Les contraintes sont un moyen permettant aux serveurs SQL d'assurer automatiquement l'intégrité d'une base de données. Les contraintes définissent des règles relatives aux valeurs autorisées dans les colonnes et elles constituent le mécanisme standard de mise en application de l'intégrité.

Il existe cinq catégories de contraintes.

1) NOT NULL indique que la colonne n'accepte pas les valeurs NULL.

NULL est une valeur spécifique aux bases de données qui représente une valeur inconnue, distincte d'un blanc ou de la valeur 0.

```
Create table client (
        nom             varchar(50)         NOT NULL,
        prenom          varchar(30)
)
```

2) Les contraintes CHECK assurent l'intégrité du domaine en limitant les valeurs placées dans une colonne.

Une contrainte CHECK indique une condition de recherche de type booléen (valeurs TRUE ou FALSE) appliquée à toutes les valeurs saisies dans la colonne ; toutes les valeurs ne prenant pas la valeur TRUE sont rejetées.

```
CREATE TABLE voiture (
        num                 int,
        marque              varchar(50),
        type                varchar(50),
        kilometrage         int CHECK (kilometrage BETWEEN 0 and 300000 )
)
```

3) Les contraintes UNIQUE garantissent le caractère unique des valeurs dans un ensemble de colonnes.

Dans une contrainte UNIQUE, deux lignes d'une table ne peuvent pas avoir la même valeur non NULL dans les colonnes. Les clés primaires mettent aussi en application l'unicité des valeurs, mais elles n'autorisent pas les valeurs NULL. Une contrainte UNIQUE est préférable à un index unique.

```
Create table client (
        num         int UNIQUE
        nom         varchar(50) ,
        prenom      varchar(30)
)
```

4) Les contraintes PRIMARY KEY identifient une colonne ou un ensemble de colonnes dont les valeurs identifient de manière unique chaque ligne d'une table.

Deux lignes d'une table ne peuvent pas avoir la même valeur de clé primaire. Dans une clé primaire, il est impossible d'entrer une valeur NULL dans une colonne. L'utilisation d'une petite colonne d'entiers comme clé primaire est recommandée. Chaque table devrait avoir une clé primaire.

Une table peut être constituée de plusieurs combinaisons de colonnes identifiant de manière unique les lignes d'une table ; chaque combinaison est une clé candidate. L'administrateur de la base de données sélectionne une des clés candidates pour en faire la clé primaire.

```
Create table client (
        num         int     PRIMARY KEY
        nom         varchar(50) ,
        prenom      varchar(30)
)
```

5) Les contraintes FOREIGN KEY identifient les relations entre les tables.

Une clé étrangère dans une table pointe sur une clé candidate dans une autre table. Vous ne pouvez pas insérer de ligne ayant une valeur de clé étrangère (à l'exception de NULL) s'il n'existe pas de clé candidate possédant cette valeur. Vous ne pouvez pas supprimer de ligne dans la table référencée s'il existe une valeur de clé étrangère référençant cette clé candidate. Dans l'exemple ci-dessous, la table **commande** crée une clé étrangère référençant la table **client** définie précédemment.

```
CREATE TABLE commande (
        num                 INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        datecommande        DATETIME,
        numclient           INT,
        FOREIGN KEY (numclient) REFERENCES client(num)
)
```

Les contraintes peuvent porter sur une colonne ou une table :

- Une contrainte de colonne est déclarée comme faisant partie de la définition de la colonne et ne s'applique qu'à celle-ci (c'est le cas des contraintes des exemples précédents).
- Une contrainte de table est déclarée indépendamment de la définition de la colonne et s'applique à plusieurs colonnes d'une table.

Les contraintes de table sont obligatoires lorsque plusieurs colonnes doivent être incluses dans une contrainte.

Si, par exemple, une table dispose de deux colonnes ou plus dans la clé primaire, vous devez utiliser une contrainte de table pour inclure les deux colonnes dans la clé primaire. Prenons l'exemple d'une table enregistrant les événements survenant sur une machine dans une usine. Supposons que des événements de plusieurs types puissent se produire au même moment, mais qu'aucun des deux événements survenant au même moment ne soit du même type. Pour reproduire ce cas de figure dans une table, vous pouvez inclure les colonnes **type** et **time** dans une clé primaire à deux colonnes.

```
CREATE TABLE factory_process (
        event_type          int,
        event_time          datetime,
        event_site          char(50),
        event_desc          char(1024),
        PRIMARY KEY (event_type, event_time)
)
```