SQL déclaratif
du standard à la pratique

Langage d'interrogation des Bases de Données

CogniTIC
Building Together Better IT Solutions

Plan du cours

Partie 1 : Introduction aux concepts

Plan du cours

Partie 2 : DDL – Data Definition Language

(Langage de définition des données)

Plan du cours

Partie 3 : DRL – Data Retrieval Language

(Langage de d'extraction des données)

Plan du cours

Auto-Evaluation

Afin de vous assurer que vous êtes en phase avec la formation et que vous intégrez la matière correctement et à un bon rythme, nous vous proposerons, à la fin de chaque module du cours, un petit tableau d'évaluation. Ce tableau a pour but de rappeler les notions phares du module et nous vous invitons à le remplir pour vous-même afin de vous rendre compte des notions que vous devez peut-être revoir, de celles pour lesquelles vous devrez demander plus d'explications au formateur ou encore que vous avez tout compris !

Dans ces petites auto-évaluations, les lettres suivantes sont utilisées pour vous aider évaluer le niveau avec lequel vous sentez avoir compris la matière :

- **Parfait (P) :** vous avez parfaitement compris cette notion et vous vous sentez à votre aise
- **Satisfaisant (S) :** vous avez compris de quoi il s'agit mais la pratique vous manque
- **Vague (V) :** vous savez de quoi il s'agit, mais cela reste un peu vague dans votre esprit. Une explication supplémentaire du formateur ou une bonne révision de votre part s'impose
- **Insatisfaisant (I) :** Vous n'avez pas du tout compris la notion abordée, il faut tout faire pour y remédier !

Auto-Evaluation

Résumé de l'ensemble des notions

Notions	P	S	V	I
Base de données et SGBD				
Schéma relationnel (création, fonctionnement, lecture)				
Contrainte de « PRIMARY KEY »				
Contrainte de « FOREIGN KEY »				
Contraintes « UNIQUE », « CHECK », « NOT NULL »				
Création de table et contraintes (CREATE TABLE)				
Auto-incrémentation				
Modification de la structure d'une table existante (ALTER TABLE)				
Réinitialisation d'une table et suppression (TRUNCATE et DROP)				

Auto-Evaluation

Résumé de l'ensemble des notions

Notions	P	S	V	I
Ordre « SELECT … FROM » simple				
Clauses WHERE, GROUP BY et HAVING, ORDER BY				
Fonctions simples et fonctions d'agrégation				
Jointures				
Requêtes imbriquées				
Ordres DML (INSERT, UPDATE, DELETE)				
Transaction (principes, loi ACID, ordres COMMIT et ROLLBACK)				

Partie 1

INTRODUCTION AUX CONCEPTS

Base de données et SGBD

De l'analyse au relationnel

Notions de tables

Contraintes

Base de données et SGBD

Une **Base de données** est une structure, le plus souvent informatique, permettant le stockage et l'exploitation de données

Pendant longtemps, nous nous sommes contentés de stocker nos données sur **des supports papiers**, dans des classeurs, dans des livres, dans des bibliothèques. Sans autre ressource technologique, cette technique rendait parfois le stockage, l'organisation et l'exploitation des données **un peu lourd**, sans compter l'**espace nécessaire** pour entreposer ces données

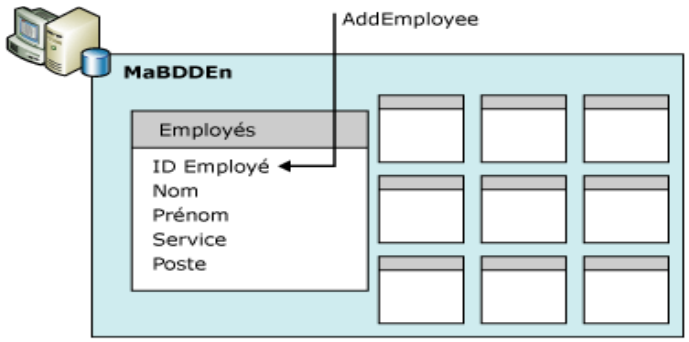
L'arrivée de l'ordinateur et des programmes informatiques a ensuite permit de **stocker les données dans des fichiers informatiques**, simplifiant et allégeant déjà énormément la gestion des données. Le gros désavantage des fichiers informatiques tels que ceux utilisés dans Excel ou Access est bien entendu que **ces fichiers ne se mettent pas à jour simultanément** pour l'ensemble des utilisateurs, chacun travaillant avec sa propre copie des données, la faisant évoluer à sa guise

Aujourd'hui, nous nous dirigeons donc vers des **bases de données dites « relationnelles »**, stockées sur un ou plusieurs **serveurs centralisés** qui peuvent être accédés par **de nombreux clients** de façon simultanée. Cela permet de rendre l'information disponible en temps réel, **partout et tout le temps**

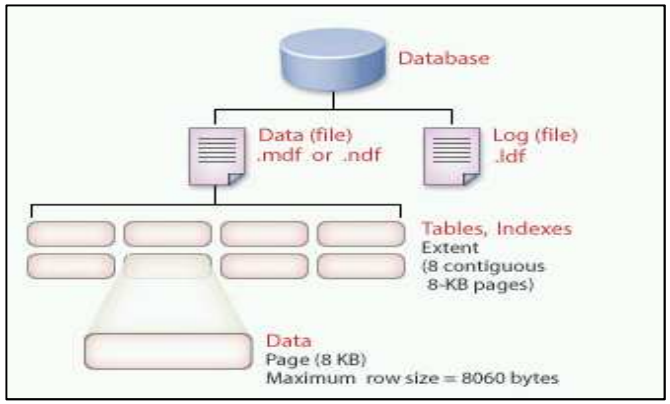
Base de données et SGBD

Un **Système de Gestion de Base de Données** (SGBD) est un programme informatique permettant de gérer des bases de données

Aujourd'hui, nombreux sont les systèmes permettant de créer des bases de données relationnelles. Un SGBD a la caractéristique de venir renforcer une base de données en lui apportant **un certains nombre de fonctionnalités supplémentaires**. C'est le nombre de fonctionnalités, leur efficacité et leur fluidité qui font qu'un SGBD sera plus populaire ou meilleur qu'un autre



Structure logique



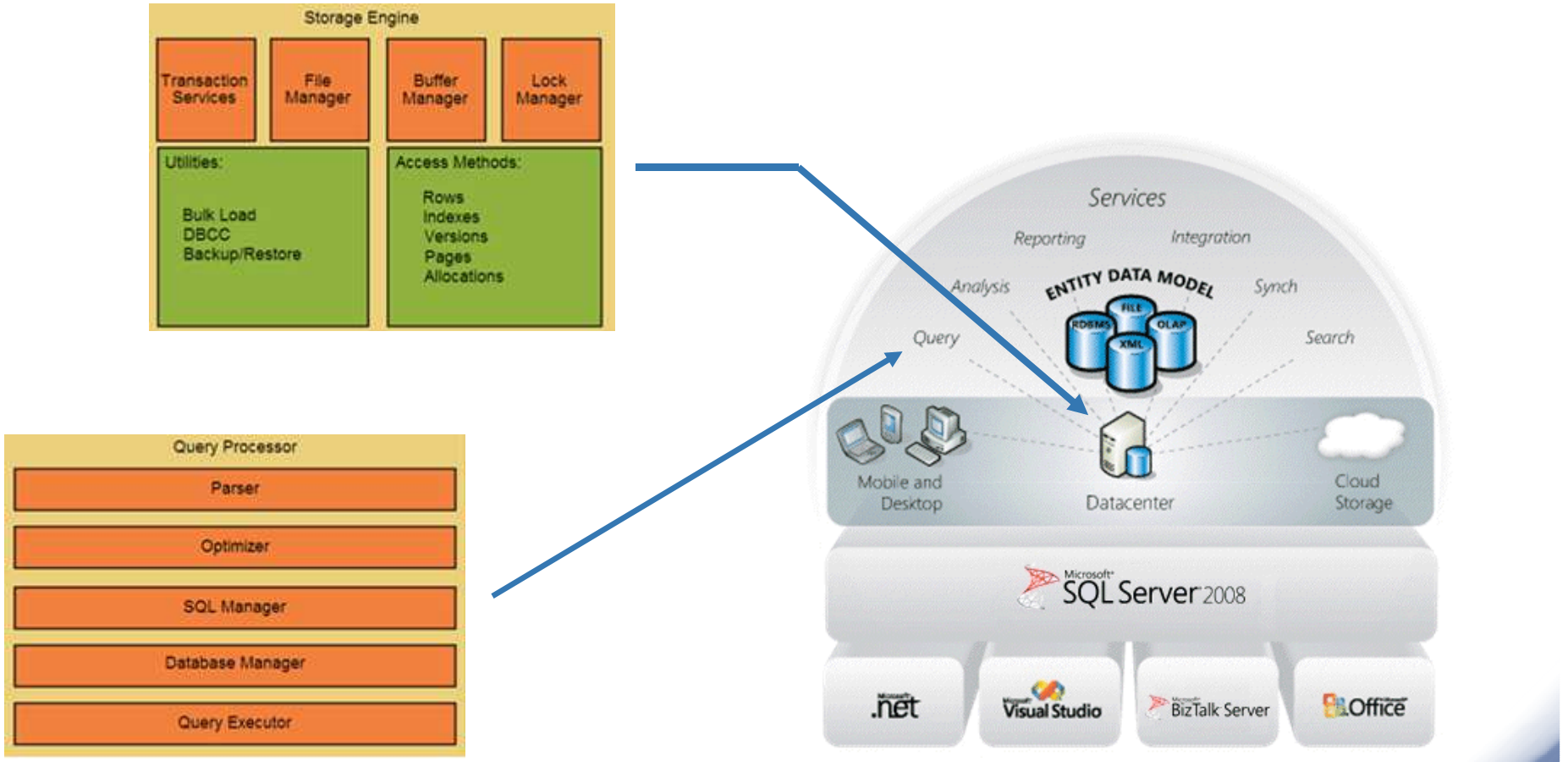
Structure physique

Base de données et SGBD

Fonctionnalités principales d'un SGBD :

