques)

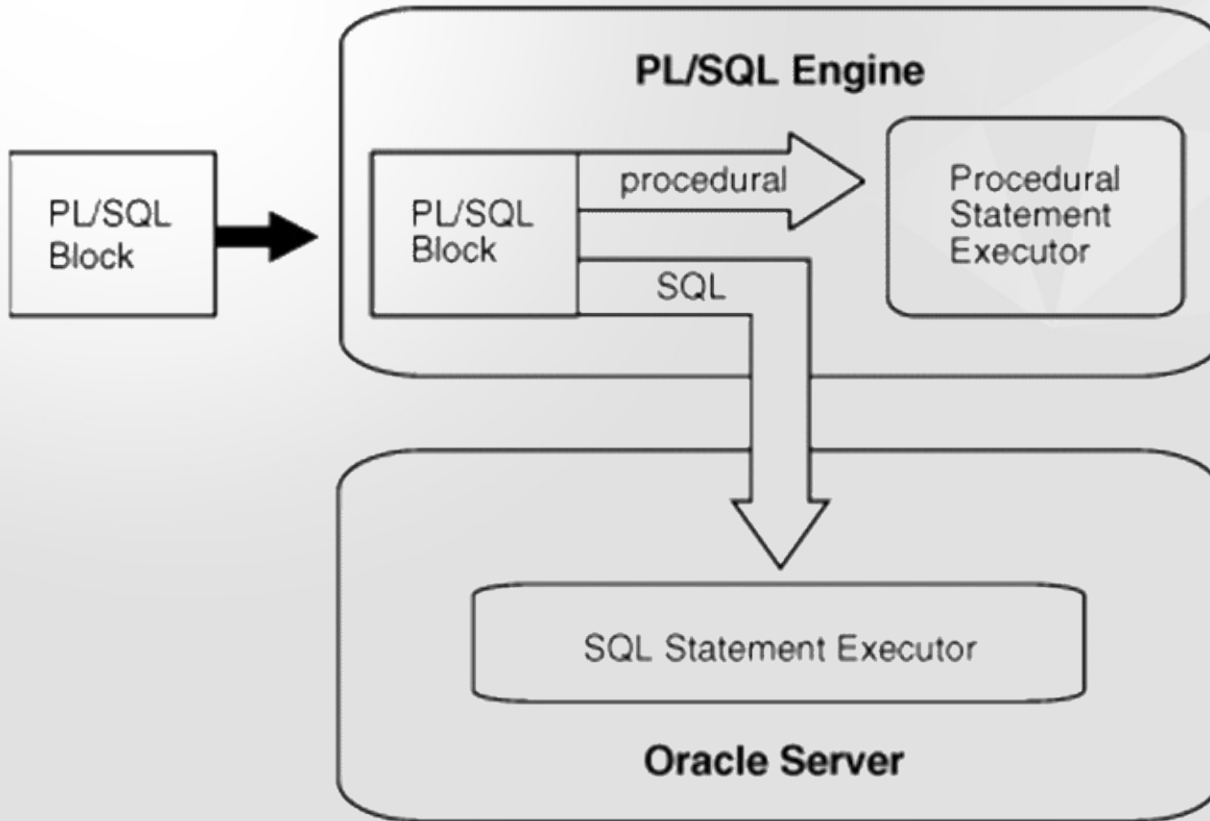
Caractéristiques du langage PL/SQL (2/2)

- ▶ SQL dynamique (construction de requêtes à la volée)
- ▶ Programmation orientée objet
- ▶ Performance (traitement par lots)
- ▶ Productivité (uniformité des outils Oracle)
- ▶ Portabilité (sur tous systèmes Oracle)
- ▶ Sécurité (procédures stockées, déclencheurs)

Architecture d'Oracle



Moteur Oracle



Oracle Database PL/SQL
User's Guide and
Reference

▶ Bloc anonyme

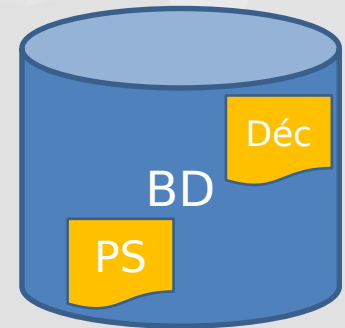
- Stocké dans un fichier
- Compilé et exécuté à la volée

