Bases de Données Relationnelles

Langage de Définition de Données (LDD)

L2 MIASH

Rafael Angarita
Maitre de Conférences
rangarit@parisnanterre.fr

Université Paris Nanterre

Université Paris Nanterre
UFR SEGMI

SQL: Structured Query Language

SQL : Un standard

- Rétrocompatible
- ANSI/ISO
  - SQL-86 – intersection of IBM SQL implementations
  - SQL-89 – small revision, integrity constraints
  - SQL-92 – schema modification, transactions, set operators, new data types, cursors, referential integrity actions, …
  - SQL:1999 – recursive queries, triggers, object-relational features, regular expressions, types for full-text, images, spatial data, …
  - SQL:2003 – SQL/XML, sequence generators
  - SQL:2006 – other extensions of XML, integration of XQuery
  - SQL:2008
  - SQL:2011 – temporal databases

3

Catégories des instructions SQL

- **Définition** des éléments d'une base de données
  - *Data Definition Language*, soit DDL
- **Manipulation** des données
  - *Data Manipulation Language*, soit DML
- **Gestion des droits d'accès** aux données
  - *Data Control Language*, soit DCL
- **Gestion des transactions**
  - *Transaction Control Language*, soit TCL
- **SQL intégré**
  - *Embedded SQL*

SQL : Langage de définition de données

- **CREATE**
- **ALTER**
- **DROP**
- Et d'autres instructions :
  - AUDIT, NOAUDIT, ANALYZE, RENAME, TRUNCATE

SQL : Langage de manipulation de données

- **INSERT**
- **UPDATE**
- **DELETE**
- **SELECT**
- Et d'autres instructions :
  - EXPLAIN, PLAN, LOCK TABLE

Langage de Définition de Données (LDD)

CREATE : **Création d'une BD**

https://dev.mysql.com/doc/refman/8.0/en/charset-database.html

CREATE DATABASE [IF NOT EXISTS] database_name
[**CHARACTER SET** charset_name]
[**COLLATE** collation_name]

Charset et Collation

http://www.dynamic-mess.com/sql/comprendre-charset-et-collation/

- **Jeux de caractères (Character set)** : Un ensemble de symboles et d'encodage
- **Interclassement (Collation)** : Un ensemble de règles comparant les caractères dans un jeu de caractères
- Un jeu peut avoir plusieurs interclassements, un par langue généralement.
- Cela permet par ex de classer par ordre alphabétique les caractères => le SGBD saura interpréter des requêtes de tri, mais également savoir si un caractère est équivalent à un autre, par exemple le 'a' et le 'à'.

Tapez la commande SQL suivante pour connaitre tous les jeux de caractères de Mysql avec leur interclassement par défaut:

SHOW CHARACTER SET;

