Gérer des tuples

Dans cet exemple on ajoute une nouvelle ligne (tuple) dans notre table, on précise d'abord la liste des colonnes auquel on va ajouter un résultat, puis on ajoute les valeurs correspondantes.

```
INSERT INTO Persons (PersonID, LastName, FirstName, Address, City) VALUES (1, 'Shumi', 'Roger', 'Rue du moulin, 13', 'Liège');
```

Les deux lignes décrites ci-dessus ont le même effet. Elles sélectionnent tous les tuples qui corespondent à la condition "PersonID=1" de la table "Persons" avec toutes les propriétés (c'est ce que veut aussi dire le `*`, c'est un "wildcard" qui inclut toutes les propriétés).

```
SELECT (PersonID, LastName, FirstName, Address, City) FROM Persons WHERE PersonID=1;
SELECT * FROM Persons WHERE PersonID=1;
```

Dans cet exemple on change les attributs Address et City dans tous les tuples qui remplissent la condition `PersonsID=1` dans la table "Persons". A noter que quand on a une chaine de caractères qui contient un `'` on peut quand même l'inclure en doublant le `'`.

```
UPDATE Persons SET Address='Rue de l''église', City='Verviers' WHERE PersonsID=1;
```

Mais on peut aussi modifier une table en se basant sur les valeurs de celle ci.

```
-- Ceci est un exemple dans une autre table où l'on modifie le montant en l'augmentant de 10%
UPDATE Bien SET montant = montant * 1.1 WHERE id_bien = 4;
```

On supprime Roger en supprimant tous les éléments de notre table "Persons" où `PersonsID=1`.

```
DELETE FROM Persons WHERE PersonsID=1;
```

Des SELECT plus avancés

Maintenant pour avoir un meilleur formatage, on peut utiliser des clauses `SELECT` plus évoluées.

Voici un exemple avec une date que l'on formatte en texte (sous le format tel que '12/12/2022') :

```
SELECT TO_CHAR(date_offre, 'dd/mm/yy')
FROM Offre
```

Renomer le nom d'une colonne avec `AS`

```
SELECT TO_CHAR(date_offre, 'dd/mm/yy') AS date
FROM Offre
```

Concaténer des colonnes et des chaines de caractères

```
SELECT 'La date est ' || date_offre || ' et le statut est ' || statut AS ma_super_colonne_inutile
FROM Offre
```

Enfin si on veut éviter que des lignes apparaissent plusieurs fois on peut utiliser le mot-clé `DISTINCT`

```
-- On prends les localités de tous les candidats mais on supprime toutes les lignes en doublon
SELECT DISTINCT localite
FROM Candidat
```

Des WHERE plus avancés

On peut faire différentes opération de base dans les clauses `WHERE` tel que >, <, <=, >=, =

```
SELECT * FROM Offre WHERE id_offre = 4
```

On peut ensuite lier plusieurs comparaisons avec des opérateurs booléens (AND, OR, NOT)

```
SELECT * FROM Offre WHERE NOT id_offre = 4 AND montant > 1000;
```

Pour vérifier si il n'y a aucune valeur, on peut utiliser `IS NULL`

```
SELECT * FROM Offre WHERE id_candidat IS NULL;
```

Enfin pour tester des patterns de chaines de caractères on peut utiliser le mot clé `LIKE`

```
-- Ne retourne que les tuples de la table bien qui ont 'calme' en lowercase dans leur description
-- % signifie "n'importe quel chaine de caractère"
-- _ signifie "n'importe quel caractère"
SELECT * FROM Bien WHERE description LIKE '%calme%';
```

```
-- Ici on veut tous les tuples de Bien qui ont le symbole % dans leur description,
-- mais % est un caractère de syntaxe de LIKE donc on doit l'escape ici on choisis \
-- Si on veut après escape '\' alors on peut simplement le mettre deux fois '\\'
SELECT * FROM Bien WHERE description LIKE '%\%%' ESCAPE '\';
```

Pour vérifier si une valeur est présente dans une certaine liste de valeur il est préférable d'utiliser `IN`

```
-- Ce code va garder tous les tuples dont la localité est soit Liège, soit Neupré
SELECT id_bien, localite
FROM Bien
WHERE localite IN ('Liège', 'Neupré');
```

Maintenant si on veut facilement savoir si une valeur est entre 2 valeurs on peut utiliser le mot-clé `BETWEEN`

```
-- Le mot-clé BETWEEN fonctionne aussi avec des dates par exemple
SELECT id_offre, date_offre, montant
FROM Offre
WHERE montant BETWEEN 200000 AND 300000
```

Utiliser ORDER BY pour changer l'ordre des tuples

On peut choisir comment on veut organiser nos tuples (ordre ascendant, descendant, alphabétique, numérique, etc)

```
-- Ceci va faire apparaitre toutes les Offres ordonné par le montant dans un ordre descendant (du plus cher au moins cher)
-- Si on remplace DESC par ASC, on aura l'inverse (du moins cher au plus cher)
SELECT * FROM Offre
ORDER BY montant DESC;
-- OUTPUT:   ID_OFFRE    MONTANT DATE_OFF S ID_CANDIDAT    ID_BIEN
--         ---------- ---------- -------- - ----------- ----------
--               19     262000 13/10/12 E         19        19
--               11     260000 02/10/12 I         11        19
--                7     257500 17/09/12 R         7         19
```

```
-- On peut aussi mettre plusieurs ORDER BY à la suite comme 'ORDER BY montant DESC, id_offre ASC'
-- En faisant ça tous les tuples qui ont le même montant seront ordonnés par l'id_offre
```

Revision #1
Created 26 April 2023 19:01:30 by SnowCode
Updated 26 April 2023 19:01:31 by SnowCode