▶ Procédure stockée

- Compilée *a priori*
- Stockée dans la base de données

▶ Déclencheur

- Procédure stockée associée à une table
- Exécution automatique à la suite d'un événement



```
[DECLARE  
  -- Types, constantes et variables]  
BEGIN  
  -- Instructions PL/SQL  
[EXCEPTION  
  -- Gestion des erreurs]  
END;
```

▶ Déclaration dans la section **DECLARE** d'un bloc PL/SQL

▶ Variables

```
ex.  date_naissance DATE;
      compteur INTEGER := 0;           -- Initialisation
      compteur2 INTEGER DEFAULT 0; -- Valeur par
défaut
      id CHAR(5) NOT NULL := 'AP001';
```

▶ Constantes

```
ex.  taux_tva CONSTANT REAL := 0.2;
```

Scalar Types

BINARY_DOUBLE
BINARY_FLOAT
BINARY_INTEGER
DEC
DECIMAL
DOUBLE PRECISION
FLOAT
INT
INTEGER
NATURAL
NATURALN
NUMBER
NUMERIC
PLS_INTEGER
POSITIVE
POSITIVEN
REAL
SIGNTYPE
SMALLINT

CHAR
CHARACTER
LONG
LONG RAW
NCHAR
NVARCHAR2
RAW
ROWID
STRING
UROWID
VARCHAR
VARCHAR2

BOOLEAN

DATE

Composite Types

RECORD
TABLE
VARRAY

Reference Types

REF CURSOR
REF object_type

LOB Types

BFILE
BLOB
CLOB
NCLOB

Oracle Database PL/SQL
User's Guide and
Reference

- ▶ Type d'une autre variable
ex. credit REAL;
 debit credit%TYPE;
- ▶ Type de l'attribut d'une table
ex. num_emp EMP.EMPNO%TYPE;
- ▶ Type des n-uplets d'une table
ex. un_etudiant STUDENT%ROWTYPE;

**À utiliser
au maximum !**

▶ Affectation simple

```
ex.   n := 0;  
      n := n + 1;
```

▶ Valeur de la base de données

```
ex.   SELECT custname INTO nom client  
      FROM customer WHERE custnum = 10;
```

```
      SELECT ename, sal INTO nom, salaire  
      FROM emp WHERE empno = 5000;
```

▶ Opérateurs arithmétiques

+ - / * **

▶ Opérateur de concaténation

||

▶ Opérateurs de comparaison

= < > <= >= <>
IS NULL LIKE BETWEEN IN

▶ Opérateurs logiques

AND OR NOT

IF-THEN, IF-THEN-ELSE ou IF-THEN-ELSIF

```
IF condition1 THEN
  -- Statements
[ELSIF condition2 THEN
  -- Instructions PL/SQL]
[ELSE
  -- Instructions PL/SQL]
END IF;
```

CASE

```
CASE variable
  WHEN val1 THEN -- Instruction PL/SQL
  WHEN val2 THEN -- Instruction PL/SQL
  WHEN val3 THEN -- Instruction PL/SQL
  [ELSE          -- Instruction par défaut]
END CASE;
```

► Pour

```
FOR iterateur IN [REVERSE] min..max LOOP
    -- Instructions PL/SQL
END LOOP;
```

► Tant que

```
WHILE condition LOOP
    -- Instructions PL/SQL
END LOOP;
```

► Répéter

```
LOOP
    -- Instructions PL/SQL
    EXIT WHEN condition;
END LOOP;
```

```
DBMS_OUTPUT.PUT('chaîne');           /* Pas de retour à la ligne */
DBMS_OUTPUT.PUT_LINE('chaîne');       /* Retour à la ligne */

DBMS_OUTPUT.PUT('Hello world !');
DBMS_OUTPUT.PUT_LINE('nom = ' || nom);
DBMS_OUTPUT.PUT_LINE('n = ' || TO_CHAR(n));
DBMS_OUTPUT.PUT_LINE('n = ' || n);
```

NB : Pour que l’affichage fonctionne, il faut mettre la variable d’environnement **SERVEROUTPUT** à **ON**.