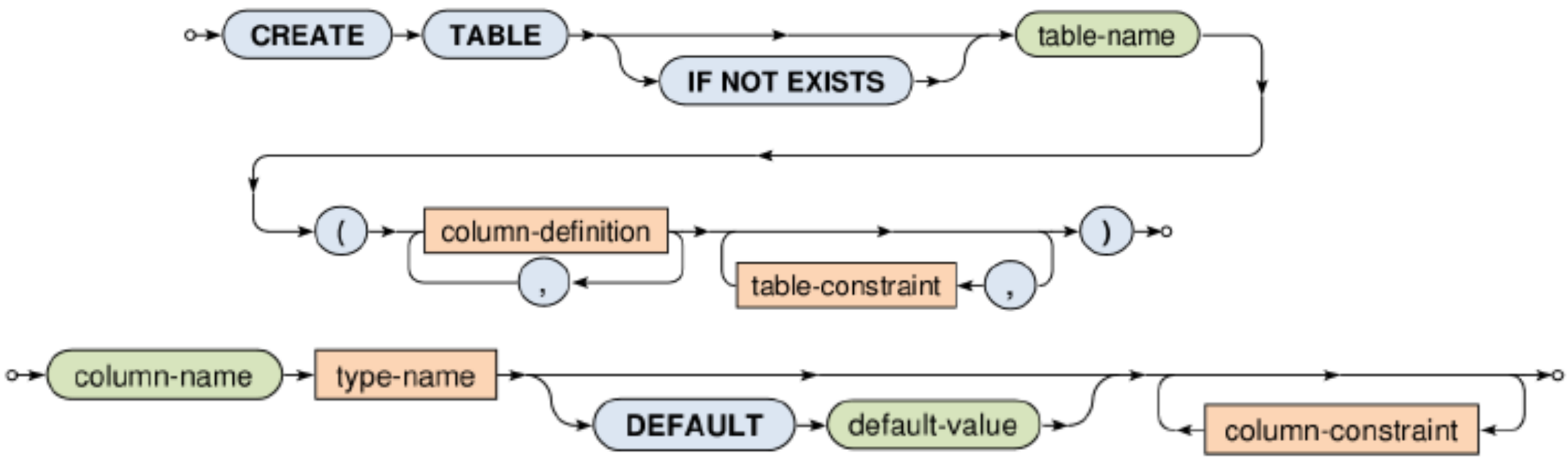
CREATE : Suffixes de l'interclassement

_ai	Accent insensitive
_as	Accent sensitive
_ci	Case insensitive
_cs	Case sensitive
_ks	Kana sensitive
_bin	Binary

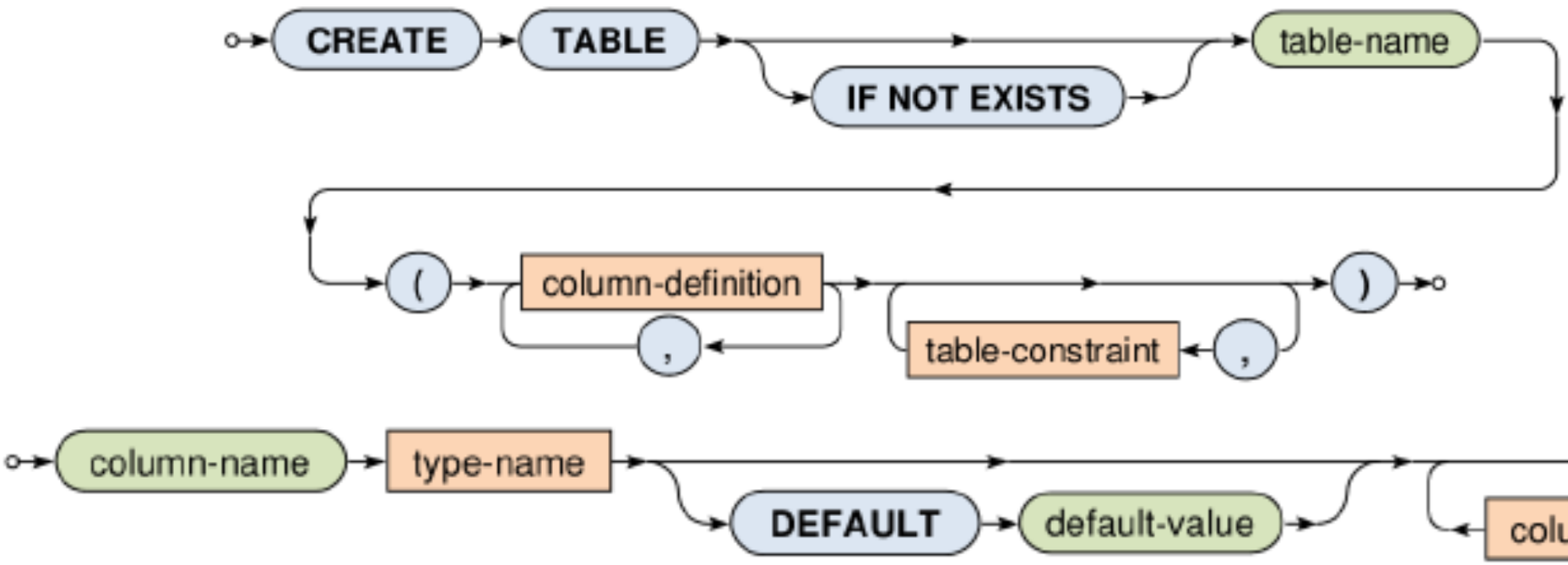
- latin1_general_ci est explicitement case insensitive et implicitement accent insensitive
- latin1_general_cs est explicitement case sensitive et implicitement accent sensitive
- utf8mb4_0900_ai_ci est explicitement case and accent insensitive.

