

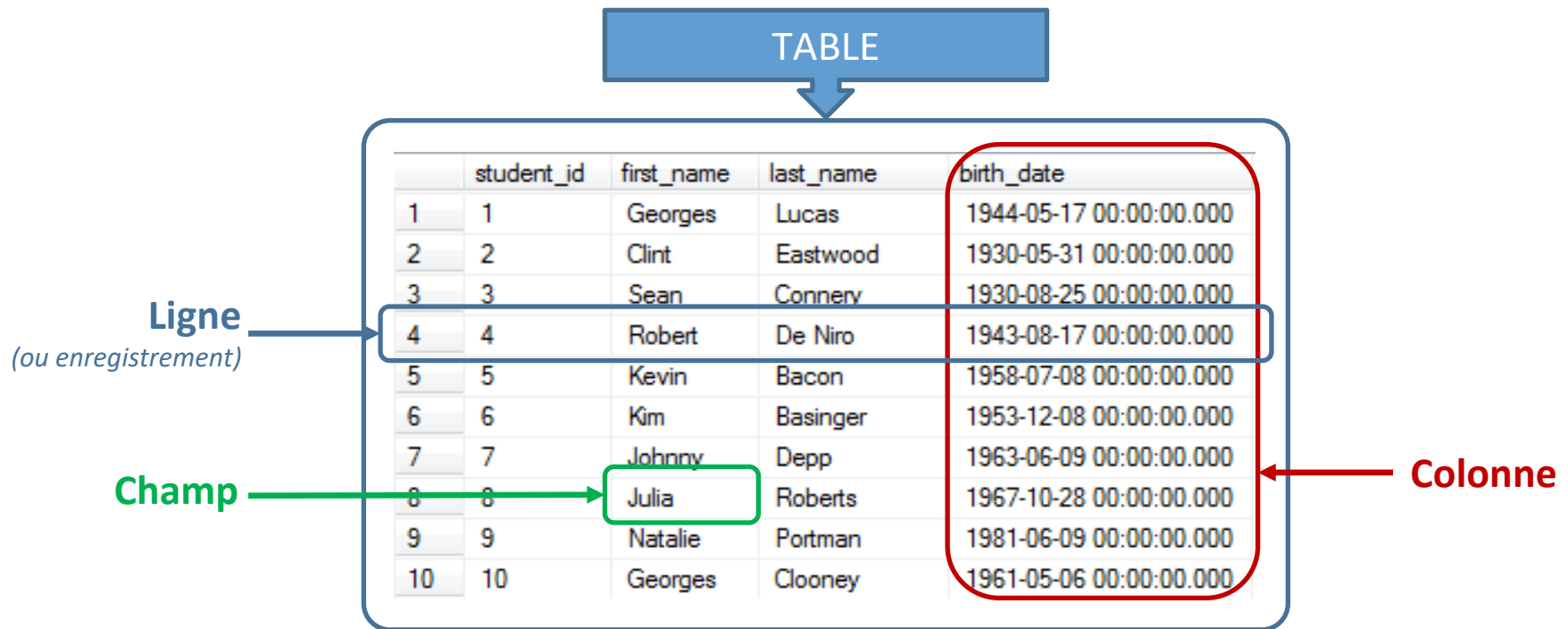
# SQL déclaratif

## du standard à la pratique

Langage d'interrogation des Bases de  
Données

# Notions de tables

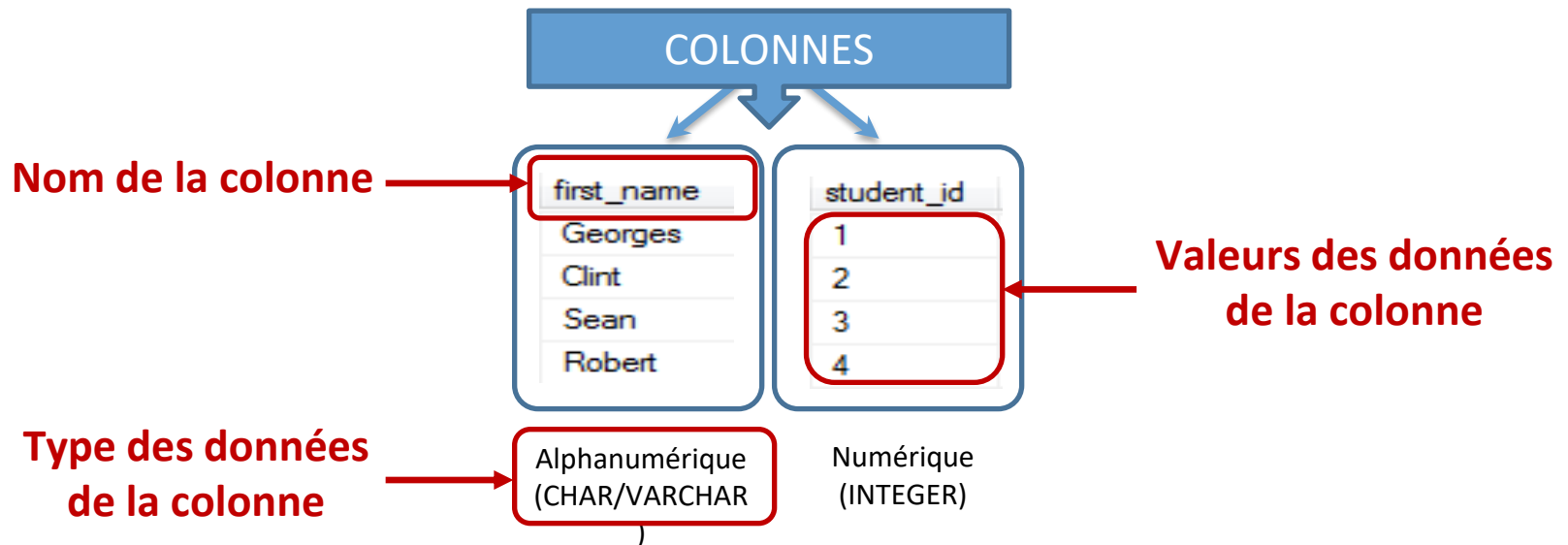
Une **Table** regroupe des **ensembles de données (d'informations)** stockés de façon permanente et décrivant chacun un acteur du système réel modélisé (**Entité**)



*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
<b>NOT NULL</b>	Force la présence d'une valeur (valeur obligatoire)
<b>UNIQUE</b>	Empêche les valeurs-doublons
<b>CHECK</b>	Conditionne les valeurs (expression conditionnelle)
<b>FOREIGN KEY</b>	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 qu'aucune valeur n'est attribuée

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

The diagram shows a table with 9 columns: student\_id, first\_name, last\_name, birth\_date, login, section\_id, year\_result, and course\_id. Annotations include: 'Pas de valeur renseignée' pointing to the first\_name column; 'Colonne facultative acceptant des valeurs NULL' pointing to the year\_result column, which contains a NULL value; and 'Colonne obligatoire NULL' pointing to the course\_id column, which contains 0 values.

	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 le 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 étrangères 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

Lien

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

# Contraintes : FOREIGN KEY

CREATE  
INSERT

Colonne appartenant à la **clé étrangère**

STUDENT			
<u>student_id</u>	..	section_id	..
1	..	1320	..
2	..	1010	..
3	..	1020	..
..	..	..	..
13	..	1020	..