SET SERVEROUTPUT ON dans SQL Developer

En cas de dépassement, la taille du tampon d’affichage doit être augmentée.

ex. **DBMS_OUTPUT.ENABLE(10000);**

-- Calcul de prix TTC

DECLARE

```
    tauxTVA CONSTANT REAL := 0.2;  
    prix product.prod_price%TYPE;
```

BEGIN

-- Affectation du prix

```
SELECT prod_price INTO prix FROM product  
    WHERE prod_code = 'Pr345blue';
```

-- Ajout de la TVA

```
prix := prix * (1 + tauxTVA);
```

-- Affichage écran

```
DBMS_OUTPUT.PUT_LINE(prix || ' euros');
```

END;

- ▶ **Définition** : Ensemble ordonné d'éléments de même type. Chaque élément est indexé par sa position dans la collection.
- ▶ Deux types de collections
 - Tableau (**VARRAY**) : taille bornée, dense
 - Liste (**TABLE**) : taille extensible, non-dense

Array of Integers

321	17	99	407	83	622	105	19	67	278
x(1)	x(2)	x(3)	x(4)	x(5)	x(6)	x(7)	x(8)	x(9)	x(10)

Fixed Upper Bound

Nested Table after Deletions

321		99	407		622	105	19		278
x(1)		x(3)	x(4)		x(6)	x(7)	x(8)		x(10)

Unbounded

Oracle Database PL/SQL
User's Guide and
Reference

1. Déclarer un type collection

ex. TYPE T_listeChaines IS TABLE OF VARCHAR(20);
 TYPE T_tableauEntiers IS VARRAY(10) OF INTEGER;

2. Déclarer une collection et l'initialiser

ex. maListe T_listeChaines := T_listeChaines('Aa', 'Bb', 'Cc');
 t T_tableauEntiers := T_tableauEntiers();

NB : Une collection peut être déclarée vide (c'est le cas de t).
Il n'est pas obligatoire d'initialiser tous les éléments d'un tableau.

► Collection entière

```
ex.      DECLARE      TYPE T1 IS TABLE OF INT;
                                TYPE T2 IS TABLE OF INT;
                                et11 T1 := T1(1, 2, 3, 4);
                                et12 T1 := T1(5, 6);
                                et2 T2 := T2();
                                BEGIN      et12 := et11;      -- Légal
                                                et2 := et11;      -- Illégal
                                ...
```

► Élément d'une collection

```
ex. et11(1) := 10;
```

- ▶ Ensemble de méthodes (\approx procédures)
Usage: `nom_collection.nom_methode[(paramètres)]`
- ▶ `EXISTS(i)` renvoie TRUE si le i^e élément existe dans la collection.
- ▶ `COUNT` renvoie le nombre d'éléments dans la collection.
- ▶ `LIMIT` renvoie la taille maximum de la collection (**NULL pour les listes**).
- ▶ `EXTEND(n)` augmente la taille de la collection de `n`.
`EXTEND(1) \Leftrightarrow EXTEND`

- ▶ **TRIM(n)** supprime **n** éléments en fin de collection (la taille de la collection diminue automatiquement).
TRIM ⇔ **TRIM(1)**
- ▶ **DELETE(i)** et **DELETE** suppriment respectivement le **i^e** élément et tous les éléments de la collection (**listes seulement**).
- ▶ **FIRST** et **LAST** renvoient respectivement l'index du premier et du dernier élément de la collection.
NB : **FIRST** = 1 et **LAST** = **COUNT** dans un tableau.
- ▶ **PRIOR(i)** et **NEXT(i)** renvoient respectivement l'index de l'élément précédent et de l'élément suivant du **i^e** élément.

```
DECLARE
    TYPE T_listeEntiers IS TABLE OF INTEGER;
    pile T_listeEntiers := T_listeEntiers();
    element INTEGER;
BEGIN
    -- On empile les valeurs 1 et 11
    pile.EXTEND;
    pile(pile.COUNT) := 1;
    pile.EXTEND;
    pile(pile.COUNT) := 11;
    -- On dépile
    element := pile(pile.COUNT); -- element = 11
    pile.TRIM;                 -- Suppression en haut de pile
```

► **Définition** : Ensemble de données liées stockées dans des **champs**.