CREATE : **Création** simple d'une **table** et des **colonnes associées**

- Construction d'une table
  - Nom de la table
  - Définition de chaque colonne
    - Nom de la colonne
    - Type de données de la colonne
    - Valeur par défaut



CREATE : **Création** simple d'une **table** et des **colonnes associées**



CREATE TABLE nom_de_la_table **(**
colonne1 type_donnees,
colonne2 type_donnees,
colonne3 type_donnees,
colonne4 type_donnees **)**

CREATE : Types de données

- **INTEGER** : Ce type permet de stocker des entiers signés codés sur 4 octets
- **VARCHAR(longueur)** : Ce type de données permet de stocker des chaînes de caractères de longueur variable. longueur doit être inférieur à 2000, il n'y a pas de valeur par défaut
- **DATE** : Ce type de données permet de stocker des données constituées d'une date
- **TIMESTAMP** : Ce type de données permet de stocker des données constituées d'une date et d'une heure
- **BOOLEAN** : Ce type de données permet de stocker des valeurs Booléenne
- **CHAR(longueur)** : Ce type de données permet de stocker des chaînes de caractères de longueur fixe. longueur doit être inférieur à 255, sa valeur par défaut est 1

CREATE : Types de données (suite)

- BIGINT : Ce type permet de stocker des entiers signés codés sur 8 octets.
- REAL : Ce type permet de stocker des réels comportant 6 chiffres significatifs codés sur 4 octets.
- DOUBLE PRÉCISION : Ce type permet de stocker des réels comportant 15 chiffres significatifs codés sur 8 octets.
- NUMERIC[(précision, [longueur])] : Ce type de données permet de stocker des données numériques à la fois entières et réelles avec une précision de 1000 chiffres significatifs.
  - longueur précise le nombre maximum de chiffres significatifs stockés et précision donne le nombre maximum de chiffres après la virgule.
- MONEY : Ce type de données permet de stocker des valeurs monétaires.
- TEXT : Ce type de données permet de stocker des chaînes de caractères de longueur variable.

Types de données numériques exacts sur MySQL

Type	Storage (Bytes)	Minimum Value Signed	Minimum Value Unsigned	Maximum Value Signed	Maximum Value Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2^{63}	0	$2^{63}-1$	$2^{64}-1$

CREATE: exemple sans contraintes

Product*(id, name, price, produced, available, weight)*

```
CREATE TABLE Product (
      id INTEGER,
      name VARCHAR(128),
      price DECIMAL(6,2),
      produced DATE,
      available BOOLEAN DEFAULT TRUE,
      weight FLOAT
);
```

CREATE : Création avec contraintes d'intégrité

- Construction d'une table
  - Nom de la table
  - Définition de chaque colonne
    - Nom de la colonne
    - Type de données de la colonne
    - Valeur par défaut
    - **Contraintes d'intégrité des colonnes**
- **Contraintes d'intégrité de la table**

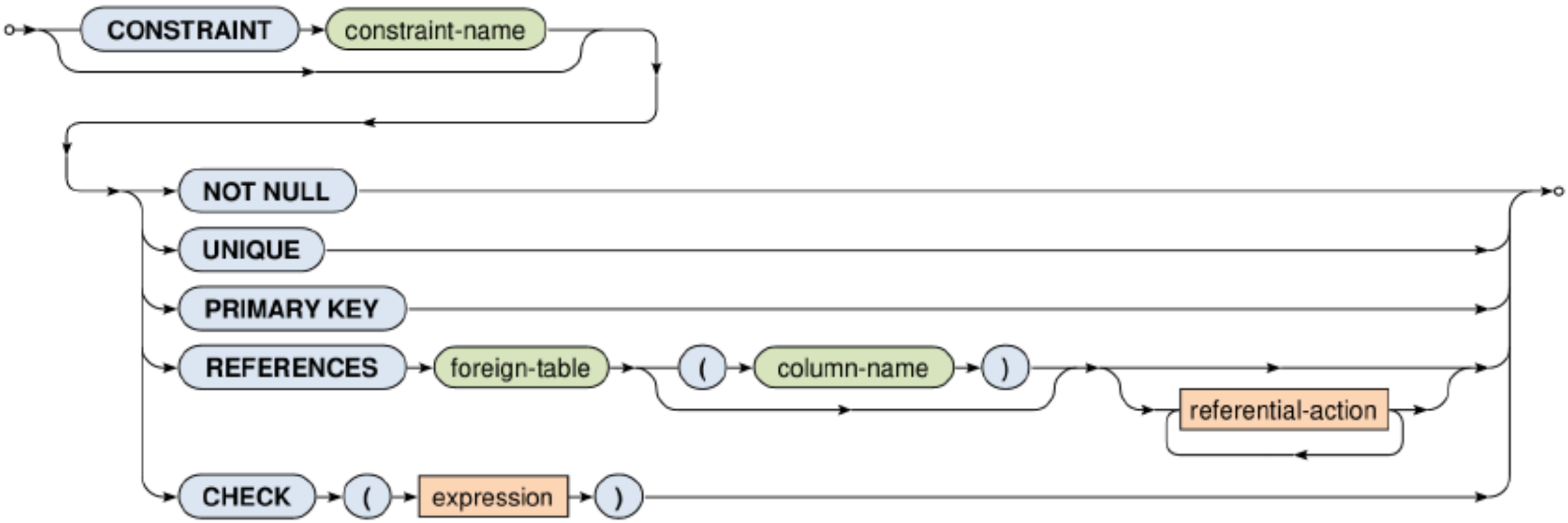
CREATE : Contraintes d'intégrité de **colonne**

