

Bases de données

Cours 5 – Langage SQL – Langage de Manipulation des Données

Adrien Ugon

ESIEE-Paris

Lundi 18 Février 2018

- 1 Requête élémentaire
- 2 Jointures
- 3 Expressions et Fonctions
- 4 Agrégation et statistiques
- 5 Vues
- 6 Ajout, Modification et suppression de valeurs

Plan

- 1 Requête élémentaire
- 2 Jointures
- 3 Expressions et Fonctions
- 4 Agrégation et statistiques
- 5 Vues
- 6 Ajout, Modification et suppression de valeurs

Requêtes élémentaires

Les requêtes élémentaires permettent de faire des extractions simples à partir d'une table de la base de données.

Le format des requêtes simples est le suivant :

```
SELECT attribut_1,attribut_2,...,attribut_n  
FROM nom_table;
```

attribut_1,attribut_2,...,attribut_n sont des attributs de la table *nom_table*.

Il s'agit d'une projection de la table *nom_table* sur les attributs *attribut_1,attribut_2,...,attribut_n*.

Requêtes élémentaires

Exemple

```
SELECT IPP, nom, prenom, adresse, code_postal  
FROM Patient;
```

Le symbole * dans les requêtes

L'utilisation du symbole * permet de spécifier que l'on cherche l'ensemble des attributs.

Le format des requêtes est alors le suivant :

```
SELECT *  
FROM nom_table;
```

Exemple

Exemple

```
SELECT *  
FROM Patient;
```

Mot-clé DISTINCT

Parfois, on ne souhaite avoir que les valeurs distinctes.

On utilise alors le mot-clé DISTINCT :

```
SELECT DISTINCT attribut_1,attribut_2,...,attribut_n  
FROM nom_table ;
```


Mot-clé DISTINCT

Exemple

Exemple

```
SELECT DISTINCT prenom, code_postal  
FROM Patient;
```

Requête avec critère de sélection

Il est possible d'ajouter des conditions pour filtrer les enregistrements à afficher.

Le mot-clé est WHERE :

```
SELECT attribut_1,attribut_2,...,attribut_n  
FROM nom_table  
WHERE expression logique ;
```

Requêtes avec critère de sélection

Exemple 1

Exemple

```
SELECT IPP, nom, prenom, adresse  
FROM Patient  
WHERE code_postal=93160 ;
```

Requêtes avec critère de sélection

Conjonction et disjonction

Conjonction de conditions

```
SELECT IPP, nom, prenom, adresse  
FROM Patient  
WHERE code_postal=93160  
AND sexe='F' ;
```

Disjonction de conditions

```
SELECT IPP, nom, prenom, adresse  
FROM Patient  
WHERE code_postal=93160  
OR sexe='F' ;
```

Requêtes avec critère de sélection

Négation

Exemple

```
SELECT IPP, nom, prenom, adresse  
FROM Patient  
WHERE NOT code_postal=93160 ;
```

Exemple

```
SELECT IPP, nom, prenom, adresse  
FROM Patient  
WHERE code_postal<>93160 ;
```

Requête avec critère de sélection

Le prédicat IN

Le prédicat IN permet de spécifier une **liste** de valeurs à filtrer pour un attribut

La syntaxe est la suivante :

[NOT] IN (*valeur_1*, *valeur_2*, ..., *valeur_n*)

Requêtes avec critère de sélection

Exemple avec le prédicat IN

Exemple

```
SELECT IPP, nom, prenom, adresse  
FROM Patient  
WHERE code_postal IN (93160, 75001, 78145);
```

Requête avec critère de sélection

Le prédicat BETWEEN

Le prédicat BETWEEN permet de spécifier une **plage** de valeurs à filtrer pour un attribut

La syntaxe est la suivante :

```
[NOT] BETWEEN valeur_min AND valeur_max
```


Requêtes avec critère de sélection

Exemple avec le prédicat BETWEEN

Exemple

```
SELECT IPP, nom, prenom, adresse  
FROM Patient  
WHERE taille BETWEEN 1.65 AND 1.85;
```

Requête avec critère de sélection

Le prédicat LIKE

Le prédicat LIKE permet de spécifier un **modèle** pour les valeurs cherchées sur un attribut

La syntaxe est la suivante :

[NOT] LIKE 'modèle'

On utilise :

- le symbole souligné (underscore) ' ' pour indiquer une lettre quelconque.
- le symbole pourcentage '%' pour indiquer n'importe quelle suite de lettres quelconques (y compris 0 caractère).