1. Déclarer un type enregistrement

```
ex.    TYPE T_etudiant IS RECORD(  
        numetu INTEGER,  
        nom VARCHAR(50),  
        age INTEGER  
    );
```

2. Déclarer un enregistrement

```
ex.    unEtudiant T_etudiant;
```

▶ Référence directe

```
ex. unEtudiant.numetu := 12212478;  
unEtudiant.nom := 'Toto';  
unEtudiant.age := 6;
```

```
unEtudiant := monEtudiant; -- de type T_etudiant
```

▶ Résultat de requête

```
ex. SELECT student_number, student_name, student_age  
    INTO unEtudiant  
    FROM student  
    WHERE student_number = 12212478;
```

```
PROCEDURE nomProcedure (param1, param2...) IS
    -- Déclarations locales (pas de clause DECLARE)
BEGIN
    -- Instructions PL/SQL
[EXCEPTION
    -- Gestion des exceptions]
END;
```



```
FUNCTION nomFonction (param1, param2...)  
RETURN typeValeurRetour IS  
    -- Déclarations locales  
BEGIN  
    -- Instructions PL/SQL  
    RETURN valeurRetour;  
[EXCEPTION  
    -- Gestion des exceptions]  
END;
```

- ▶ **Déclaration** : Tout sous-programme doit être défini avant d'être appelé.

⇒ définition dans la section **DECLARE** d'un bloc PL/SQL

- ▶ **Définition et mode de passage des paramètres**

nom_param [IN | OUT | IN OUT] TYPE

ex. resultat OUT REAL

- **IN**: Paramètre d'entrée (lecture seule / par valeur)
- **OUT**: Paramètre de sortie (écriture seule / par référence)
- **IN OUT**: Paramètre d'entrée-sortie (lecture-écriture / par référence)

```
PROCEDURE ConversionUSD_EUR (prixUSD IN REAL,  
                             prixEUR OUT REAL) IS  
  
    taux CONSTANT REAL := 0.89;  
  
BEGIN  
    prixEUR := prixUSD * taux;  
END;
```

Exemple de fonction (récursive)

-- Calcul de n!

```
FUNCTION facto (n INTEGER) RETURN INTEGER IS

BEGIN
  IF n = 1 THEN -- Condition d'arrêt
    RETURN 1;
  ELSE
    RETURN n * facto(n - 1); -- Appel récursif
  END IF;
END;
```

-- Exemple

```
DECLARE
```

```
    hundredBucks CONSTANT REAL := 100;
```

```
    resultEuro REAL;
```

```
    fact10 INTEGER;
```

```
BEGIN
```

```
    ConversionUSD_EUR(hundredBucks, resultEuro);
```

```
    fact10 := facto(10);
```

```
END;
```

- ▶ **Définition** : Structure de données qui stocke le résultat d'une requête retournant plusieurs n-uplets.
- ▶ **Déclaration** : `CURSOR nom_curseur IS requete_SQL;`

ex. `CURSOR calc_TVA IS
 SELECT prod_num, price * 1.2 AS prix_TTC
 FROM product;`

NB : Les n-uplets du curseur sont de type `calc_TVA%ROWTYPE`.

```
-- Parcours complet du curseur
```

```
DECLARE
```

```
    CURSOR calcTVA IS
```

```
        SELECT prod_num, price * 1.2 AS prixTTC
```

```
        FROM product;
```

```
    nuplet calcTVA%ROWTYPE;
```

```
BEGIN
```

```
    FOR nuplet IN calcTVA LOOP
```

```
        DBMS_OUTPUT.PUT_LINE(
```

```
            nuplet.prod_num
```

```
            || ' : ' ||
```

```
            nuplet.prixTTC);
```

```
    END LOOP;
```

De très loin
le plus courant !

```
-- Parcours ad hoc du curseur
```

```
DECLARE
```

```
    -- Comme précédemment
```

```
BEGIN
```

```
    OPEN calcTVA;
```

```
    FETCH calcTVA INTO nuplet;
```

```
-- 1re ligne
```

```
    WHILE calcTVA%FOUND LOOP
```

```
        -- Instructions PL/SQL
```

```
        FETCH calcTVA INTO nuplet;
```

```
-- Ligne suivante
```

```
    END LOOP;
```

```
    CLOSE calcTVA;
```

```
END;
```