NOT NULL	Empêche d'enregistrer une valeur nulle pour une colonne
NULL	Autorise d'enregistrer une valeur nulle pour une colonne
UNIQUE	Désigne l'attribut comme clé secondaire de la table. Deux n-uplets ne peuvent recevoir des valeurs identiques pour cette colonne, mais l'insertion de valeur NULL est toutefois autorisée
DEFAULT valeur	Attribuer une valeur par défaut si aucune données n'est indiquée pour cette colonne lors de l'ajout d'une ligne dans la table
PRIMARY KEY	Désigne l'attribut comme clé primaire de la table. Equivalente à la contrainte UNIQUE NOT NULL

CREATE : Contraintes d'intégrité de **colonne** (suite)

FOREIGN KEY [colonne] REFERENCES table [(colonne)] *[ON DELETE CASCADE]*	Contrainte d'intégrité référentielle pour l'attribut de la table en cours de définition. Les valeurs prises par cet attribut doivent exister dans la colonne de la table étrangère 'table', cette colonne possède une contrainte PRIMARY KEY ou UNIQUE En l'absence de précision d'attribut colonne, l'attribut retenu est celui correspondant à la clé primaire de la table 'table'.
CHECK (condition)	Vérifie lors de l'insertion de n-uplets que l'attribut réalise la condition condition.

CREATE : Contraintes d'intégrité de **colonne** (résumé)



CREATE : Contraintes d'intégrité de **colonne (exemples)**

Person(<u>id</u>, personalNumber, address, age, serialNumber, color)

```
CREATE TABLE Person(
      id INTEGER PRIMARY KEY,
      personalNumber INTEGER,
      adress VARCHAR(256),
      age INTEGER,
      serialNumber INTEGER NOT NULL,
      color STRING
);
```

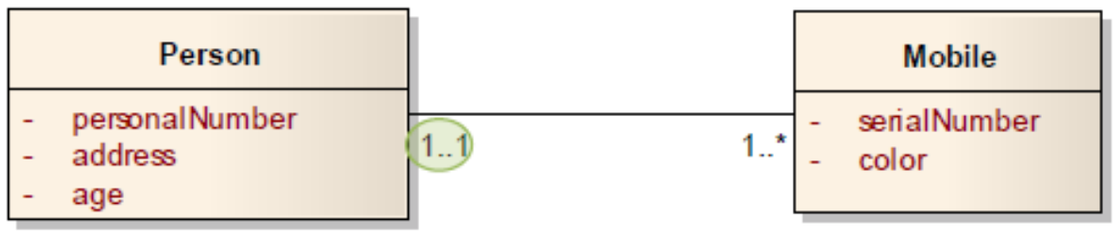
CREATE : Contraintes d'intégrité de **colonne (exemples)**

```
CREATE TABLE Producer (
   id INTEGER PRIMARY KEY,
   name VARCHAR(128),
   country VARCHAR(64)
);
```