- **Cohérence des données**
  Un SGBD doit garantir que les données contenues dans les bases de données respectent et respecteront toujours les **règles de logique relationnelles établies**
- **Concurrence d'accès aux données**
  Lorsque deux utilisateurs essayent d'accéder simultanément aux données, le système doit être capable d'assurer un ordre logique d'exécution des requêtes de chaque utilisateur, **en respectant les règles de transactions** qu'il a établi
- **Sécurité des données**
  Certainement le point le plus délicat qu'un SGBD doit pouvoir gérer, **la sécurité et l'accès au données** est bien souvent la chose la plus importantes pour les entreprises. Le système doit pouvoir garantir que seuls les utilisateurs autorisés accèderont aux données dont ils ont besoin
- **Pérennité des données**
  Avoir des données, c'est bien, les récupérer après un crash, c'est mieux ! Un SGBD doit donner accès à des outils ou des **méthodes de sauvegarde et de récupération** des données afin de pouvoir faire face à n'importe quelle panne logicielle, matérielle ou autre

Base de données et SGBD



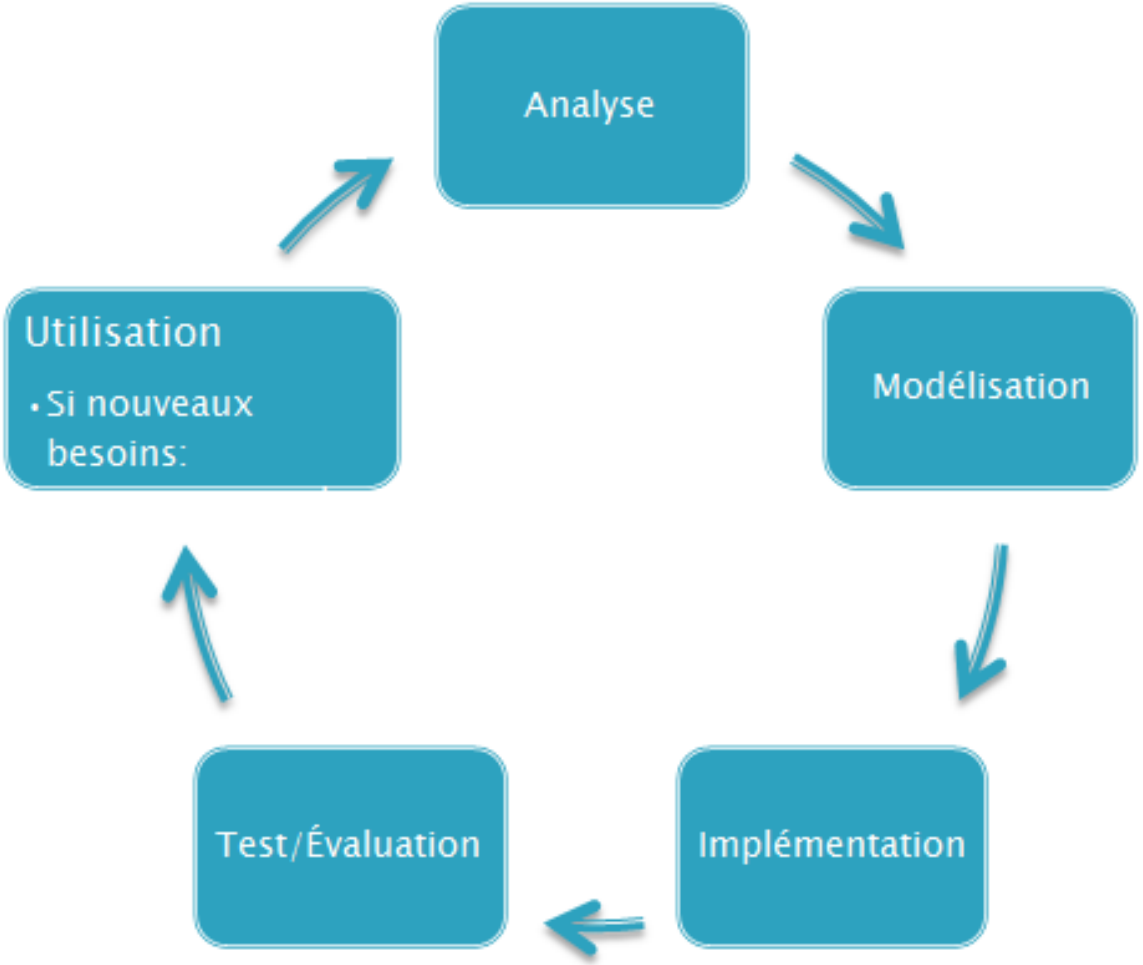
De l'analyse au relationnel

Avant d'arriver à la création de la base de données elle-même, **il est nécessaire d'analyser en profondeur le problème rencontré**. Cette phase d'analyse est nécessaire afin de ne rien oublier et de gagner un temps précieux au niveau du développement et de l'implémentation de la solution

La phase d'analyse passera par **plusieurs étapes** contenant chacune **un certain nombre de schémas et de diagrammes**. Il s'agira la plupart du temps d'appliquer un modèle d'analyse tel que *UML*, dans son intégralité ou partiellement du moins. Ces différents schémas permettront d'établir **le « schéma relationnel » de la base de données**, qui doit permettre aux développeurs de générer la base de données elle-même

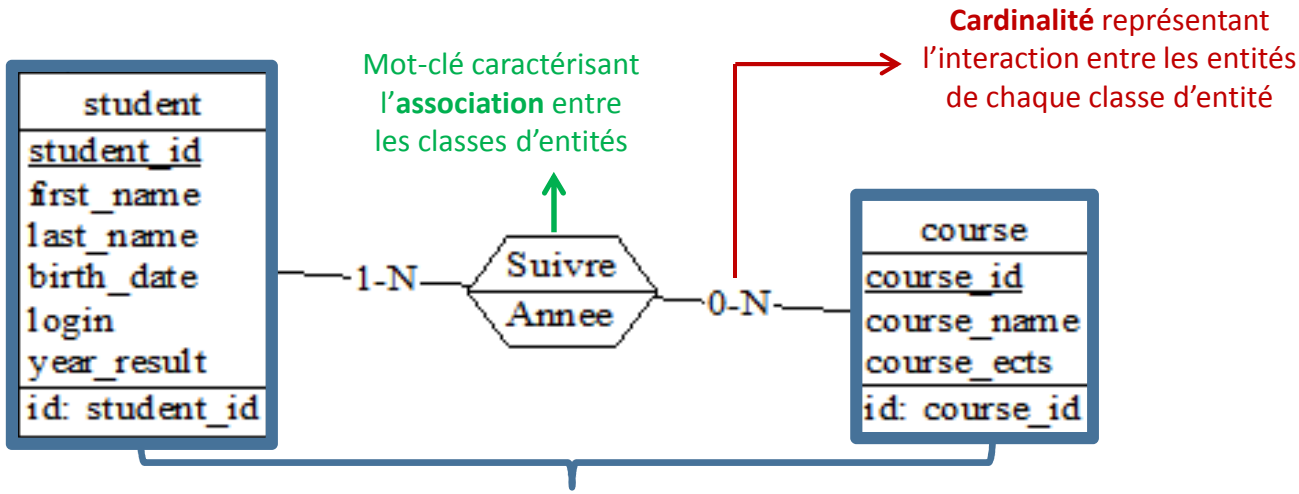
Dans certains cas simples, il est possible de se limiter à deux schémas. Le « **schéma Entités-Associations »** donnera alors directement la possibilité de passer au **schéma relationnel**

De l’analyse au relationnel



De l'analyse au relationnel

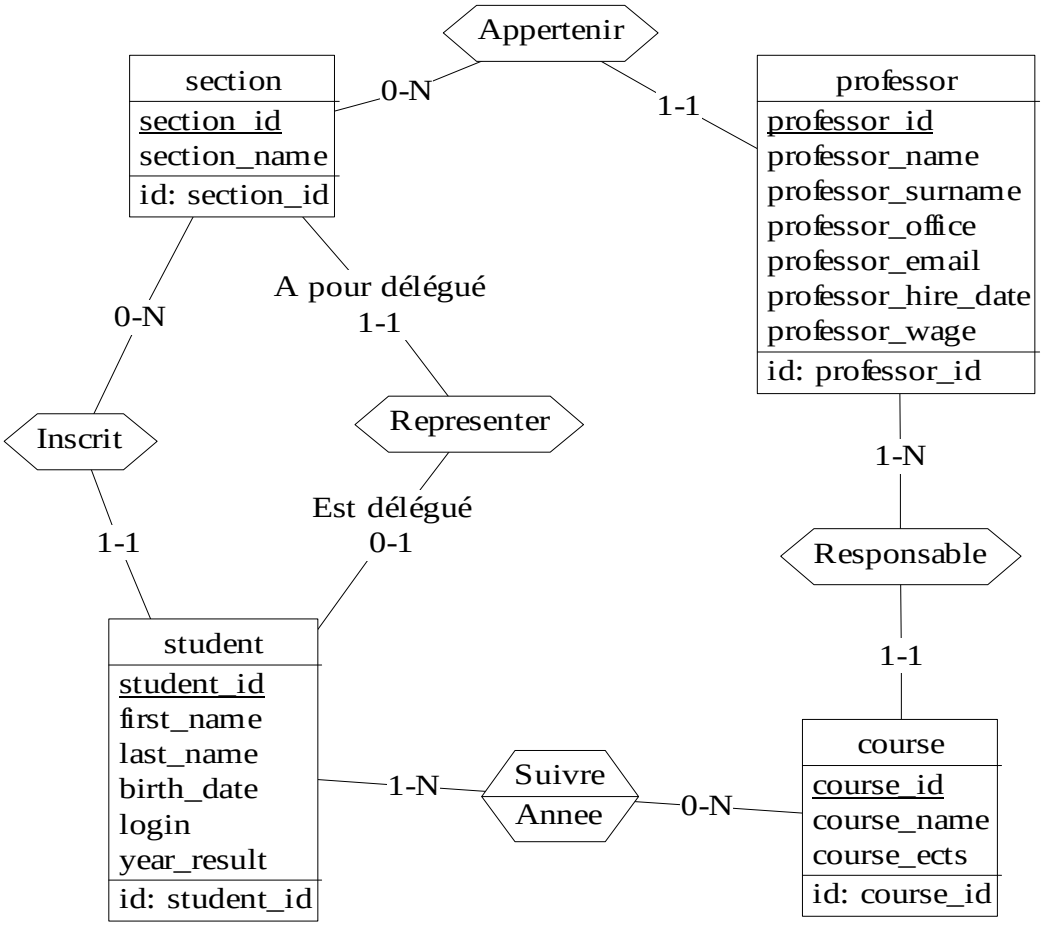
Le schéma entités-associations (EA) modélise simplement chaque acteur (**entité**) d'un système donné, en spécifiant chacun de leurs attributs qu'il est nécessaire d'enregistrer. Il indique **également la façon dont les acteurs interagissent** les uns avec les autres