- ▶ **%NOTFOUND** est égal à FALSE si FETCH renvoie un résultat.
- ▶ **%FOUND** est égal à TRUE si FETCH renvoie un résultat.
- ▶ **%ROWCOUNT** renvoie le nombre de n-uplets lus.
- ▶ **%ISOPEN** est égal à TRUE si le curseur est ouvert.

```
DECLARE
    CURSOR c(s number) IS SELECT ename, sal FROM emp WHERE sal >= s;
    nuplet c%ROWTYPE;

BEGIN
    OPEN c(2500);
    FETCH c INTO nuplet;
    WHILE c%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(nuplet.ename || ' : ' || nuplet.sal);
        FETCH c INTO nuplet;
    END LOOP;
    CLOSE c;
END;
```

- ▶ Quand une erreur survient, une **exception** est **levée** (exécutée).
- ▶ Gestion des erreurs dans des routines séparées du programme principal
- ▶ **Avantages**
 - Gestion systématique des erreurs
 - Gestion groupée des erreurs similaires
 - Lisibilité du code
- ▶ **Fonctions PL/SQL de gestion des erreurs**
 - **SQLCODE** : Code de la dernière exception levée
 - **SQLERRM** : Message d'erreur associé

Libellé erreur	Code erreur	SQLCODE
CURSOR_ALREADY_OPEN	ORA-06511	-6511
DUP_VAL_ON_INDEX	ORA-00001	-1
INVALID_CURSOR	ORA-01001	-1001
INVALID_NUMBER	ORA-01722	-1722
LOGIN_DENIED	ORA-01017	-1017
NO_DATA_FOUND	ORA-01403	-1403
NOT_LOGGED_ON	ORA-01012	-1012
PROGRAM_ERROR	ORA-06501	-6501
STORAGE_ERROR	ORA-06500	-6500
TIMEOUT_ON_RESOURCE	ORA-00051	-51
TOO_MANY_ROWS	ORA-01422	-1422
VALUE_ERROR	ORA-06502	-6502
ZERO_DIVIDE	ORA-01476	-1476

- ▶ **Déclaration** (section **DECLARE**)

```
nomException EXCEPTION;
```

- ▶ **Lever l'exception** (section **BEGIN**)

```
IF condition THEN  
    RAISE nom_exception;  
END IF;
```

- ▶ **Gérer l'exception** (section **EXCEPTION**)

```
WHEN nom_exception THEN -- Instruction(s) PL/SQL ;
```

Exemple d'exception

```
DECLARE
  c INTEGER;
  personne EXCEPTION;
BEGIN
  SELECT COUNT(*) INTO c FROM emp;
  IF c = 0 THEN
    RAISE personne;
  END IF;
EXCEPTION
  WHEN personne THEN
    RAISE_APPLICATION_ERROR(-20501, 'Table vide !');
END; -- Code d'erreur compris entre -20999 et -20001
```

```
DECLARE
  i INTEGER := &saisie;
  e1 EXCEPTION;
  e2 EXCEPTION;
BEGIN
  IF i = 1 THEN
    RAISE e1;
  ELSIF i = 2 THEN
    RAISE e2;
  ELSE
    i := i / 0;
  END IF;
EXCEPTION
  WHEN e1 THEN RAISE_APPLICATION_ERROR(-20001, 'Exception 1');
  WHEN e2 THEN RAISE_APPLICATION_ERROR(-20002, 'Exception 2');
  WHEN OTHERS THEN RAISE_APPLICATION_ERROR(-20999, SQLERRM);
END;
```

▶ **Définition** : Procédures précompilées stockées de manière permanente dans la base de données

▶ **Création**

CREATE PROCEDURE nom_proc (paramètres) AS ...

ex. CREATE PROCEDURE HelloWorld AS
BEGIN
 DBMS_OUTPUT.PUT_LINE('Hello World!');
END;

▶ **Exécution**

ex.