Nomme une contrainte, **pas obligatoire** dans le cas des contraintes de **colonne**

```
CREATE TABLE Product (
   id INTEGER CONSTRAINT IC_Product_PK PRIMARY KEY,
   name VARCHAR(128) UNIQUE,
   price DECIMAL(6,2) CONSTRAINT IC_Product_Price NOT NULL,
   produced DATE CHECK (produced >= '2015-01-01'),
   available BOOLEAN DEFAULT TRUE NOT NULL,
   weight FLOAT,
   producer INTEGER,
   FOREIGN KEY(producer) REFERENCES Producer(id)
);
```

CREATE : Contraintes d'intégrité de **colonne (exemples)**



Person(<u>personalNumber</u>, address, age)
Mobile(<u>serialNumber</u>, color, personalNumber)
Mobile.personalNumber ⊆ Person.personalNumber

```
CREATE TABLE Person(
   personalNumber INTEGER PRIMARY KEY,
   adress VARCHAR(256),
   age INTEGER,
);
CREATE TABLE Mobile(
   serialNumber INTEGER PRIMARY KEY,
   color STRING,
   personalNumber INTEGER,
   FOREIGN KEY(personalNumber) REFERENCES Person(personalNumber)
);
```

CREATE : Contraintes d'intégrité de **table**

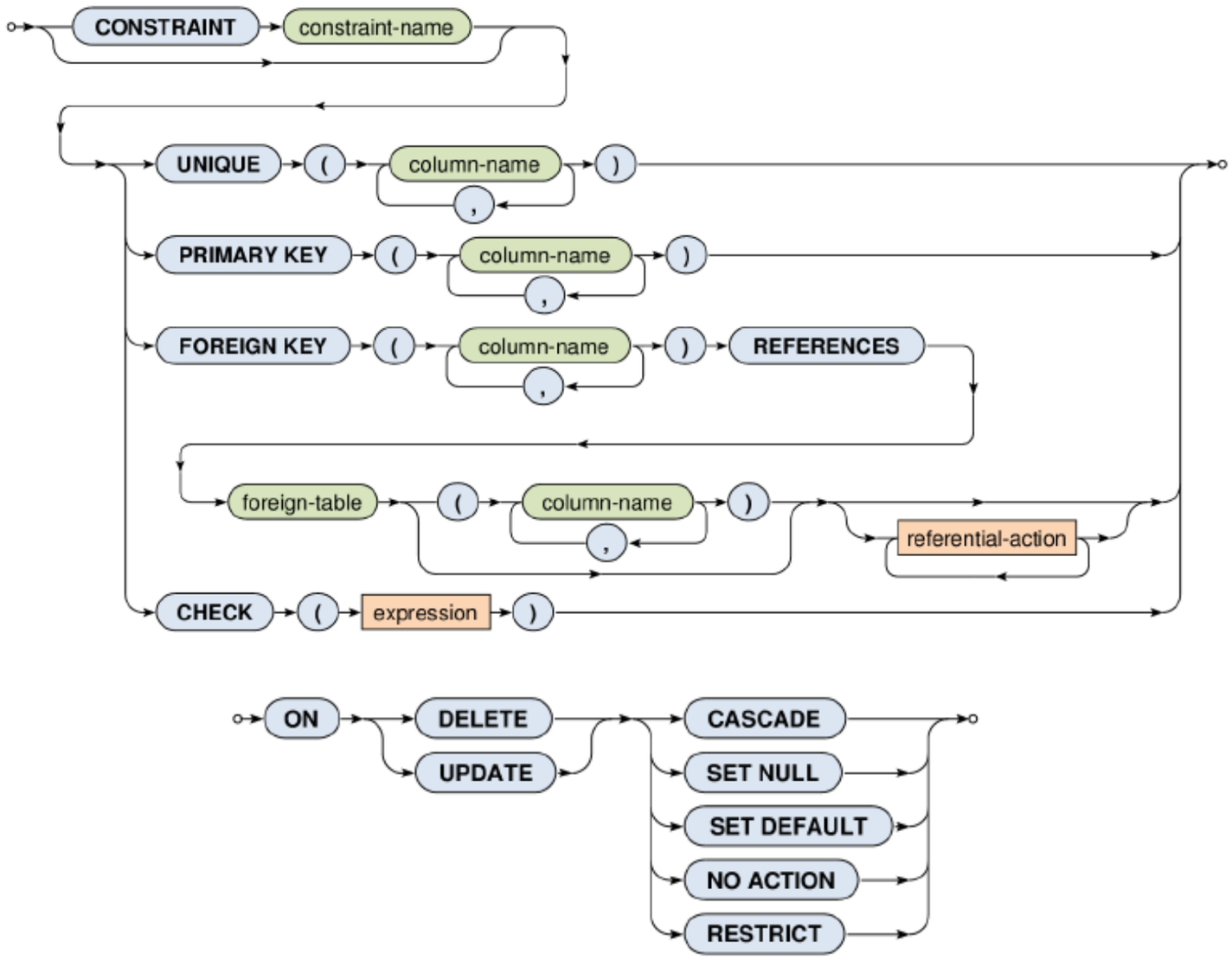
Analogue aux contraintes des colonnes mais pour les colonnes multiples