Classes d'entités reprenant les **attributs** (caractéristiques) de chacune des **entités** (acteurs) du système modélisé

De l'analyse au relationnel

Exemple de ***schéma EA*** représentant le système d'information d'une université



De l'analyse au relationnel

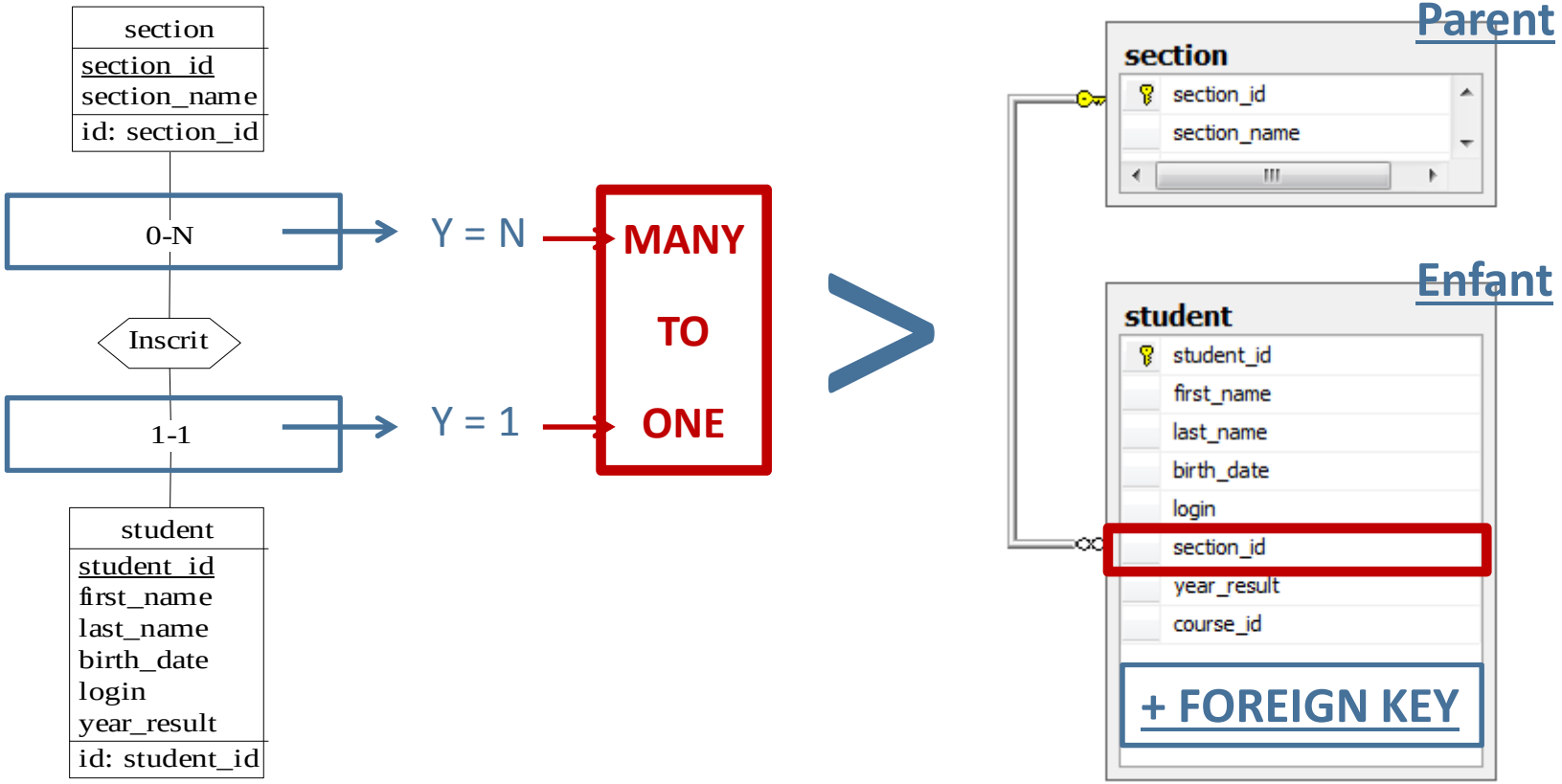
Le schéma relationnel est la traduction du schéma entités-associations
Un schéma relationnel représente le plan de la base de données
Il peut être lu aussi bien par l'analyste que par le programmeur

La traduction du schéma entités-associations se fait en appliquant certaines règles simples dont voici un aperçu succinct :

- Les **classes d'entités** deviennent des **tables**
- Les **attributs** deviennent des **colonnes**
- Les **associations** deviennent des **contraintes d'intégrité référentielles** selon les règles établies dans les slides suivants

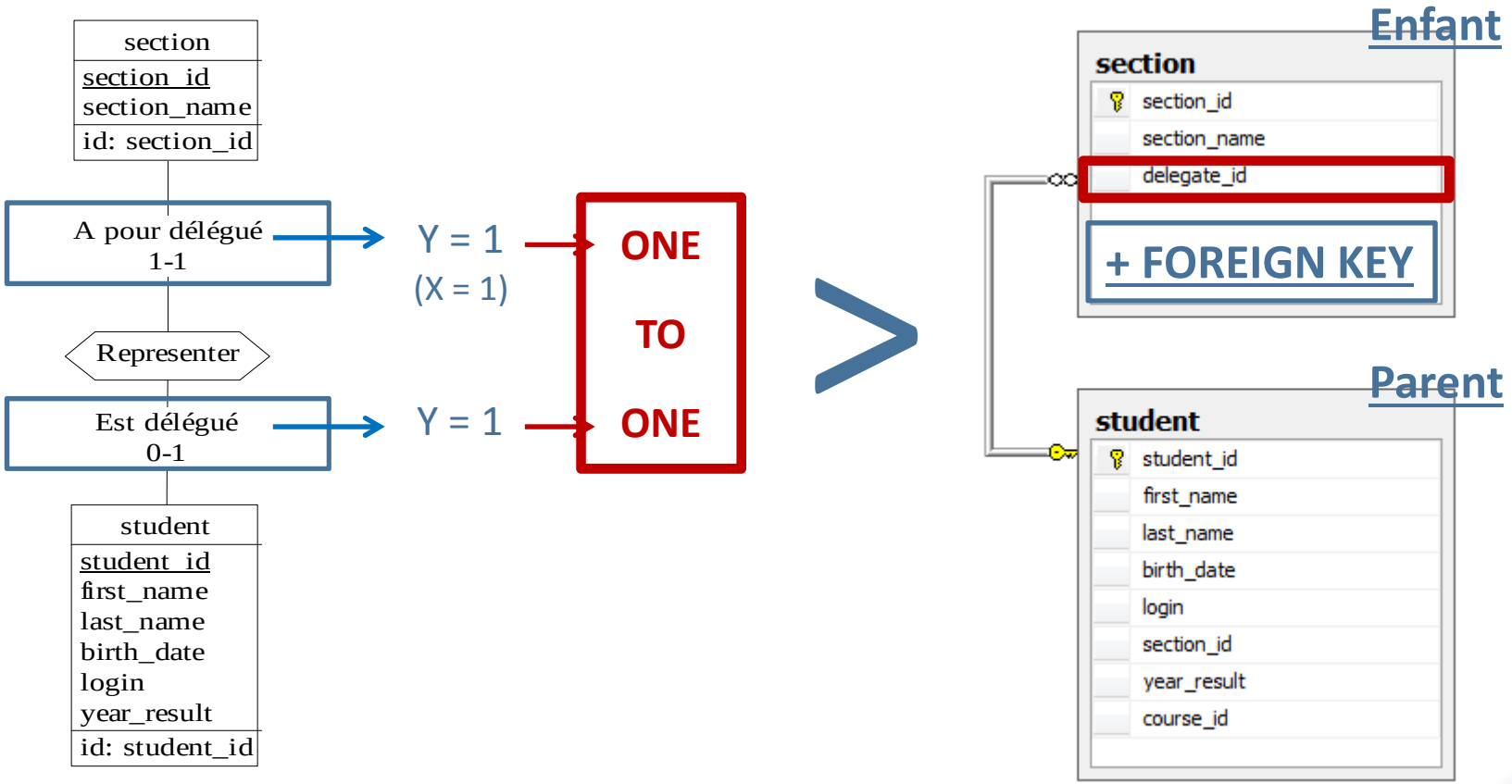
De l'analyse au relationnel

Traduction d'une association de type « One-to-Many » :