Requêtes avec critère de sélection

Exemples avec le prédicat BETWEEN

Exemple 1

```
SELECT IPP, nom, adresse, code_postal  
FROM Patient  
WHERE prenom LIKE 'T _ T _';
```

Cela filtre les valeurs suivantes :

- TOTO;
- TATA;
- TITI;
- TATO;
- ...

Requêtes avec critère de sélection

Exemples avec le prédicat BETWEEN

Exemple 1

```
SELECT IPP, nom, adresse, code_postal  
FROM Patient  
WHERE prenom LIKE 'T % T %';
```

Cela filtre les valeurs suivantes :

- TINTIN;
- TABATA;
- TESSADIT;
- THEOTINE;
- TRISTANE;
- ...

Requête avec critère de sélection

Le prédicat IS [NOT] NULL

Le prédicat IS NULL permet de chercher les tuples où un attribut a la valeur NULL.

La syntaxe est la suivante :

IS [NOT] NULL

Requêtes avec critère de sélection

Exemples avec le prédicat IS NULL

Exemple

```
SELECT IPP, nom, adresse, code__postal  
FROM Patient  
WHERE date__mariage IS NULL;
```

Évaluation des expressions logiques avec un membre NULL

la valeur UNKNOWN

Considérons une expression logique de la forme :

$$E1 \text{ comp } E2$$

où :

- E1 et E2 sont des expressions de type prédéfini
- comp est un des opérateurs suivants : =, <>, <, <=, >, >=

On suppose que E1 ou E2 a la valeur NULL.

L'expression est alors évaluée à la valeur UNKNOWN.

Requête avec critère de sélection

Les prédicats IS [NOT] TRUE, IS [NOT] FALSE, IS [NOT] UNKNOWN

Les prédicats IS [NOT] TRUE, IS [NOT] FALSE, IS [NOT] UNKNOWN permettent de chercher les tuples où une expression logique est évaluée à la valeur TRUE, FALSE ou UNKNOWN.

expression logique	TRUE	FALSE	UNKNOWN
IS TRUE	TRUE	FALSE	FALSE
IS FALSE	FALSE	TRUE	FALSE
IS UNKNOWN	FALSE	FALSE	TRUE
IS NOT TRUE	FALSE	TRUE	TRUE
IS NOT FALSE	TRUE	FALSE	TRUE
IS NOT UNKNOWN	TRUE	TRUE	FALSE

Trier les résultats

Il est possible de trier le résultat d'une requête en spécifiant les attributs à utiliser pour ordonner les valeurs.

Le tri utilise le prédicat ORDER BY.

Sa syntaxe est la suivante :

```
ORDER BY attribut_1, attribut_2, ..., attribut_n [ASC|DESC]
```

ASC Les données sont ordonnées dans l'ordre *croissant*.

DESC Les données sont ordonnées dans l'ordre *décroissant*.

Tri des résultats

Exemple

Exemple

```
SELECT IPP, nom, adresse, date_naissance, code_postal  
FROM Patient  
ORDER BY date_naissance DESC;
```

Plan

1 Requête élémentaire

2 Jointures

- Jointure simple
- θ -jointure
- Jointure naturelle
- Jointure externe
- Union, intersection et différence de requêtes

3 Expressions et Fonctions

4 Agrégation et statistiques

5 Vues

Plan

1 Requête élémentaire

2 Jointures

- Jointure simple
- θ -jointure
- Jointure naturelle
- Jointure externe
- Union, intersection et différence de requêtes

3 Expressions et Fonctions

4 Agrégation et statistiques

5 Vues

Jointure simple

La jointure des tables T_1, T_2, \dots, T_n résulte en leur produit cartésien.