NOT NULL	Empêche d'enregistrer une valeur nulle pour une colonne
NULL	Autorise d'enregistrer une valeur nulle pour une colonne
UNIQUE	Désigne l'attribut comme clé secondaire de la table. Deux n-uplets ne peuvent recevoir des valeurs identiques pour cette colonne, mais l'insertion de valeur NULL est toutefois autorisée
DEFAULT valeur	Attribuer une valeur par défaut si aucune données n'est indiquée pour cette colonne lors de l'ajout d'une ligne dans la table
PRIMARY KEY	Désigne l'attribut comme clé primaire de la table. Equivalente à la contrainte UNIQUE NOT NULL

CREATE : Contraintes d'intégrité de **table** (suite)

FOREIGN KEY (colonne…) REFERENCES table [(colonne)]* *[ON DELETE CASCADE]*	Contrainte d'intégrité référentielle pour l'attribut de la table en cours de définition. Les valeurs prises par cet attribut doivent exister dans l'attribut colonne qui possède une contrainte PRIMARY KEY ou UNIQUE dans la table *'table'*. En l'absence de précision d'attribut colonne, l'attribut retenu est celui correspondant à la clé primaire de la table table spécifiée.
CHECK (condition)	Vérifie lors de l'insertion de n-uplets que l'attribut réalise la condition condition.

CREATE : Contraintes d'intégrité de **table** (résumé)



CREATE : Contraintes d'intégrité de **table (exemples)**

Producer(<u>name, country</u>)
Product(<u>id</u>,.., producerName, producerCountry)
Product(producerName, producercountry) ⊆ Producer(name, country)

```
CREATE TABLE Producer (
  name VARCHAR(128),
  country VARCHAR(3),
  CONSTRAINT IC_Producer_PK PRIMARY KEY (name,   country)
);
```

```
CREATE TABLE Product (
  id INTEGER PRIMARY KEY,
  …
  producerName VARCHAR(128),
  producerCountry VARCHAR(3),
  CONSTRAINT  IC_Product_Producer_FK
       FOREIGN KEY (producerName,
                    producerCountry)
       REFERENCES Producer (name,
       country)
);
```

CREATE : Contraintes d'intégrité de **table (exemples)**

Person(<u>personalNumber</u>, address, age)
Mobile(<u>serialNumber</u>, color)
Ownership(<u>personalNumber, serialNumber</u>)
Ownership.personalNumber ⊆ Person.personalNumber
Ownership.serialNumber ⊆ Mobile.serialNumber

```
CREATE TABLE Person(
  personalNumber INTEGER PRIMARY KEY,
  adress VARCHAR(256),
  age INTEGER,
);
CREATE TABLE Mobile(
  serialNumber INTEGER PRIMARY KEY,
  color STRING,
);
```

```
CREATE TABLE Ownership(
  personalNumber INTEGER NOT NULL
    REFERENCES Person(personalNumber),
  serialNumber INTEGER NOT NULL
    REFERENCES Mobile(serialNumber),
  CONSTRAINT
    PRIMARY KEY(personalNumber,serialNumber)
);
```

Intégrité référentielle

Primary Table

CompanyId	CompanyName
1	Apple
2	Samsung

~~15 .....~~ ?

Related Table

CompanyId	ProductId	ProductName
1	1	iPhone
15	2	Mustang

Associated Record ✓

Orphaned Record ✗

CREATE: intégrité référentielle

- l'**intégrité référentielle** est une **contrainte que le concepteur de bases de données s'impose afin de garantir l'intégrité des données** stockées.

- Vérifier qu'une valeur de clé étrangère existe bien en tant que valeur de clé primaire dans une autre table.