sous SQL Developer

EXECUTE HelloWorld

en PL/SQL

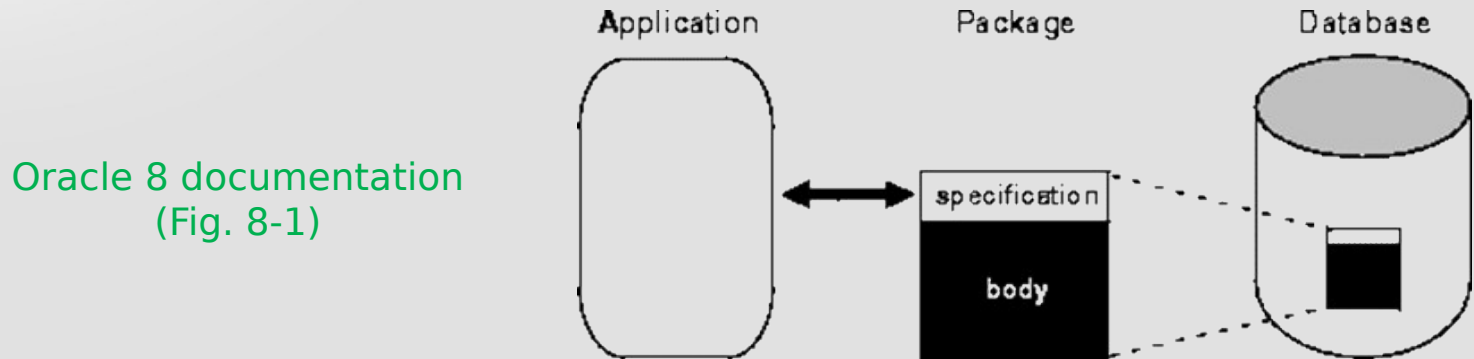
HelloWorld;

▶ **Suppression**

ex.

DROP PROCEDURE HelloWorld;

- ▶ **Définition** : Ensemble de types, curseurs, variables et sous-programmes interreliés et stockés ensemble
- ▶ Un paquetage est subdivisé en deux parties :
 - **Spécification** : interface (déclarations **publiques**),
 - **Corps** : déclarations **privées** et code.



-- Définition de la spécification

```
CREATE [OR REPLACE] PACKAGE nom_paquetage AS
    [-- Définition de types publics]
    [-- Déclaration de curseurs publics]
    [-- Déclaration de variables globales publiques (à éviter !)]
    [-- Déclaration de sous-programmes publics]
END;
```

-- Définition du corps (optionnelle)

```
CREATE [OR REPLACE] PACKAGE BODY nom_paquetage AS
    [-- Définition de types privés]
    [-- Spécification de curseurs publics et privés]
    [-- Déclaration de variables globales privées (à éviter !)]
    [-- Spécification de sous-programmes publics et privés]
END;
```

```
CREATE OR REPLACE PACKAGE Employes AS
    TYPE T_nuplet IS RECORD (ename emp.ename%TYPE,
                             salary emp.sal%TYPE);
    CURSOR salaireDec RETURN nuplet;
    PROCEDURE Embaucher (
        numemp NUMBER,
        nom      VARCHAR,
        job      VARCHAR,
        mgr      NUMBER,
        sal      NUMBER,
        comm     NUMBER,
        numdep   NUMBER);
    PROCEDURE Licencier (emp_id NUMBER);
END;
```

```
CREATE OR REPLACE PACKAGE BODY Employes AS
    CURSOR salaireDec RETURN nuplet IS
        SELECT empno, sal FROM emp ORDER BY sal DESC;
    PROCEDURE Embaucher (numemp NUMBER,
                        nom      VARCHAR, job      VARCHAR,
                        mgr      NUMBER, sal      NUMBER,
                        comm     NUMBER, numdep NUMBER) IS
    BEGIN
        INSERT INTO emp VALUES (numemp, nom, job,
                                mgr, SYSDATE, sal, comm, numdep);
    END;
    PROCEDURE Licencier (emp_id NUMBER) IS
    BEGIN
        DELETE FROM emp WHERE empno = emp_id;
    END;
END;
```

- ▶ **Définition** : Procédure stockée associée à une table et **exécutée automatiquement** lorsque des **événements** liés à des actions sur la table surviennent (mises à jour, principalement).
- ▶ Les déclencheurs complètent des contraintes d'intégrité en permettant de créer des règles d'intégrité complexes. Ce sont des éléments des **bases de données actives**.