La requête est la suivante :

```
SELECT [DISTINCT] attribut_1, attribut_2, ..., attribut_m  
FROM T_1, T_2, ..., T_n;
```

Jointure simple

Exemple

Soient les deux tables suivantes :

Patient

id	nom	prenom	date_naissance
10	DUPONT	Jean	1984-03-12
28	BAUDOUIN	Anne	2003-06-29
134	ZIGUE	Luc	1943-08-24
179	JOUE	Odile	1963-01-19

Consultation

id_patient	heure_rdv	examen
10	2018-02-18 14 :30 :00	scanner
28	2018-03-28 15 :45 :00	angiographie
134	2018-03-01 09 :15 :00	ECG
134	2018-03-01 11 :30 :00	echocardiographie
134	2018-03-01 14 :00 :00	ECG effort
179	2018-02-18 08 :30 :00	EMG

Exemple de requête

```
SELECT *
FROM Patient, Consultation ;
```

Quel est le résultat de cette requête ?

Jointure simple

Exemple — Résultat de la requête

id	nom	prenom	date_naissance	id_patient	heure_rdv	examen
10	DUPONT	Jean	1984-03-12	10	2018-02-18 14 :30 :00	scanner
28	BAUDOUIN	Anne	2003-06-29	10	2018-02-18 14 :30 :00	scanner
134	ZIGUE	Luc	1943-08-24	10	2018-02-18 14 :30 :00	scanner
179	JOUE	Odile	1963-01-19	10	2018-02-18 14 :30 :00	scanner
10	DUPONT	Jean	1984-03-12	28	2018-03-28 15 :45 :00	angiographie
28	BAUDOUIN	Anne	2003-06-29	28	2018-03-28 15 :45 :00	angiographie
134	ZIGUE	Luc	1943-08-24	28	2018-03-28 15 :45 :00	angiographie
179	JOUE	Odile	1963-01-19	28	2018-03-28 15 :45 :00	angiographie
10	DUPONT	Jean	1984-03-12	134	2018-03-01 09 :15 :00	ECG
28	BAUDOUIN	Anne	2003-06-29	134	2018-03-01 09 :15 :00	ECG
134	ZIGUE	Luc	1943-08-24	134	2018-03-01 09 :15 :00	ECG
179	JOUE	Odile	1963-01-19	134	2018-03-01 09 :15 :00	ECG
10	DUPONT	Jean	1984-03-12	134	2018-03-01 11 :30 :00	echocardiographie
28	BAUDOUIN	Anne	2003-06-29	134	2018-03-01 11 :30 :00	echocardiographie
134	ZIGUE	Luc	1943-08-24	134	2018-03-01 11 :30 :00	echocardiographie
179	JOUE	Odile	1963-01-19	134	2018-03-01 11 :30 :00	echocardiographie
10	DUPONT	Jean	1984-03-12	134	2018-03-01 14 :00 :00	ECG effort
28	BAUDOUIN	Anne	2003-06-29	134	2018-03-01 14 :00 :00	ECG effort
134	ZIGUE	Luc	1943-08-24	134	2018-03-01 14 :00 :00	ECG effort
179	JOUE	Odile	1963-01-19	134	2018-03-01 14 :00 :00	ECG effort
10	DUPONT	Jean	1984-03-12	179	2018-02-18 08 :30 :00	EMG
28	BAUDOUIN	Anne	2003-06-29	179	2018-02-18 08 :30 :00	EMG
134	ZIGUE	Luc	1943-08-24	179	2018-02-18 08 :30 :00	EMG
179	JOUE	Odile	1963-01-19	179	2018-02-18 08 :30 :00	EMG

Plan

1 Requête élémentaire

2 Jointures

- Jointure simple
- θ -jointure
- Jointure naturelle
- Jointure externe
- Union, intersection et différence de requêtes

3 Expressions et Fonctions

4 Agrégation et statistiques

5 Vues

θ -jointure

La θ -jointure entre deux tables utilise un attribut commun. Cet attribut est utilisé pour faire la jointure et ses valeurs sont utilisées pour générer les tuples.

La requête est la suivante :

```
SELECT [DISTINCT] attribut_1, attribut_2, ..., attribut_m
FROM Table_1 [INNER] JOIN Table_2 ON expression_logique
[INNER] JOIN Table_3 ON expression_logique
... ;
```

θ -jointure

Exemple

Soient les deux tables suivantes :

Patient			
id	nom	prenom	date_naissance
10	DUPONT	Jean	1984-03-12
28	BAUDOUIN	Anne	2003-06-29
134	ZIGUE	Luc	1943-08-24
179	JOUE	Odile	1963-01-19

Consultation		
id_patient	heure_rdv	examen
10	2018-02-18 14 :30 :00	scanner
28	2018-03-28 15 :45 :00	angiographie
134	2018-03-01 09 :15 :00	ECG
134	2018-03-01 11 :30 :00	echocardiographie
134	2018-03-01 14 :00 :00	ECG effort
179	2018-02-18 08 :30 :00	EMG

Exemple de requête

```

SELECT *
FROM Patient INNER JOIN Consultation ON
Patient.id=Consultation.id_patient;

```

Quel est le résultat de cette requête ?