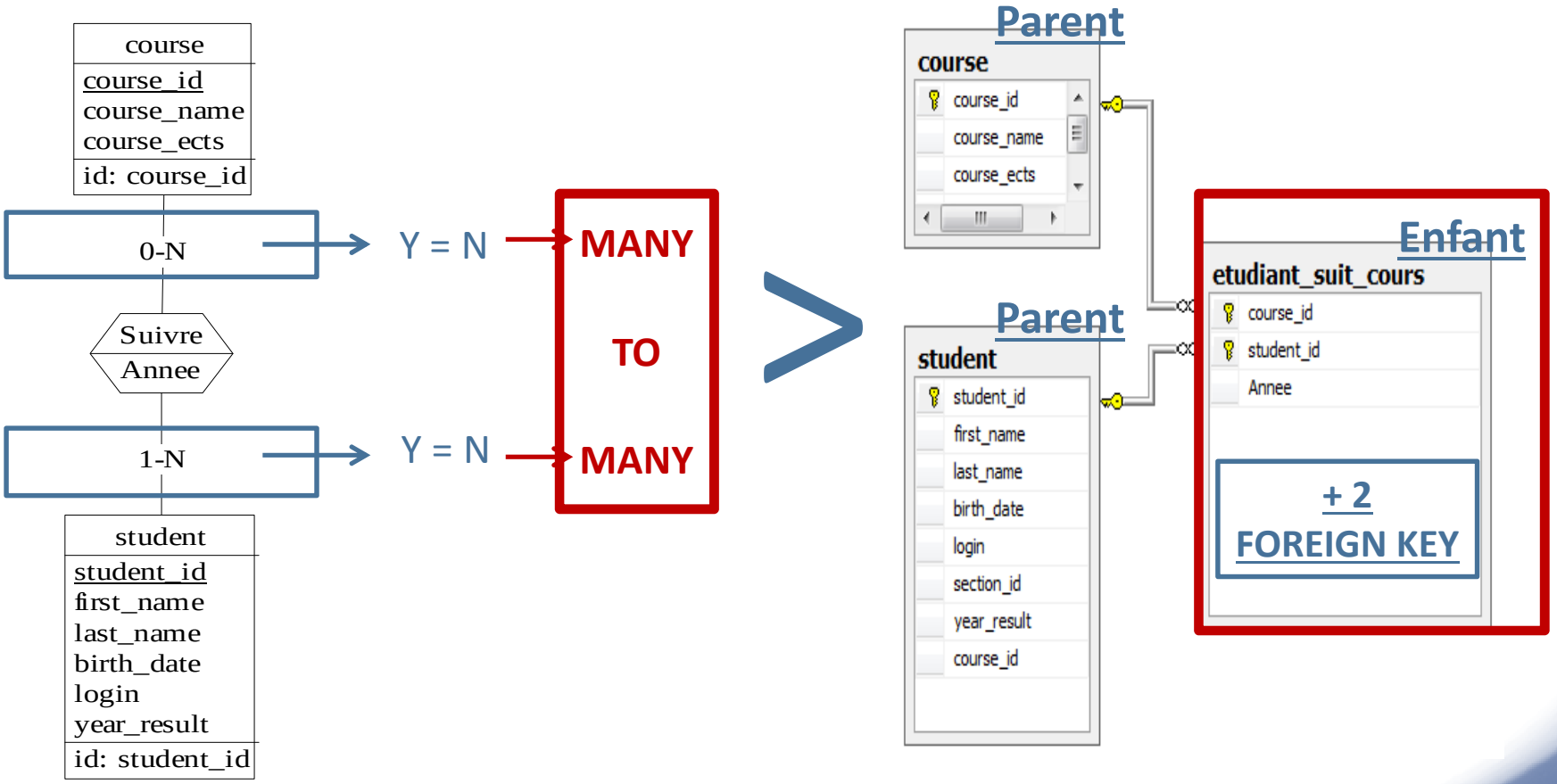
De l'analyse au relationnel

Traduction d'une association de type « One-to-One » :

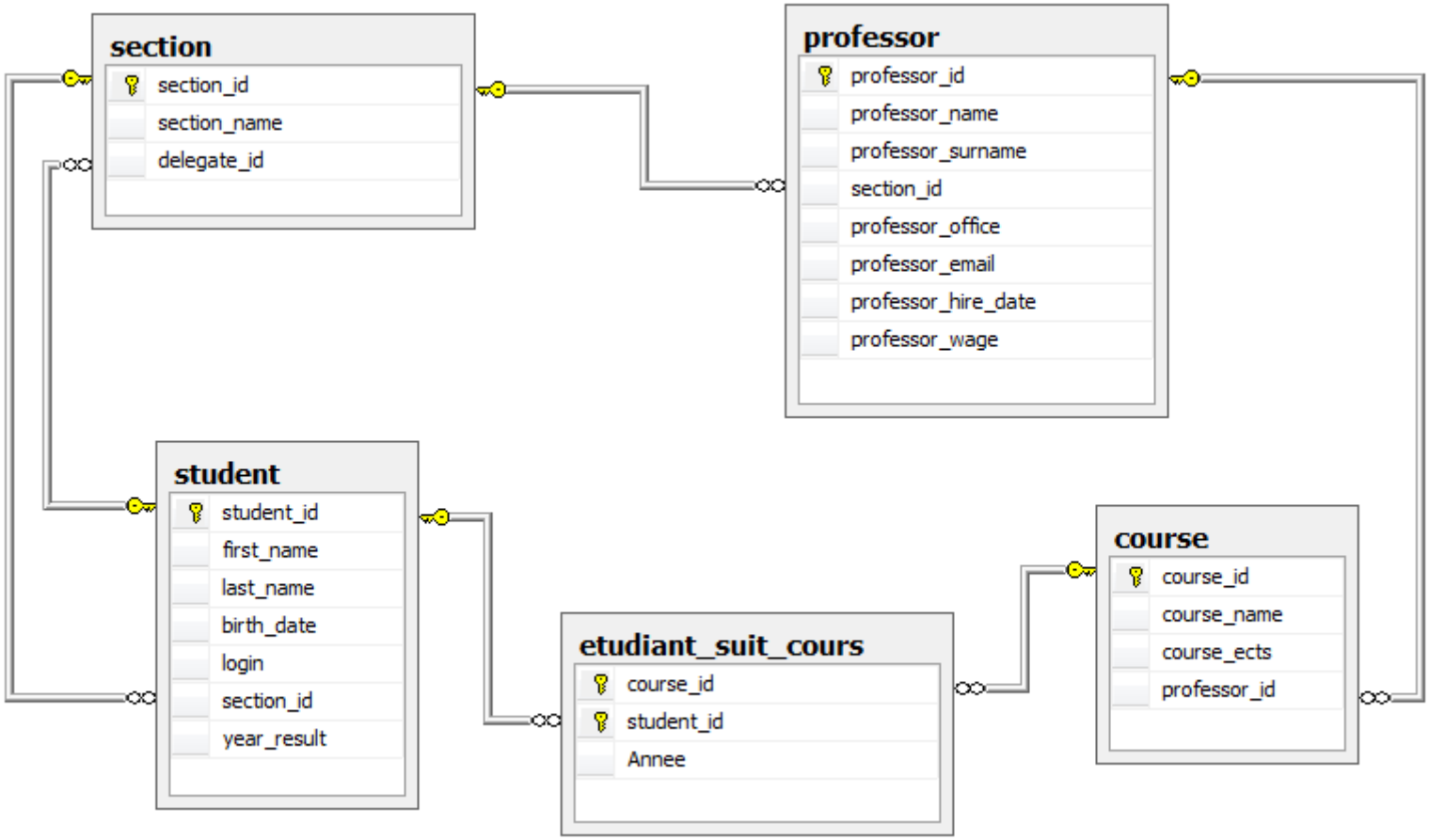


De l'analyse au relationnel

Traduction d'une association de type « Many-to-Many » :

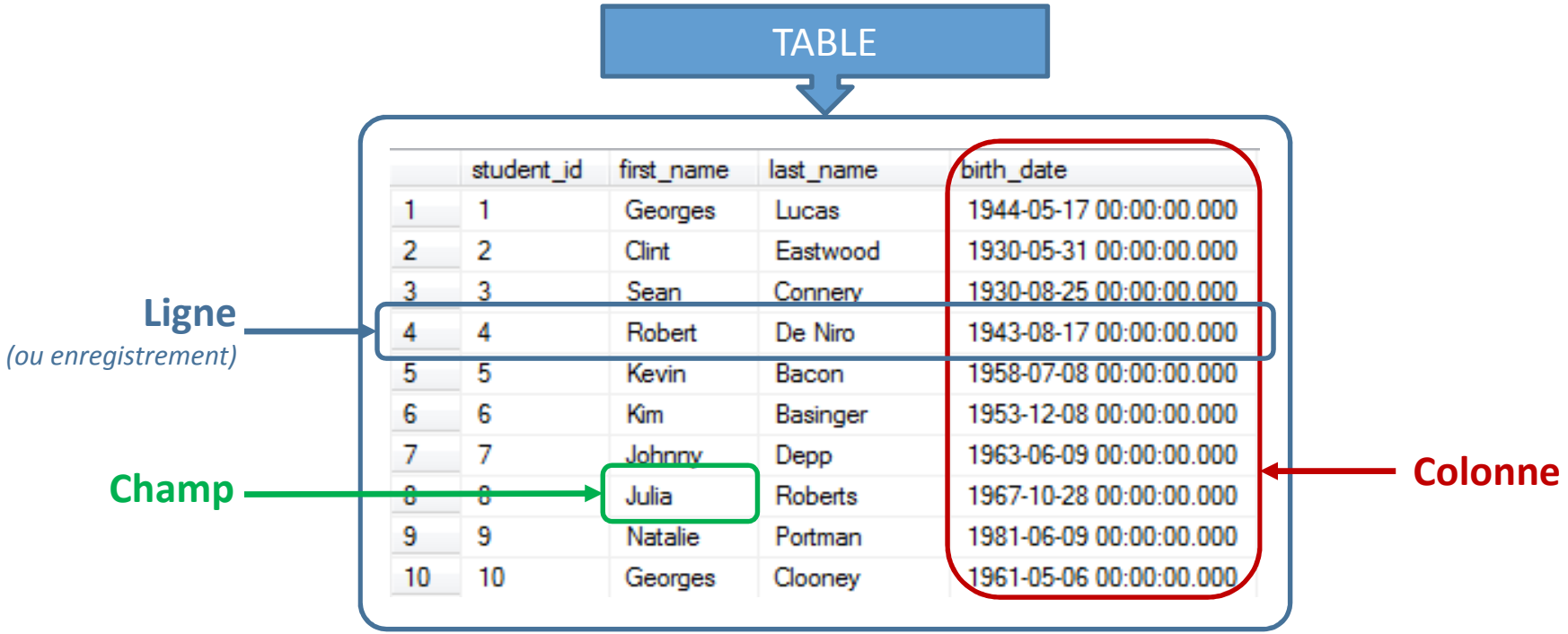


De l’analyse au relationnel



Notions de tables

Une **Table** regroupe des **ensembles de données (d'informations)** stockés de façon permanente et décrivant chacun un acteur du système réel modélisé **(Entité)**