Principaux types de déclencheurs

	Insertion	Deletion	Update
Before	1	2	3
After	4	5	6

```
CREATE [OR REPLACE] TRIGGER nom_declencheur
    BEFORE | AFTER
    INSERT | DELETE | UPDATE
| [INSERT] [[OR] DELETE] [[OR] UPDATE]

ON nom_table

[FOR EACH ROW]

-- Bloc PL/SQL codant les actions à effectuer
```

- ▶ **:NEW.nom_attribut** : Valeur d'un attribut **après** mise à jour
ex. `INSERT INTO client (1, 'NouveauClient');`
 - :NEW.NumCli prend la valeur 1 dans le déclencheur.
 - :NEW.Nom prend la valeur 'NouveauClient' dans le déclencheur.
- ▶ **:OLD.nom_attribut** : Valeur d'un attribut **avant** mise à jour
ex. `DELETE FROM client WHERE NumCli = 33;`
 - :OLD.NumCli prend la valeur 33 dans le déclencheur.

Exemple de déclencheur (1/2)

```
-- Emulation de clé primaire sur la table client
```

```
CREATE OR REPLACE TRIGGER clientPK  
BEFORE INSERT OR UPDATE ON client  
FOR EACH ROW
```

```
DECLARE
```

```
  n INTEGER;  
  cleExistante EXCEPTION;  
  cleNULLe EXCEPTION;
```

```
BEGIN
```

```
  -- La clé est-elle vide ?
```

```
  IF :NEW.NumCli IS NULL THEN  
    RAISE cleNULLe;  
  END IF;
```

Exemple de déclencheur (2/2)

```
-- La clé existe-t-elle déjà ?
SELECT COUNT(NumCli) INTO n FROM client
  WHERE NumCli = :NEW.NumCli;
IF n > 0 THEN
  RAISE cleExistante;
END IF;

EXCEPTION
  WHEN cleExistante THEN
    RAISE_APPLICATION_ERROR(-20501,
      'Clé primaire déjà utilisée !');
  WHEN cleNULLE THEN
    RAISE_APPLICATION_ERROR(-20502,
      'Une clé primaire doit avoir une valeur !');

END;
```

▶ Exemples

- Procédure stockée qui met la table EMP à jour
⇒ **SQL statique** (la requête est connue à la compilation)
- Procédure stockée qui met à jour une table dont le nom est un paramètre
⇒ **SQL dynamique** (la requête complète n'est pas connue à la compilation)

▶ Définition du SQL dynamique : Construction d'une requête SQL à la volée dans un bloc PL/SQL

- ▶ **Exécution** : EXECUTE IMMEDIATE **requete** -- **requete est une chaîne**
[INTO res1, res2...];
- ▶ **Note** :
 - Requêtes paramétrées : **valeurs** de la base de données
→ requêtes statiques
 - Si l'on veut paramétrer des **objets** (tables, vues, attributs...)
→ requêtes dynamiques
- ▶ **NB** : Les requêtes qui altèrent la structure de la base de données (CREATE, DROP, ALTER...), même statiques, **doivent** être exécutées en mode dynamique.

```
DECLARE
    requete VARCHAR(250);
    nomTable CHAR(4) := 'dept';
    numDep dept.deptno%TYPE := 50;
    n INTEGER;
BEGIN
    -- Construction de requête par concatenation
    requete := 'DELETE FROM ' || nomTable || ' WHERE deptno = ' || numDep;
    EXECUTE IMMEDIATE requete;
    -- Récupération d'un résultat de requête dynamique
    requete := 'SELECT COUNT(*) FROM ' || nomTable;
    EXECUTE IMMEDIATE requete INTO n;
END;
```

```
DECLARE -- Exemple
  TYPE T_cursDyn IS REF CURSOR;           -- Pointeur vers un curseur
  empCV T_cursDyn;                       -- Curseur dynamique
  nom emp.ename%TYPE;
  salaire emp.sal%TYPE := 10000;
BEGIN
  OPEN empCV FOR -- Le curseur est forcé explicitement
    'SELECT ename, sal FROM emp
     WHERE sal > ' || salaire;
  FETCH empCV INTO nom, salaire;
  WHILE empCV%FOUND LOOP
    -- Instructions PL/SQL
    FETCH empCV INTO nom, salaire;
  END LOOP;
  CLOSE empCV;
END;
```

Кінець