θ -jointure

Exemple — Résultat de la requête

id	nom	prenom	date_naissance	id_patient	heure_rdv	examen
10	DUPONT	Jean	1984-03-12	10	2018-02-18 14 :30 :00	scanner
28	BAUDOUIN	Anne	2003-06-29	28	2018-03-28 15 :45 :00	angiographie
134	ZIGUE	Luc	1943-08-24	134	2018-03-01 09 :15 :00	ECG
134	ZIGUE	Luc	1943-08-24	134	2018-03-01 11 :30 :00	echocardiographie
134	ZIGUE	Luc	1943-08-24	134	2018-03-01 14 :00 :00	ECG effort
179	JOUBE	Odile	1963-01-19	179	2018-02-18 08 :30 :00	EMG

Plan

1 Requête élémentaire

2 Jointures

- Jointure simple
- θ -jointure
- **Jointure naturelle**
- Jointure externe
- Union, intersection et différence de requêtes

3 Expressions et Fonctions

4 Agrégation et statistiques

5 Vues

Jointure naturelle

La jointure naturelle est une θ -jointure qui se fait automatiquement sur les colonnes de même nom.

La requête est la suivante :

```
SELECT [DISTINCT] attribut_1, attribut_2, ..., attribut_n  
FROM Table_1  
NATURAL JOIN Table_2  
NATURAL JOIN Table_3  
... ;
```

θ -jointure

Exemple

Soient les deux tables suivantes :

Patient

id	nom	prenom	date_naissance
10	DUPONT	Jean	1984-03-12
28	BAUDOUIN	Anne	2003-06-29
134	ZIGUE	Luc	1943-08-24
179	JOUE	Odile	1963-01-19

Consultation

id	heure_rdv	examen
10	2018-02-18 14 :30 :00	scanner
28	2018-03-28 15 :45 :00	angiographie
134	2018-03-01 09 :15 :00	ECG
134	2018-03-01 11 :30 :00	echocardiographie
134	2018-03-01 14 :00 :00	ECG effort
179	2018-02-18 08 :30 :00	EMG

Exemple de requête

```
SELECT *
FROM Patient NATURAL JOIN Consultation ;
```

Quel est le résultat de cette requête ?

Jointure naturelle

Exemple — Résultat de la requête

id	nom	prenom	date_naissance	heure_rdv	examen
10	DUPONT	Jean	1984-03-12	2018-02-18 14 :30 :00	scanner
28	BAUDOUIN	Anne	2003-06-29	2018-03-28 15 :45 :00	angiographie
134	ZIGUE	Luc	1943-08-24	2018-03-01 09 :15 :00	ECG
134	ZIGUE	Luc	1943-08-24	2018-03-01 11 :30 :00	echocardiographie
134	ZIGUE	Luc	1943-08-24	2018-03-01 14 :00 :00	ECG effort
179	JOUE	Odile	1963-01-19	2018-02-18 08 :30 :00	EMG

Plan

1 Requête élémentaire

2 Jointures

- Jointure simple
- θ -jointure
- Jointure naturelle
- **Jointure externe**
- Union, intersection et différence de requêtes

3 Expressions et Fonctions

4 Agrégation et statistiques

5 Vues

Jointure externe

La jointure externe permet de compléter les enregistrements d'une première table avec ceux d'une deuxième table.

Il existe :

- la jointure externe gauche ;
- la jointure externe droite ;
- la jointure externe complète.