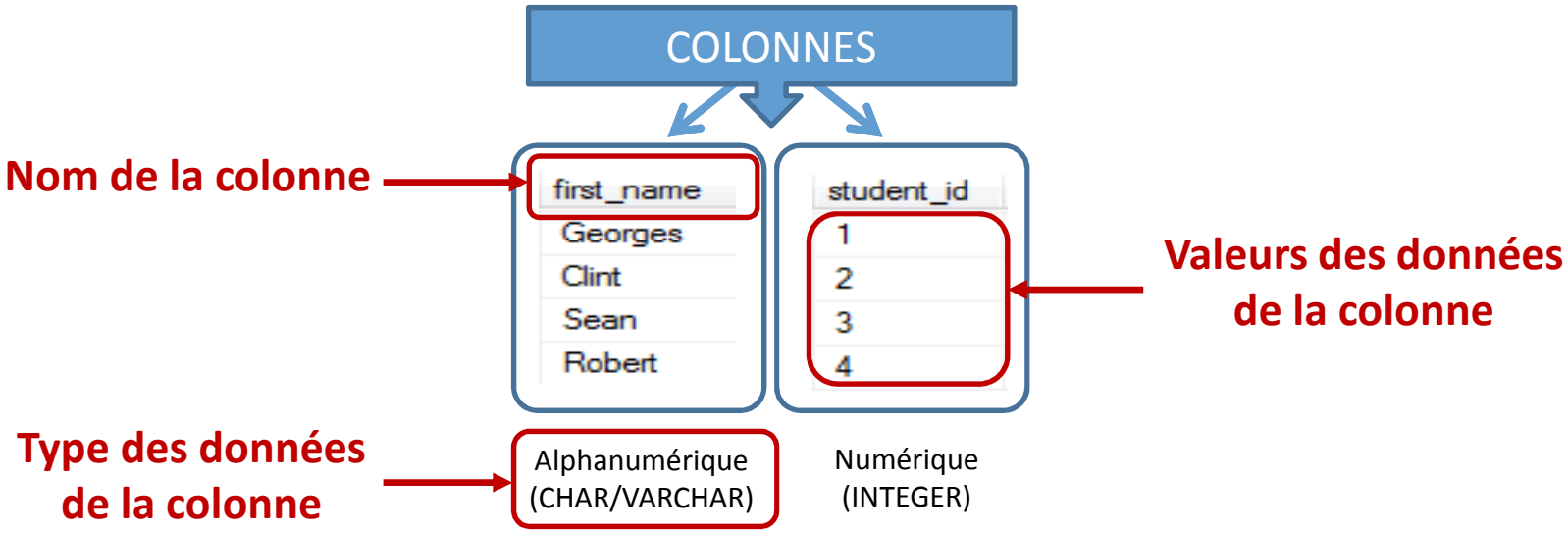


	student_id	first_name	last_name	birth_date
1	1	Georges	Lucas	1944-05-17 00:00:00.000
2	2	Clint	Eastwood	1930-05-31 00:00:00.000
3	3	Sean	Connery	1930-08-25 00:00:00.000
4	4	Robert	De Niro	1943-08-17 00:00:00.000
5	5	Kevin	Bacon	1958-07-08 00:00:00.000
6	6	Kim	Basinger	1953-12-08 00:00:00.000
7	7	Johnny	Depp	1963-06-09 00:00:00.000
8	8	Julia	Roberts	1967-10-28 00:00:00.000
9	9	Natalie	Portman	1981-06-09 00:00:00.000
10	10	Georges	Clooney	1961-05-06 00:00:00.000

La Table étudiant contient des informations sur les étudiants uniquement

Notions de tables

Une **Colonne** est un **attribut** d'une table, elle représente une **caractéristique particulière de l'objet réel** représenté



Contraintes

Une **Contrainte** est objet intégré à une table ***(comme les colonnes)***
Une contrainte possède **un nom, un type**
et **concerne au moins une colonne** de la table

On distingue principalement ***5 types de contraintes différentes***

Seule la contrainte ***« NOT NULL »*** *n'aura* ***pas de nom***

Il n'existera au maximum ***qu'une seule contrainte de « Clé Primaire »*** *pour chaque table*

Contraintes	Description
NOT NULL } **CLE PRIMAIRE**	Force la présence d'une valeur (valeur obligatoire)
UNIQUE	Empêche les valeurs-doublons
CHECK	Conditionne les valeurs (expression conditionnelle)
FOREIGN KEY	Conditionne les valeurs par rapport à une autre table

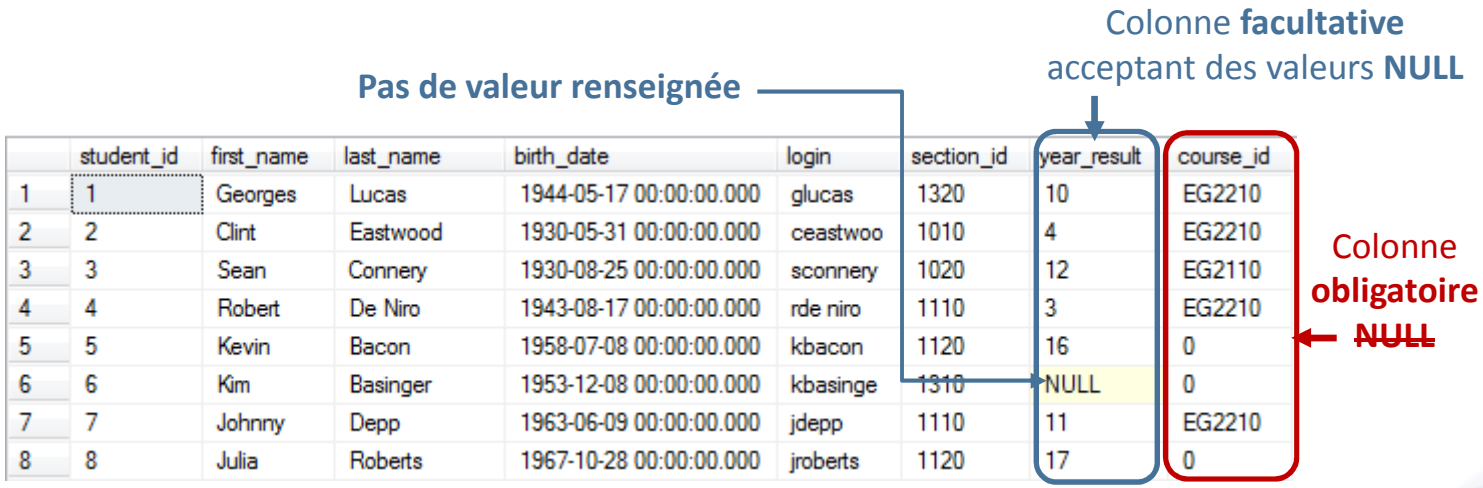
Contraintes : **NOT NULL**

Ajouter **la contrainte « NOT NULL »** à la colonne d'une table obligera cette colonne à **contenir une valeur** pour chacune des lignes de la table

*Une colonne non-obligatoire est **facultative**, elle peut contenir la valeur **NULL***

*La valeur **NULL** spécifie que aucune valeur n'est attribuée*

***NULL** représente l'absence de valeur, elle n'est ni 0 ni une case vide*



	student_id	first_name	last_name	birth_date	login	section_id	year_result	course_id
1	1	Georges	Lucas	1944-05-17 00:00:00.000	glucas	1320	10	EG2210
2	2	Clint	Eastwood	1930-05-31 00:00:00.000	ceastwoo	1010	4	EG2210
3	3	Sean	Connery	1930-08-25 00:00:00.000	sconnery	1020	12	EG2110
4	4	Robert	De Niro	1943-08-17 00:00:00.000	rde niro	1110	3	EG2210
5	5	Kevin	Bacon	1958-07-08 00:00:00.000	kbacon	1120	16	0
6	6	Kim	Basinger	1953-12-08 00:00:00.000	kbasinge	1310	NULL	0
7	7	Johnny	Depp	1963-06-09 00:00:00.000	jdepp	1110	11	EG2210
8	8	Julia	Roberts	1967-10-28 00:00:00.000	jroberts	1120	17	0

Contraintes : **UNIQUE**

La contrainte d'unicité « UNIQUE » a la particularité d'empêcher **une colonne ou une combinaison de colonnes**, d'accepter deux valeurs identiques pour deux lignes différentes de la table

Colonne **non candidate** contient des **doublons**

Colonne **UNIQUE** **aucun doublon**

	student_id	first_name	last_name	birth_date	login	section_id	year_result	course_id
1	1	Georges	Lucas	1944-05-17 00:00:00.000	glucas	1320	10	EG2210
2	2	Clint	Eastwood	1930-05-31 00:00:00.000	ceastwoo	1010	4	EG2210
3	3	Sean	Connery	1930-08-25 00:00:00.000	sconnery	1020	12	EG2110
4	4	Robert	De Niro	1943-08-17 00:00:00.000	rde niro	1110	NULL	EG2210
5	5	Kevin	Bacon	1958-07-08 00:00:00.000	kbacon	1120	16	0
6	6	Kim	Basinger	1953-12-08 00:00:00.000	kbasinge	1310	NULL	0
7	7	Johnny	Depp	1963-06-09 00:00:00.000	jdepp	1110	11	EG2210
8	8	Julia	Roberts	1967-10-28 00:00:00.000	jroberts	1120	17	0
9	9	Natalie	Portman	1981-06-09 00:00:00.000	nportman	1010	4	EG2210
10	10	Georges	Clooney	1961-05-06 00:00:00.000	gclooney	1020	4	EG2110

