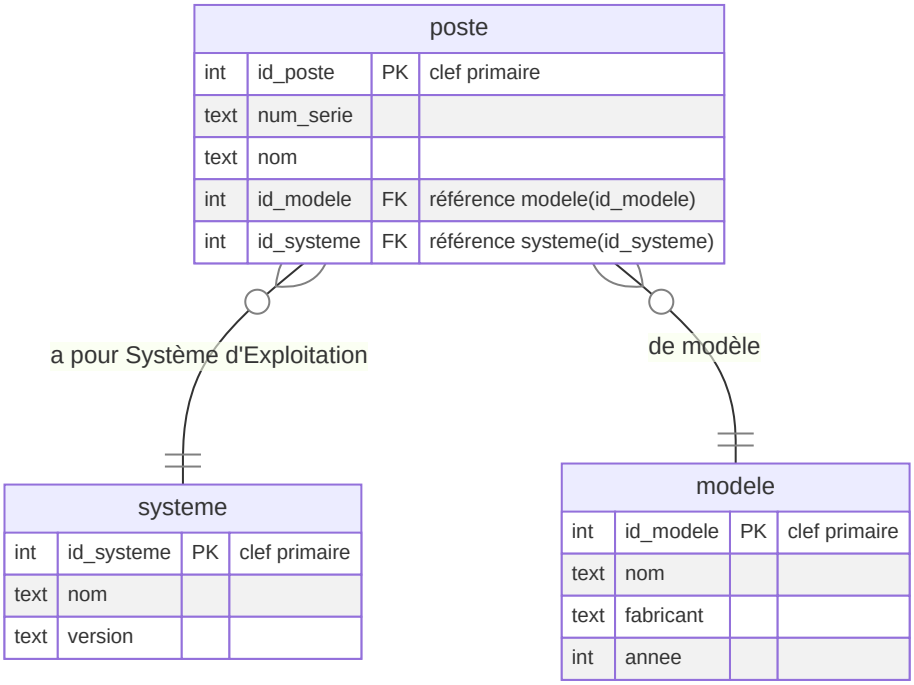
Méthode *Création de table et contraintes d'intégrité*

La clause SQL `CREATE` permet de créer les relations/tables d'une base de données en précisant pour chacune son **schéma relationnel** avec ses *contraintes d'intégrité* :

- *contrainte de domaine* associant à chaque attribut/colonne son domaine ;
- *contrainte de relation* avec déclaration d'une **clef primaire** pour chaque relation ;
- *contrainte d'intégrité référentielle* avec déclaration des éventuelles **clefs étrangères** avec les références aux clefs primaires d'autres relations qui leur sont liées.



Voici un exemple de requêtes permettant de créer le schéma relationnel représenté ci-dessus.

```
CREATE TABLE modele(
      id_modele INT PRIMARY KEY,
      nom TEXT,
      fabricant TEXT,
      annee INT);

CREATE TABLE systeme(
      id_systeme INT PRIMARY KEY,
      nom TEXT,
      version TEXT);

CREATE TABLE poste(
      id_poste INT PRIMARY KEY,
      num_serie TEXT UNIQUE NOT NULL,
      nom TEXT,
      id_modele INT,
      id_systeme INT,
      FOREIGN KEY (id_modele) REFERENCES modele(id_modele),
      FOREIGN KEY (id_systeme) REFERENCES systeme(id_systeme));
```

Méthode *Syntaxe générale d'une requête SQL*

```
SELECT attributs, fonctions d'agrégation (AS...)
FROM table (avec jointure éventuelle ... JOIN ... ON ... = ...)
WHERE conditions
ORDER BY attributs (... ASC ou DESC) ;
```

Méthode *Fonctions d'agrégation*

```
MIN(a) (minimum)  MAX(a) (maximum)  AVG(a) (moyenne)  SUM(a) (somme)

COUNT(a)   COUNT(DISTINCT a) (élimine les doublons) COUNT(*) (compte les
    lignes)
```

Méthode *Projection*

On peut récupérer des attributs et des fonctions d'agrégation :

```
SELECT nom, AVG(annee) - 2022, MAX(annee - 2022)

SELECT AVG(annee) - 2022 AS age_moyen (ceci est un renommage) ;

SELECT DISTINCT annee (pour enlever les doublons) ;
```

La projection se fait dans le `SELECT` *! Ne pas confondre avec la sélection qui se fait dans le* `WHERE`.

Méthode *Sélection*

```
SELECT ....
FROM ....
WHERE (annee >= 2018 AND annee <= 2021) OR (id_systeme = 4) ;
```

La condition peut être formée avec

```
AND   OR   NOT   <   <=   >   >=   =   <> (ou !=)
```

Méthode *Jointure*

```
SELECT ...
FROM T1 JOIN T2
    ON T1.a1 = T2.a2
```

Jointure sur deux tables

```
SELECT ...
FROM T1 JOIN T2 JOIN T3
    ON T1.a1 = T2.a2 AND T3.a3 = T2.b2
```

Jointure sur trois tables (il y a d'autres syntaxes possibles)

La jointure se fait sur des attributs qu'il est préférable de préfixer par le nom de la table. Sinon, il pourrait y avoir ambigüité (si des attributs de tables différentes portent le même nom).

Méthode *Classement*

```
ORDER BY anneee ou ORDER BY annee ASC (ordre croissant par défaut)

ORDER BY annee DESC (ordre décroissant)

ORDER BY annee DESC, nom ASC (on trie d'abord par année puis par nom)
```

Méthode *Requêtes imbriquées*

```
SELECT ...
FROM ...
WHERE a1 > (
           SELECT
                ...
          )
```

```
SELECT nom
FROM poste
WHERE annee = (
              SELECT MAX(annee)
              FROM eleves
              )
```

Ne pas oublier les parenthèses. L'indentation permet d'y voir clair.

Méthode *Insertion de valeurs*

Pour insérer une nouvelle ligne dans une table, on utilise la clause `INSERT` avec la syntaxe :

```
INSERT INTO table VALUES (val1, ..., valn) ;
```

Deux remarques importantes :

- *L'ordre des valeurs est celui des colonnes lors de la création de la table.*
- *L'insertion sera refusée par le SGBD si elle ne respecte pas les contraintes d'intégrité (relation, domaine, référentielle).*

Méthode *Suppression de lignes*

Pour supprimer un ensemble de lignes dans une table, on utilise la clause `DELETE` avec une condition de sélection pour spécifier la ou les lignes ciblée(s) :

```
DELETE FROM table WHERE condition ;
```

Méthode *Suppression de table*

Pour supprimer une table de la base, on utilise la clause `DROP TABLE`.

```
DROP TABLE table;
```

Toute suppression doit laisser la base dans un état qui respecte les contraintes d'intégrité référentielle.

Méthode *Mise à jour de table*

Pour mettre à jour un attribut d'une table avec une nouvelle valeur pour un ensemble de lignes, on utilise la clause `UPDATE` avec une condition de sélection sur la ou les lignes ciblées :

```
UPDATE table SET attribut = valeur WHERE condition ;
```

Toute mise à jour doit laisser la base dans un état qui respecte les contraintes d'intégrité référentielle : on ne peut pas mettre à jour une valeur référencée par une clef étrangère dans un autre table.