Jointure externe gauche

La requête de jointure externe gauche est la suivante :

```
SELECT [DISTINCT] attribut_1, attribut_2, ..., attribut_n  
FROM Table_1  
LEFT [OUTER] JOIN Table_2 ON expression_logique_1  
LEFT [OUTER] JOIN Table_3 ON expression_logique_2  
... ;
```

Jointure externe gauche

Exemple

Soient les deux tables suivantes :

Consultation

id_patient	heure_rdv	examen
10	2018-02-18 14 :30 :00	scanner
28	2018-03-28 15 :45 :00	angiographie
134	2018-03-01 09 :15 :00	ECG
134	2018-03-01 11 :30 :00	echocardiographie
134	2018-03-01 14 :00 :00	ECG effort
179	2018-02-18 08 :30 :00	EMG

Examen

nom	batiment
echocardiographie	Lavoisier
radiologie	Proust
scanner	babinsky

Exemple de requête

```
SELECT *
FROM Consultation
LEFT JOIN Examens ON Consultation.examen=Examen.nom ;
```

Quel est le résultat de cette requête ?

Jointure externe gauche

Exemple — Résultat de la requête

id_patient	heure_rdv	examen	nom	batiment
10	2018-02-18 14 :30 :00	scanner	scanner	babinsky
28	2018-03-28 15 :45 :00	angiographie	NULL	NULL
134	2018-03-01 09 :15 :00	ECG	NULL	NULL
134	2018-03-01 11 :30 :00	echocardiographie	echocardiographie	Lavoisier
134	2018-03-01 14 :00 :00	ECG effort	NULL	NULL
179	2018-02-18 08 :30 :00	EMG	NULL	NULL

Jointure externe droite

Exemple

Soient les deux tables suivantes :

Consultation

id_patient	heure_rdv	examen
10	2018-02-18 14 :30 :00	scanner
28	2018-03-28 15 :45 :00	angiographie
134	2018-03-01 09 :15 :00	ECG
134	2018-03-01 11 :30 :00	echocardiographie
134	2018-03-01 14 :00 :00	ECG effort
179	2018-02-18 08 :30 :00	EMG

Examen

nom	batiment
echocardiographie	Lavoisier
radiologie	Proust
scanner	babinsky

Exemple de requête

```
SELECT *
FROM Consultation
RIGHT JOIN Examens ON Consultation.examen=Examen.nom ;
```

Quel est le résultat de cette requête ?

Jointure externe droite

Exemple — Résultat de la requête

id_patient	heure_rdv	examen	nom	batiment
10	2018-02-18 14 :30 :00	scanner	scanner	babinsky
134	2018-03-01 11 :30 :00	echocardiographie	echocardiographie	Lavoisier
NULL	NULL	NULL	radiologie	Proust

Jointure externe totale

Exemple

Soient les deux tables suivantes :

Consultation

id_patient	heure_rdv	examen
10	2018-02-18 14 :30 :00	scanner
28	2018-03-28 15 :45 :00	angiographie
134	2018-03-01 09 :15 :00	ECG
134	2018-03-01 11 :30 :00	echocardiographie
134	2018-03-01 14 :00 :00	ECG effort
179	2018-02-18 08 :30 :00	EMG

Examen

nom	batiment
echocardiographie	Lavoisier
radiologie	Proust
scanner	babinsky

Exemple de requête

```
SELECT *
FROM Consultation
FULL JOIN Examens ON Consultation.examen=Examen.nom ;
```

Quel est le résultat de cette requête ?

Jointure externe totale

Exemple — Résultat de la requête

id_patient	heure_rdv	examen	nom	batiment
10	2018-02-18 14 :30 :00	scanner	scanner	babinsky
28	2018-03-28 15 :45 :00	angiographie	NULL	NULL
134	2018-03-01 09 :15 :00	ECG	NULL	NULL
134	2018-03-01 11 :30 :00	echocardiographie	echocardiographie	Lavoisier
134	2018-03-01 14 :00 :00	ECG effort	NULL	NULL
179	2018-02-18 08 :30 :00	EMG	NULL	NULL
NULL	NULL	NULL	radiologie	Proust

Plan

1 Requête élémentaire

2 Jointures

- Jointure simple
- θ -jointure
- Jointure naturelle
- Jointure externe
- Union, intersection et différence de requêtes

3 Expressions et Fonctions

4 Agrégation et statistiques

5 Vues

L'opérateur UNION

L'opérateur UNION permet de faire l'union de 2 requêtes.
Sa syntaxe est la suivante :

```
requete_1  
UNION  
requete_2
```

Le résultat comporte alors les résultats de la requête 1 et de la requête 2.