Combinaison de colonnes **UNIQUE** **aucun doublon**

Contraintes : **PRIMARY KEY**

La contrainte de clé primaire d'une table désigne un ensemble (souvent minimal) de colonnes qui identifient de manière unique les enregistrements d'une table
La combinaison des colonnes qui la composent doit être **UNIQUE** et **NON NULL**

- *Il ne peut exister qu'une et une seule clé primaire par table* **MAIS** celle-ci peut être composée d'une combinaison de plusieurs colonnes

- **Une clé primaire est dite « naturelle »** si la ou les colonnes qui la composent sont des attributs représentant réellement une caractéristique de l'objet que la table décrit

- **Une clé primaire sera « artificielle »** si elle est représentée par une colonne dont les valeurs ne représentent rien dans la monde réel. Ces colonnes artificielles sont régulièrement utilisées car elles sont faciles à manipuler et, comme elles ont le plus souvent comme valeurs des nombres, leur contenu peut être géré directement par le SGBD. On dira alors, qu'en plus d'être une clé primaire artificielle, **les valeurs de la colonne sont auto-incrémentées**, le plus souvent de 1 à l'infini

Contraintes : PRIMARY KEY

Combinaison **non candidate**
(contient des valeurs **NULL**)

	student_id	first_name	last_name	birth_date	login	section_id	year_result	course_id
1	1	Georges	Lucas	1944-05-17 00:00:00.000	glucas	1320	10	EG2210
2	2	Clint	Eastwood	1930-05-31 00:00:00.000	ceastwoo	1010	4	EG2210
3	3	Sean	Connery	1930-08-25 00:00:00.000	sconnery	1020	12	EG2110
4	4	Robert	De Niro	1943-08-17 00:00:00.000	rde niro	1110	NULL	EG2210
5	5	Kevin	Bacon	1958-07-08 00:00:00.000	kbacon	1120	16	0
6	6	Kim	Basinger	1953-12-08 00:00:00.000	kbasinge	1310	NULL	0
7	7	Johnny	Depp	1963-06-09 00:00:00.000	jdepp	1110	11	EG2210
8	8	Julia	Roberts	1967-10-28 00:00:00.000	jroberts	1120	17	0
9	9	Natalie	Portman	1981-06-09 00:00:00.000	nportman	1010	4	EG2210
10	10	Georges	Clooney	1961-05-06 00:00:00.000	gclooney	1020	4	EG2110

PRIMARY KEY
dite **« artificielle »**
(**PEUT** être **auto-incrémentée**)

Combinaison **UNIQUE** ET **NOT NULL**
candidate à la création d'une
PRIMARY KEY « naturelle »

Contraintes : **FOREIGN KEY**

La contrainte de clé étrangère d'une table (**enfant**) permet d'éviter la redondance d'information au niveau de cette table par le biais d'une colonne de référence pointant vers une colonne identifiante d'une autre table (**parent**)
Contenant, elle, le détail de l'objet référencé

- La clé étrangère est une contrainte de la table, elle concerne une ou plusieurs colonnes de la table, mais ces colonnes ne sont pas implicitement créées lors de la création de la clé étrangère. De la même manière, modifier ou supprimer la clé étrangère ne modifie pas les caractéristiques de la colonne concernée
- Il peut exister plusieurs clés étrangère dans chaque table
- Plusieurs colonnes peuvent composer la même clé étrangère d'une table, si cette clé étrangère fait référence à plusieurs colonnes d'une autre table (clé primaire composite, clé d'unicité composite)
- Les colonnes référencées entre elles doivent être du même type
- Les valeurs d'une colonne utilisée comme clé étrangère peuvent être **NULL**

Contraintes : FOREIGN KEY

Table Livres

Information redondante

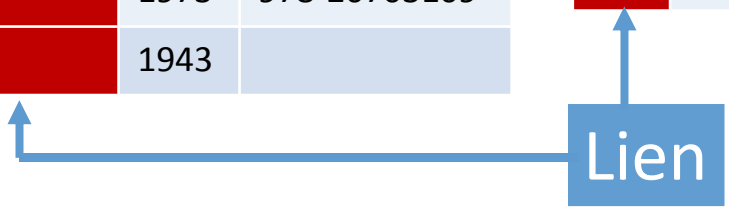
Titre	PrenomAuteur	NomAuteur	PseudoAuteur	NaissAuteur	DateLivre	ISBN
La Peste	Albert	Camus	Bebert	07/11/1913	1974	978-20703604
La Chute	Albert	Camus	Bebert	07/11/1913	1973	978-20703109
Huis-clos	Jean-Paul	Sartre	Jy-Pé	21/06/1905	1943	

Table Livres

Titre	RefAuteur	Date	ISBN
La Peste	1	1974	978-20703604
La Chute	1	1973	978-20703109
Huis-clos	2	1943	

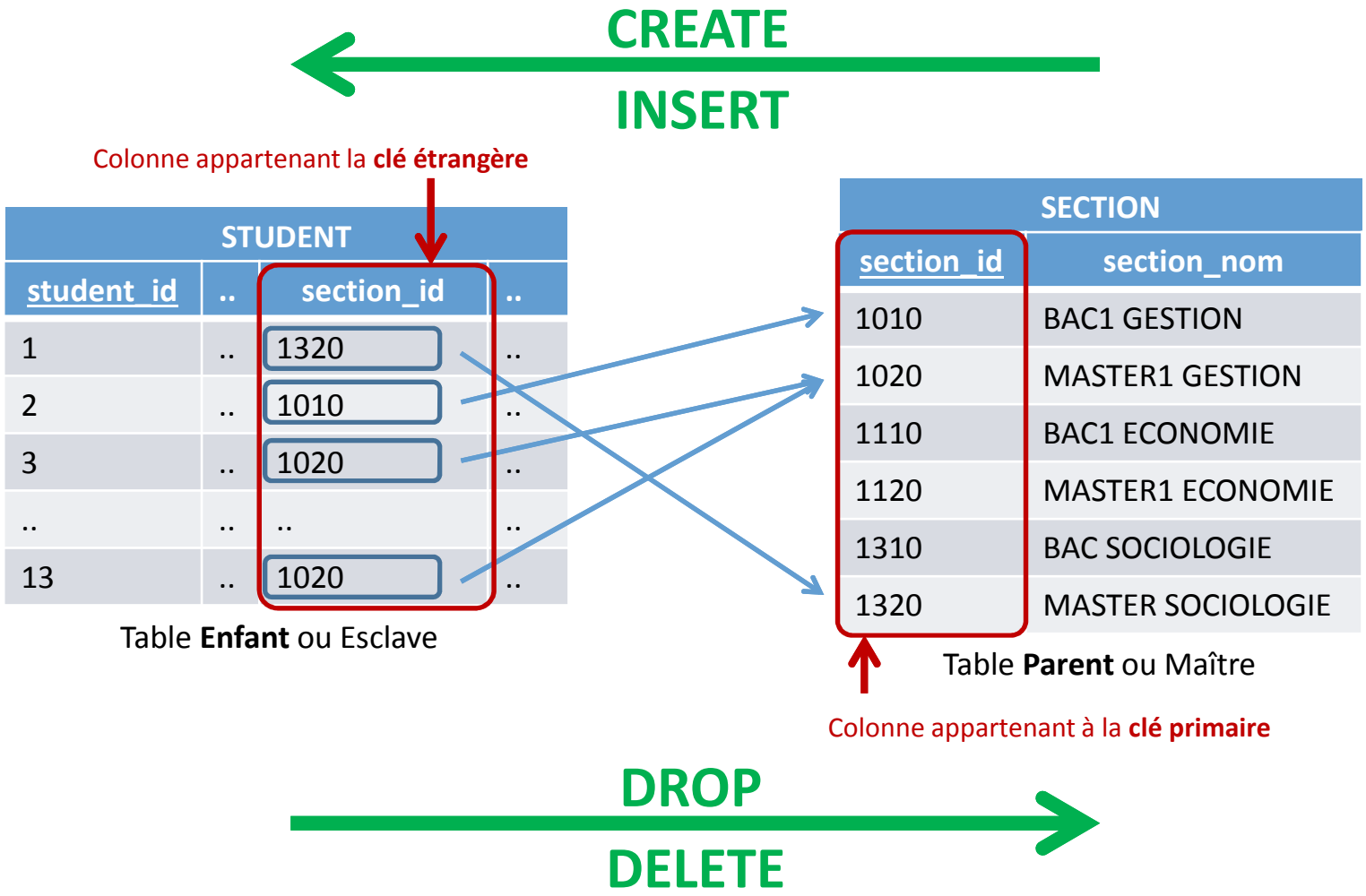
Table Auteurs

Ref	Nom	Prenom	Pseudo	DateNaiss
1	Albert	Camus	Bebert	07/11/1913
2	Jean-Paul	Sartre	Jy-Pé	21/06/1905



La table ***Livres*** est liées à la table ***Auteurs*** par le champ ***RefAuteur***.