Table **Enfant** ou Esclave

SECTION	
<u>section_id</u>	section_nom
1010	BAC1 GESTION
1020	MASTER1 GESTION
1110	BAC1 ECONOMIE
1120	MASTER1 ECONOMIE
1310	BAC SOCIOLOGIE
1320	MASTER SOCIOLOGIE

Table **Parent** ou Maître

Colonne appartenant à la **clé primaire**

DROP  
DELETE

# Contraintes : CHECK

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

	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

$0 \leq \text{year\_result} \leq 20$

$\text{Birth\_date} > 01-01-1930$

## 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 définit le type qu'auront les valeurs de cette colonne. Les types principaux sont **INT** (pour entiers), **VARCHAR(taille)** (pour une chaîne 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 ( <i>INT</i> )	4	Valeurs entières de $-2^{31}$ à $2^{31}$
DECIMAL( <i>x,y</i> ) ( <i>DEC</i> )	5 à 17	Valeurs numériques contenant « x » chiffres au total, dont « y » après la virgule
VARCHAR( <i>x</i> )	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( <i>x</i> )	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) ( $1 \leq x \leq 53$ )	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 <b>DÉPRÉCIÉ</b>	2 <sup>31</sup> -1 (2 147 483 647)	Chaîne de caractères d'une taille variable pouvant aller jusqu'à 2 <sup>31</sup> -1 caractères (non-Unicode)
NTEXT <b>DÉPRÉCIÉ</b>	Nombre de caractères * 2	Chaîne de caractères d'une taille variable pouvant aller jusqu'à 2 <sup>30</sup> -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 <i>(DÉPRÉCIÉ, utiliser « ROWVERSION »)</i>	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>

# 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 sont 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

# 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)  
)
```

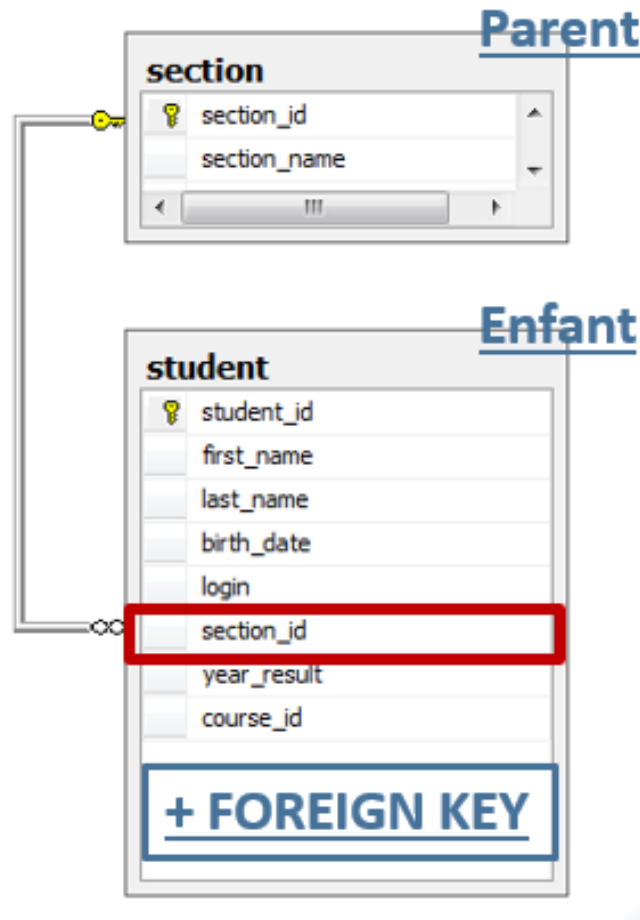
# Contraintes : **FOREIGN KEY**

```
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**



# 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)  
)
```

Première notation possible

# 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)  
)
```

Deuxième notation possible

# 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)  
)
```

Troisième notation possible



# 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 CONTRAINT (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 réalisé 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 compteur utilisé 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
```



# Exercices

# 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
Transformation du schéma EA vers les schéma Relationnel				
Schéma Relationnel (utilité, création, lecture)				
Structure d'une table ( <i>en théorie</i> )				
Notion de « contrainte » de façon générale ( <i>en théorie</i> )				
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 »				