- Exemple : Des factures sont reliées à un client. L'intégrité référentielle empêchera la suppression d'un client si des factures sont déjà stockées dans la base de données. Aussi, si un client est supprimé alors toutes ses factures seront supprimées également

CREATE: intégrité référentielle (exemple)

```
CREATE TABLE Producer (
   id INTEGER PRIMARY KEY,
   name VARCHAR(128),
   country VARCHAR(64)
);
```

```
CREATE TABLE Product (
   id INTEGER PRIMARY KEY,
   …
   producer INTEGER
      REFERENCES Producer (id) ON DELETE CASCADE
);
```

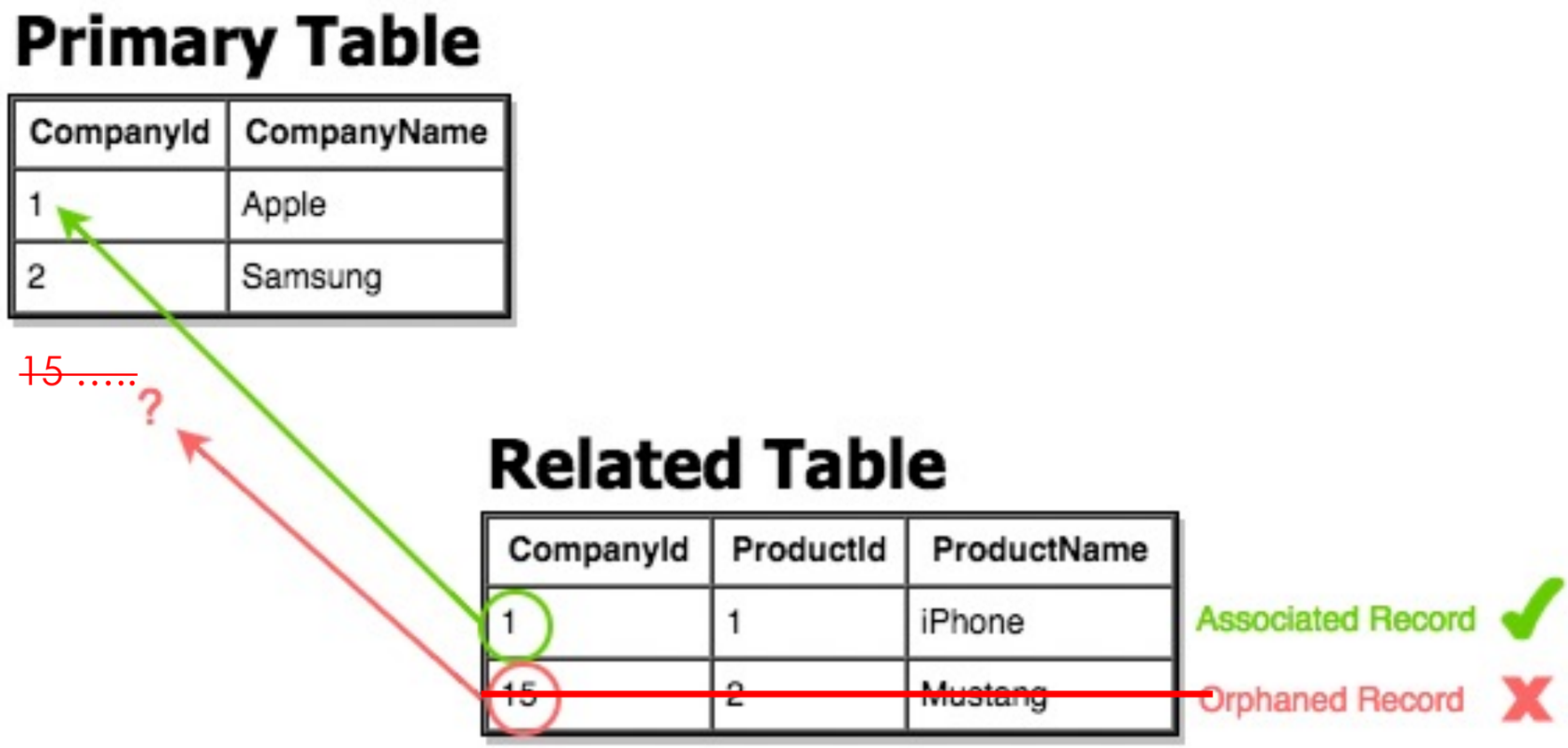
Si un Producteur est supprimé Alors toutes les lignes (tuples) de la table Product où ce producteur apparait sont supprimées