Contraintes : FOREIGN KEY



Contraintes : **CHECK**

La contrainte CHECK d'une table permet de poser une condition spécifique sur les colonnes de la tables, afin d'y empêcher l'insertion de n'importe quelle valeur

0 ≤ year_result ≤ 20

	student_id	first_name	last_name	birth_date	login	section_id	year_result	course_id
1	1	Georges	Lucas	1944-05-17 00:00:00.000	glucas	1320	10	EG2210
2	2	Clint	Eastwood	1930-05-31 00:00:00.000	ceastwoo	1010	4	EG2210
3	3	Sean	Connery	1930-08-25 00:00:00.000	sconnery	1020	12	EG2110
4	4	Robert	De Niro	1943-08-17 00:00:00.000	rde niro	1110	NULL	EG2210
5	5	Kevin	Bacon	1958-07-08 00:00:00.000	kbacon	1120	16	0
6	6	Kim	Basinger	1953-12-08 00:00:00.000	kbasinge	1310	NULL	0
7	7	Johnny	Depp	1963-06-09 00:00:00.000	jdepp	1110	11	EG2210
8	8	Julia	Roberts	1967-10-28 00:00:00.000	jroberts	1120	17	0
9	9	Natalie	Portman	1981-06-09 00:00:00.000	nportman	1010	4	EG2210
10	10	Georges	Clooney	1961-05-06 00:00:00.000	gclooney	1020	4	EG2110

Birth_date > 01-01-1930

Partie 2

DDL – DATA DEFINITION LANGUAGE

CREATE TABLE

IDENTITY et DEFAULT

Contraintes

ALTER TABLE

TRUNCATE TABLE

DROP TABLE

CREATE TABLE

CREATE TABLE *nom_table* **(**
nom_colonne1 **TYPE,**
nom_colonne2 **TYPE,**
nom_colonne3 **TYPE**
)

```
CREATE TABLE student (
  student_id int,
  first_name varchar(50),
  last_name varchar(50),
  birth_date datetime,
  login varchar(50),
  section_id int,
  year_result int,
  course_id varchar(6)
)
```

CREATE TABLE

- **Une table possède un NOM**

  Ce nom peut être composé de **128 caractères** et est choisi par l'utilisateur. Il est conseillé de choisir des **noms clairs et concis**. Les **caractères spéciaux** comme les accents et les espaces blancs sont **interdits**

- **Une table possède des COLONNES**

  Le nom de ces colonnes est limité par les mêmes contraintes que celles énoncées pour le nom de la table

- **Les colonnes ont un TYPE**

  Le type de la colonne défini le type qu'auront les valeurs de cette colonne. Les types principaux sont **INT** (pour entiers), **VARCHAR(taille)** (pour une chaine de caractères), **DECIMAL(X,Y)** ou **FLOAT** (pour un chiffre décimal), **DATE** (pour une date). Une liste plus détaillée des différents types de données est fournie dans les slides suivants

CREATE TABLE

Types principaux de données

Nom	Taille sur le disque (octets)	Description
INTEGER *(INT)*	4	Valeurs entières de -2^31 à 2^31
DECIMAL(x,y) *(DEC)*	5 à 17	Valeurs numériques contenant « x » chiffres au total, dont « y » après la virgule
VARCHAR(x)	Nombre de caractères + 2	Chaîne de caractères d'une taille variable pouvant aller de 1 caractère à 8000 caractères (non-Unicode)
DATE	3	Dates du 01/01/0001 au 31/12/9999
BIT	1	Valeurs 1, 0 ou NULL
VARBINARY(x)	Nombres de bits + 2	Valeurs binaires d'une taille de 1 à 8000 (stockage de fichiers de type son, image ou vidéo)

CREATE TABLE

Types de données numériques

Nom	Taille sur le disque (octets)	Description
TINYINT	1	Valeurs entières de 0 à 255
SMALLINT	2	Valeurs entières de -2^15 à 2^15 - 1
BIGINT	8	Valeurs entières de -2^63 à 2^63
NUMERIC(x,y)	5 à 17	*(Idem DECIMAL)*

Types de données d'approximation

Nom	Taille sur le disque (octets)	Description
FLOAT(x) *(1 ≤ x ≤ 53)*	4 à 8 (dépend de x)	Valeurs numériques d'approximation (e)
REAL	4	Valeurs numériques d'approximation

CREATE TABLE

Types de données textuelles

Nom	Taille sur le disque (octets)	Description
CHAR(x)	4	*(Idem VARCHAR mais de taille fixe, non-Unicode)*
NCHAR(x)	Nombre de caractères * 2	Chaîne de caractères d'une taille fixe pouvant aller de 1 caractère à 4000 caractères (Unicode)
NVARCHAR(x)	Nbre de caractères * 2 + 2	Chaîne de caractères d'une taille variable pouvant aller de 1 caractère à 4000 caractères (Unicode)
TEXT ***DÉPRÉCIÉ***	2^31-1 (2 147 483 647)	Chaîne de caractères d'une taille variable pouvant aller jusqu'à 2^31-1 caractères (non-Unicode)
NTEXT ***DÉPRÉCIÉ***	Nombre de caractères * 2	Chaîne de caractères d'une taille variable pouvant aller jusqu'à 2^30-1 (1 073 741 823) caractères (Unicode)

CREATE TABLE

Types de données de dates et heures

Nom	Taille sur le disque (octets)	Description
TIME	5	Heures allant de 00:00:00 à 23:59:59.9999999
SMALLDATETIME	4	Dates allant du 01/01/1900 au 06/06/2079 et heures allant de 00:00:00 à 23:59:59
DATETIME	8	Dates allant du 01/01/1753 au 31/12/9999 et heures allant de 00:00:00 à 23:59:59.997
DATETIME2	6 à 8	Dates allant du 01/01/0001 au 31/12/9999 et heures allant de 00:00:00 à 23:59:59.9999999

CREATE TABLE

Autres types de données

Nom	Taille sur le disque (octets)	Description
TIMESTAMP ***(DÉPRÉCIÉ, utiliser « ROWVERSION »)***	8	Champ auto-incrémenté qui génère une nouvelle valeur chaque fois que la ligne est mise à jour *(ex : 0x00000000000007D4)*
UNIQUEIDENTIFIER	16	Valeurs hexadécimales uniques *(ex : 6F9619FF-8B86-D011-B42D-00C04FC964FF)*
XML	Maximum 2Go	Données XML

Vous trouverez une liste complète des types utilisés sur SQL-Server à l'adresse suivante :
http://msdn.microsoft.com/en-us/library/ms187752.aspx

CREATE TABLE

ORACLE

Types principaux de données sous Oracle 12c

Nom	Taille sur le disque (octets)	Description
NUMBER(x,y)	1 à 22	Valeurs numériques contenant « x » chiffres au total, dont « y » après la virgule. (x,y) non-obligatoire, valeur max par défaut si non spécifiés Valeurs possibles : -1.0x10^130 à 1.0^126
VARCHAR2(x)	Nombre de caractères	Chaîne de caractères d'une taille variable pouvant aller de 1 caractère à 4000 caractères (non-Unicode)
DATE	7	Dates du 01/01/0001 au 31/12/9999

Vous trouverez une liste complète des types utilisés sur Oracle 12c à l'adresse suivante :
http://docs.oracle.com/cd/E16655_01/server.121/e17209/sql_elements001.htm#SQLRF50972

IDENTITY et DEFAULT

CREATE TABLE *nom_table* **(**
nom_colonne1 **TYPE IDENTITY(1,1),**
nom_colonne2 **TYPE DEFAULT** *valeur_par_défaut*,
nom_colonne3 **TYPE**
)

```
CREATE TABLE student (
  student_id int IDENTITY(1,1),
  first_name varchar(50),
  last_name varchar(50) DEFAULT 'Smith',
  birth_date datetime,
  login varchar(50),
  section_id int,
  year_result int DEFAULT 0,
  course_id varchar(6)
)
```

IDENTITY et DEFAULT

- **Les valeurs d'une colonne peuvent être AUTO-INCRÉMENTÉES**

  Sous SQL-Server, le mot-clé **« IDENTITY »** permet de demander au système d'attribuer lui-même des valeurs à la colonne concernée. Ces valeurs commencent bien souvent à 1 et son incrémentées de 1 à chaque nouvelle ligne **« (1,1) »**. Il ne faut pas s'occuper de cette colonne lors de l'insertion d'une nouvelle ligne dans la table

  ATTENTION : *Sous SQL Server, il n'est pas possible* ***(en utilisant du code)*** *de rajouter l'auto-incrémentation d'une colonne lorsque la table est déjà créée, il faut y penser à la création ! Sinon il sera nécessaire de supprimer et recréer la table…*

  Remarque : *Comme expliqué plus loin, l'ordre DDL « TRUNCATE TABLE » permet de vider la table de ses données et de réinitialiser le compteur de l'auto-incrémentation par la même occasion*

- **Une colonne peut posséder une VALEUR PAR DÉFAUT**

  Le mot-clé **« DEFAULT »** permet d'insérer une valeur par défaut dans la colonne concernée, si aucune valeur n'est spécifiée pour cette colonne lors de l'insertion d'une nouvelle ligne dans la table. Si une valeur est renseignée, elle remplace bien entendu la valeur par défaut. Le contrainte ***« NOT NULL »*** n'a aucun sens lorsque ***« DEFAULT »*** est utilisé et peut être source d'erreur sur certaines plateformes

IDENTITY et DEFAULT

ORACLE

CREATE TABLE *nom_table* **(**
nom_colonne1 **TYPE GENERATED ALWAYS AS IDENTITY,**
nom_colonne2 **TYPE DEFAULT** *valeur_par_défaut*,
nom_colonne3 **TYPE**
)

```
CREATE TABLE student (
  student_id   NUMBER GENERATED ALWAYS AS IDENTITY,
  first_name   VARCHAR2(50),
  last_name    VARCHAR2(50) DEFAULT 'Smith',
  birth_date   DATE,
  login        VARCHAR2(50),
  section_id   INT,
  year_result  INTEGER DEFAULT 0,
  course_id    CHAR(6)
);
```

Contraintes

CREATE TABLE *nom_table* **(**
nom_colonne1 **TYPE,**
nom_colonne2 **TYPE,**
CONSTRAINT *nom_contrainte* **TYPE CONTRAINTE (***colonne_concernée***)**
)

CREATE TABLE *nom_table* **(**
nom_colonne1 **TYPE,**
nom_colonne2 **TYPE CONSTRAINT** *nom_contrainte* **TYPE CONTRAINTE,**
nom_colonne3 **TYPE**
)

