Term NSI **SQL et contraintes d'intégrité**

1) mémento de syntaxe SQL.
SQL : *Structured Query Language* = langage de requête structuré
Les requêtes sont exécutées par un **SGBD** : Système de Gestion de Bases de Donnée

Sélection
Sélection de certains champs d'une table :
```
SELECT champ1, champ2... FROM table  WHERE condition;
```
Sélection de tous les champs d'une table
```
SELECT  *  FROM table  WHERE condition;
```
Mot clé `DISTINCT` pour n'afficher que des réponses distinctes :
```
SELECT DISTINCT champ1, champ2...
```
Ajout en fin de requête, d'une critère de classement : `ORDER BY`
en précisant éventuellement `DESC` ou `ASC` (par défaut)

Jointure
```
SELECT * FROM table1 JOIN table2 ON table1.champ = table2.champ;
```

Mise à jour de certaines lignes
```
UPDATE nom_de_la_table
SET nom_de_colonne_modifiée1 = valeur1 , nom_de_colonne_modifiée2 = valeur2
WHERE condition;
```

Insertion
```
INSERT INTO nom_de_la_table(colonne1, colonne2...) VALUES (valeur1, valeur2...);
```

Suppression
```
DELETE FROM nom_de_la_table WHERE condition;
```

2) Comment SQL permet de vérifier certaines **contraintes d'intégrité**

a) contraintes de domaine
Chaque champ d'une table possède un domaine : en cas d'insertion ou de mise à jour, le SGBD vérifie que les données à écrire dans la base sont compatibles avec le domaine du champ correspondant.

SQL permet de définir différents types, dont voici seulement les plus courants
CHAR(n) : une chaîne de caractères de longueur n exactement.
VARCHAR(n) : une chaîne de caractères de longueur au maximum n.
TEXT : une chaîne de caractère de n'importe quelle longueur (acceptée par le système)

DATE : une date au format 'AAAA-MM-JJ'
TIME : une heure au format 'mm:mm:ss'
TIMESTAMP : un instant au format 'AAAA-MM-JJ mm:mm:ss'

INT : ou INTEGER, entier relatif sur 32 bits
DECIMAL(t,f) : décimal signé de t chiffres dont f après la virgule
REAL : (représentation approchée) flottant sur 32 bits

Remarque : les contraintes de domaine sont renseignées lors de la création d'une table

```
CREATE TABLE emprunt (code_barre CHAR(15),
                      isbn CHAR(14),
                      retour DATE);
```

b) contraintes d'unicité

Si une table possède une **clé primaire** (composée d'un ou plusieurs champs), alors chaque valeur de cette clé est nécessairement unique dans la table.
Le SGBD assure donc que toute insertion ou mise à jour des données ne pourra se faire qu'à condition de respecter cette contrainte d'unicité.

Remarque 1 : la contrainte de **clé primaire** peut être renseignée lors de la création de la table, ou par une requête spécifique

```
CREATE TABLE auteur (a_id INT PRIMARY KEY,
                     nom VARCHAR(90) NOT NULL,
                     prenom VARCHAR(90) NOT NULL);

ALTER TABLE emprunt ADD PRIMARY KEY(isbn);
```

Remarque 2 : l'attribut NOT NULL impose que le champ ne puisse pas rester « vide ».
Une valeur du domaine doit impérativement être attribuée à ce champ dans chaque ligne.

Remarque 3 : on peut ajouter une contrainte d'unicité pour un champ qui n'est pas une clé primaire

```
ALTER TABLE usager ADD UNIQUE( email);
```

c) contraintes de référence

Si une table RA possède une **clé étrangère** qui est la clé primaire d'une table RB, le SGBD peut assurer que toute valeur de RA se réfère bien à une ligne existant dans RB.
En particulier :

- toute insertion ou mise à jour d'une ligne de RA ne se fera qu'en cohérence avec les données contenues dans RB
- lors de la suppression d'une ligne de RB pour laquelle il existe une référence dans RA, le comportement du SGBD peut être programmé, avec notamment :
  - RESTRICT : (par défaut) rend impossible une suppression dans RB s'il existe encore une référence depuis RA.
  - CASCADE : supprime la ligne de RB ainsi que toutes les lignes de RA qui y font référence !!!
- ainsi, la base de donnée reste toujours intègre.

Remarque : les contraintes de clé étrangère peuvent être renseignées lors de la création de la table, ou par une requête spécifique (elles reçoivent alors un nom, souvent préfixé 'fk' = *foreign key*)

```
ALTER TABLE emprunt ADD CONSTRAINT fk_cb FOREIGN KEY (code_barre)
            REFERENCES usager(code_barre);

ALTER TABLE emprunt ADD CONSTRAINT fk_isbn FOREIGN KEY (isbn) REFERENCES livre(isbn);
```

d) contraintes utilisateur

Il est possible de déclarer des contraintes arbitraires sur les attributs d'une même ligne, avec CHECK.
Imaginons une table produit contenant un id (clé primaire) et un prix (décimal).
Pour éviter les erreurs de saisie, on peut ajouter une contrainte CHECK pour vérifier que le prix est positif.

```
CREATE TABLE produit (id : INT PRIMARY KEY,
                      prix : DECIMAL(10,2) NOT NULL,
                      CHECK (prix > 0));
```