CREATE: intégrité référentielle

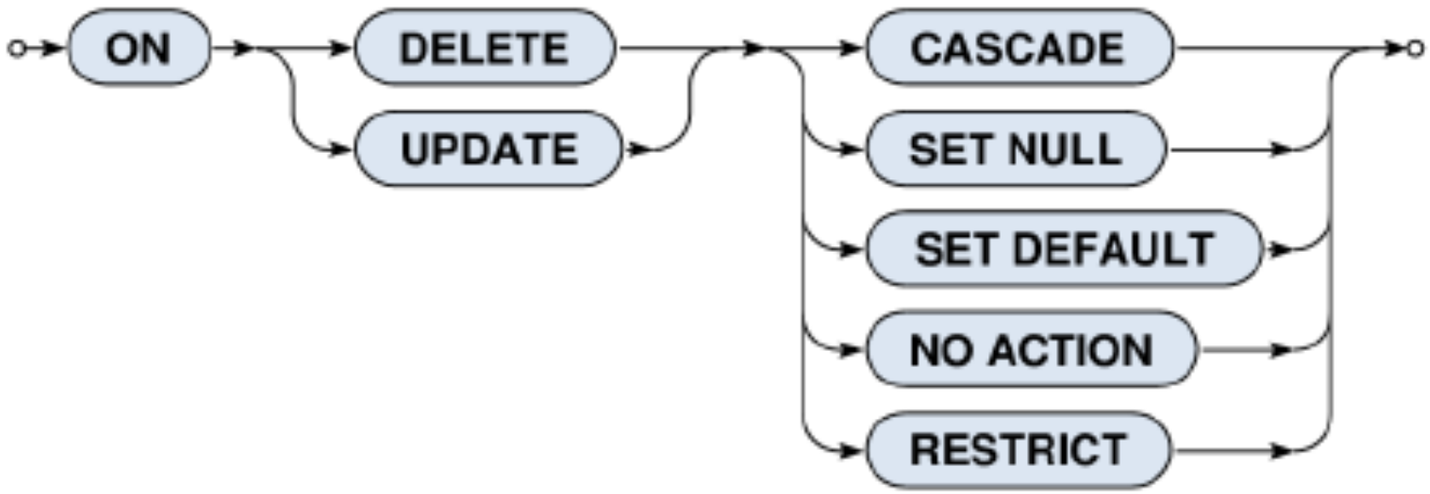
- Situations de déclenchement
  - **ON UPDATE, ON DELETE**
    - Lorsque l'action est déclenchée
    - Encore une fois, celles-ci sont considérées comme des opérations sur la table référencée
- Actions référentielles
  - **CASCADE**
    - La ligne avec la valeur de référence est également mise à jour/supprimée
  - **SET NULL** – la valeur de référence est définie sur NULL
  - **SET DEFAULT** – la valeur de référence est définie sur sa valeur par défaut
  - **NO ACTION** – par défaut - aucune action n'a lieu

Intégrité référentielle

Avec **ON DELETE**

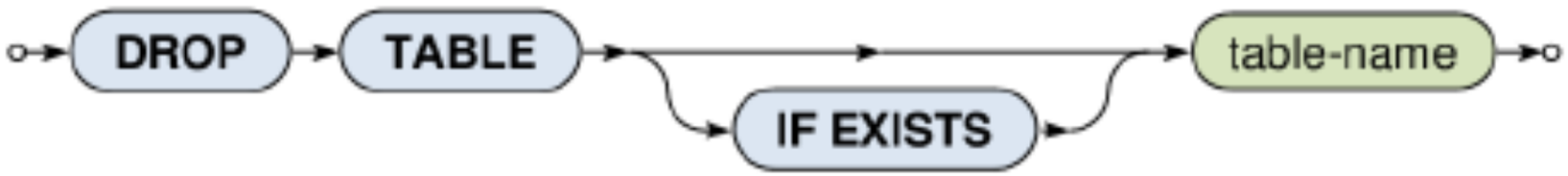


CREATE: intégrité référentielle (résumé)



DROP

- Complémentaire à la création de table
  - La définition de la table ainsi que le contenu de la table sont supprimés



ALTER

- Ajout/modification/suppression de colonnes de tableau/contrainte d'intégrité