CREATE TABLE *nom_table* **(**
nom_colonne1 **TYPE,**
nom_colonne2 **TYPE TYPE CONTRAINTE,**
nom_colonne3 **TYPE**
)

Contraintes : **NOT NULL**

```
CREATE TABLE student (
  student_id int NOT NULL,
  first_name varchar(50),
  last_name varchar(50) NOT NULL,
  birth_date datetime,
  login varchar(50),
  section_id int,
  year_result int,
  course_id varchar(6) NOT NULL
)
```

La contrainte NOT NULL *est la seule contrainte qui* ***n'est pas nommée****,*
elle est incluse dans la définition de la colonne

Contraintes : **UNIQUE**

```
CREATE TABLE student (
  student_id int NOT NULL UNIQUE,
  first_name varchar(50),
  last_name varchar(50) NOT NULL,
  birth_date datetime,
  login varchar(50) CONSTRAINT UK_login UNIQUE,
  section_id int,
  year_result int,
  course_id varchar(6) NOT NULL,
  CONSTRAINT UK_Lname_Bdate UNIQUE (last_name, birth_date)
)
```

Nom contrainte — Type contrainte — Combinaison de colonnes concernées

Toute contrainte aura un nom *dans le système. Si nous ne la nommons pas nous-même, le système choisira lui-même un nom pour la contrainte*

Contraintes : **PRIMARY KEY**

```
CREATE TABLE student (
    student_id int,
    first_name varchar(50),
    last_name varchar(50),
    birth_date datetime,
    login varchar(50),
    section_id int,
    year_result int,
    course_id varchar(6),
    CONSTRAINT PK_student PRIMARY KEY (student_id)
)
```

Contraintes : **PRIMARY KEY**

```
CREATE TABLE student (
    student_id int PRIMARY KEY,
    first_name varchar(50),
    last_name varchar(50),
    birth_date datetime,
    login varchar(50),
    section_id int,
    year_result int,
    course_id varchar(6)
)
```

Contraintes : **FOREIGN KEY**

CREATE TABLE *nom_table* **(**
nom_colonne1 **TYPE,**
nom_colonne2 **TYPE,**
CONSTRAINT *nom_contrainte* **TYPE CONTRAINTE (***colonne_concernée***)**
REFERENCES *table_référencée* **(***colonne_référencée***)**
)

CREATE TABLE *nom_table* **(**
nom_colonne1 **TYPE,**
nom_colonne2 **TYPE REFERENCES** *table_référencée*,
nom_colonne3 **TYPE**
)

Le paramètre ***« colonne_référencée »*** *peut être omis*
si les colonnes portent le même nom dans la table enfant et la table parent

Contraintes : FOREIGN KEY

```
CREATE TABLE student (
    student_id int,
    first_name varchar(50),
    last_name varchar(50),
    birth_date datetime,
    login varchar(50),
    section_id int,
    year_result int,
    course_id varchar(6),
    CONSTRAINT FK_student_section FOREIGN KEY (section_id)
        REFERENCES section (section_id)
)
```

Contraintes : FOREIGN KEY

```
CREATE TABLE student (
    student_id int,
    first_name varchar(50),
    last_name varchar(50),
    birth_date datetime,
    login varchar(50),
    section_id int
        CONSTRAINT FK_student_section
        REFERENCES section (section_id),
    year_result int,
    course_id varchar(6)
)
```

Contraintes : FOREIGN KEY

```
CREATE TABLE student (
    student_id int,
    first_name varchar(50),
    last_name varchar(50),
    birth_date datetime,
    login varchar(50),
    section_id int REFERENCES section,
    year_result int,
    course_id varchar(6)
)
```

Contraintes : FOREIGN KEY

Les options ***« ON DELETE »*** et ***« ON UPDATE »*** peuvent être rajoutées à la contrainte de clé étrangère de façon à ne pas lever d'erreur lorsqu'une ligne de la table référencée est supprimée ou mise à jour

Dans ce cas, selon l'action spécifiée, la ligne de la table enfant sera supprimée ou modifiée

- L'action ***« CASCADE »*** répercute la modification (ou suppression de ligne) sur la table enfant
- L'action ***« SET NULL »*** met à NULL les valeurs correspondantes dans la table enfant (possible uniquement si la colonne accepte les valeurs NULL…)
- L'action ***« SET DEFAULT »*** remplace les valeurs correspondantes dans la table enfant par la valeur prévue par défaut (si aucune valeur par défaut n'a été renseignée, la valeur sera NULL)
- Si rien n'est spécifié, l'action par défaut est ***« NO ACTION »*** qui lève une erreur

```
CREATE TABLE nom_table (
nom_colonne1 TYPE,
nom_colonne2 TYPE,
CONSTRAINT nom_contrainte TYPE CONTRAINTE (colonne_concernée)
    REFERENCES table_référencée (colonne_référencée)
    ON DELETE action ON UPDATE action
)
```

Contraintes : FOREIGN KEY

```
CREATE TABLE student (
    student_id int PRIMARY KEY,
    first_name varchar(50),
    last_name varchar(50),
    birth_date datetime,
    login varchar (50),
    section_id int
    CONSTRAINT FK_student_section REFERENCES section
       ON DELETE SET NULL ON UPDATE CASCADE,
    year_result int,
    course_id varchar(6)
)
```

Contraintes : **CHECK**

```
CREATE TABLE student (
    student_id int PRIMARY KEY,
    first_name varchar(50),
    last_name varchar(50)
    CONSTRAINT CK_last_name CHECK (last_name IS NOT NULL),
    birth_date datetime,
    login varchar (50),
    section_id int,
    year_result int CHECK (year_result BETWEEN 0 AND 20),
    course_id varchar(6),
    CONSTRAINT CK_birth_date CHECK (YEAR(birth_date) > 1970)
)
```

ALTER TABLE

L'ordre DDL « ALTER TABLE » permet de modifier la structure d'une table déjà existante. Certaines actions nécessiteront parfois de lourdes procédures, notamment lorsque l'on souhaite changer le type d'une colonne contenant déjà des données. *C'est pourquoi il est très important d'avoir réaliser une bonne analyse au préalable…*

Ajouter/modifier une colonne

ALTER TABLE *nom_table* **ADD** *nom_colonne* **TYPE** *contraintes*
ALTER TABLE *nom_table* **ALTER COLUMN** *nom_colonne* **NOUVEAU_TYPE …**

```
ALTER TABLE student
ADD year_result int CHECK (year_result BETWEEN 0 AND 20)
```

```
ALTER TABLE student ALTER COLUMN birth_date date
```

ALTER TABLE

Ajouter une contrainte

ALTER TABLE *nom_table* **ADD CONSTRAINT** *nom_contrainte* **TYPE (***colonne***) …**

```
ALTER TABLE student
ADD CONSTRAINT FK_student_section FOREIGN KEY (section_id)
    REFERENCES section ON DELETE SET NULL
```

Désactiver/réactiver une contrainte

ALTER TABLE *nom_table* **NOCHECK CONSTRAINT** *nom_contrainte*
ALTER TABLE *nom_table* **CHECK CONSTRAINT** *nom_contrainte*

```
ALTER TABLE student NOCHECK CONSTRAINT FK_student_section
```

ALTER TABLE

Supprimer une colonne

ALTER TABLE *nom_table* **DROP COLUMN** *nom_colonne*

```
ALTER TABLE student DROP COLUMN year_result
```

Supprimer une contrainte

ALTER TABLE *nom_table* **DROP CONSTRAINT** *nom_contrainte*

```
ALTER TABLE student DROP CONSTRAINT PK_student
```

TRUNCATE TABLE

L'ordre DDL « TRUNCATE TABLE » permet de vider une table de son contenu. Cet ordre permet par la même occasion de réinitialiser le conteur utiliser pour la colonne éventuellement auto-incrémentée

Cette opération est ***plus rapide et efficace qu'un DELETE*** s'il s'agit de supprimer l'ensemble des données d'une table car l'ordre DELETE va créer une entrée par ligne supprimée au niveau du journal de transactions

TRUNCATE TABLE *nom_table*

```
TRUNCATE TABLE student
```

DROP TABLE

L'ordre DDL « DROP TABLE » permet de supprimer une table. Attention aux contraintes de clés étrangères…

DROP TABLE *nom_table*

```
DROP TABLE student
```

Auto-Evaluation

Ce premier module contenait une série de notions importantes, autant théoriques que pratiques. Nous vous invitons (suite à la réalisation des exercices) à évaluer le niveau de maîtrise que vous estimez avoir acquis personnellement concernant ces notions

Rappel de la signification des lettres dans les tableaux d'auto-évaluation :

- **Parfait (P) :** vous avez parfaitement compris cette notion et vous vous sentez à votre aise
- **Satisfaisant (S) :** vous avez compris de quoi il s'agit mais la pratique vous manque
- **Vague (V) :** vous savez de quoi il s'agit, mais cela reste un peu vague dans votre esprit. Une explication supplémentaire du formateur ou une bonne révision de votre part s'impose
- **Insatisfaisant (I) :** Vous n'avez pas du tout compris la notion abordée, il faut tout faire pour y remédier !

Auto-Evaluation

Notions à évaluer (1/2)

Notions	P	S	V	I
Base de données ***(en théorie)***				
Système de gestion de bases de données ***(en théorie)***				
Cycle d'analyse d'un projet ***(en théorie)***				
Schéma Entité-Association ***(en théorie)***				
Transformation du schéma EA vers les schéma Relationnel				
Schéma Relationnel (utilité, création, lecture)				
Structure d'une table ***(en théorie)***				
Notion de « contrainte » de façon générale ***(en théorie)***				
Ordre « CREATE TABLE »				

Auto-Evaluation

Notions à évaluer (2/2)

Notions	P	S	V	I
Contrainte de « PRIMARY KEY »				
Contrainte de « FOREIGN KEY »				
Contraintes « UNIQUE », « NOT NULL » et « CHECK »				
Auto-incrémentation				
Valeur par défaut				
Ordre « ALTER TABLE »				
Ordre « DROP TABLE »				
Ordre « TRUNCATE TABLE »				