L'opérateur UNION

Exemple

Exemple

```
SELECT RPPS, nom, prenom, date_naissance  
FROM Medecin  
UNION  
SELECT ADELI, nom, prenom, date_naissance  
FROM Infirmier ;
```

L'opérateur INTERSECTION

L'opérateur INTERSECTION permet de faire l'intersection de 2 requêtes.
Sa syntaxe est la suivante :

```
requete_1  
INTERSECTION  
requete_2
```

Le résultat comporte alors les résultats qui sont présents à la fois dans les résultats de la requête 1 et dans ceux de la requête 2.

L'opérateur INTERSECTION

Exemple

Exemple

```
SELECT IPP, nom, prenom, date_naissance  
FROM Patient LEFT JOIN Consultation ON  
Patient.id=Consultation.id_patient  
WHERE service="cardiologie" INTERSECTION  
SELECT IPP, nom, prenom, date_naissance  
FROM Patient LEFT JOIN Consultation ON  
Patient.id=Consultation.id_patient  
WHERE service="pneumologie";
```

L'opérateur EXCEPT

L'opérateur EXCEPT permet de faire la différence entre de 2 requêtes. Sa syntaxe est la suivante :

```
requete_1  
EXCEPT  
requete_2
```

Le résultat comporte alors les résultats qui sont présents dans les résultats de la requête 2 mais pas dans ceux de la requête 1.

L'opérateur EXCEPT

Exemple

Exemple

```
SELECT id, nom, prenom, date_naissance  
FROM Personnel  
EXCEPT  
SELECT RPPS, nom, prenom, date_naissance  
FROM Medecin ;
```

Plan

- 1 Requête élémentaire
- 2 Jointures
- 3 Expressions et Fonctions**
 - Expressions
 - Fonctions
 - Fonctions pour les valeurs numériques
 - Opérateur CASE
- 4 Agrégation et statistiques
- 5 Vues

Plan

- 1 Requête élémentaire
- 2 Jointures
- 3 Expressions et Fonctions
 - Expressions
 - Fonctions
 - Fonctions pour les valeurs numériques
 - Opérateur CASE
- 4 Agrégation et statistiques
- 5 Vues

Opération sur les attributs

Il est possible de faire des opérations sur les attributs.

- faire des calculs ou des comparaisons sur les attributs numériques;
- faire des comparaisons de dates.

Le format des requêtes simples est le suivant :

```
SELECT expression_1 AS N_1, expression_2 AS  
N_2, ..., expression_n AS N_n  
FROM nom_table;
```

Opération sur les attributs

Exemple

Exemple

```
SELECT DCI, dosage, quantite, dosage*quantite AS dose_totale  
FROM Medicament ;
```

Traitement de la valeur NULL

Si un des attributs utilisé dans une expression a une valeur NULL, alors l'expression aura aussi pour valeur :

- NULL pour les expressions numériques ;
- UNKNOWN pour les expressions booléennes.

Traitement de la valeur NULL

Exemple

Exemple

```
SELECT DCI, dosage, quantite, dosage*quantite AS dose_totale  
FROM Medicament ;
```

si dosage vaut NULL, alors dose_totale vaut NULL.

Plan

- 1 Requête élémentaire
- 2 Jointures
- 3 Expressions et Fonctions**
 - Expressions
 - **Fonctions**
 - Fonctions pour les valeurs numériques
 - Opérateur CASE
- 4 Agrégation et statistiques
- 5 Vues

Plan

1 Requête élémentaire

2 Jointures

3 Expressions et Fonctions

- Expressions
- Fonctions
- **Fonctions pour les valeurs numériques**
 - Fonctions pour les chaînes de caractères
 - Fonctions pour les données temporelles
 - Fonctions pour la gestion de la valeur NULL
 - Fonctions de conversion
- Opérateur CASE

4 Agrégation et statistiques

Fonctions d'arrondi

SQL définit 3 fonctions permettant d'arrondir les flottants à des entiers :

FLOOR arrondit à l'entier inférieur le plus proche.

CEIL arrondit à l'entier supérieur le plus proche.

ROUND arrondit à l'entier le plus proche.

Leur syntaxe est la suivante :

FLOOR(*attribut*)
CEIL(*attribut*)
ROUND(*attribut*)

Concaténation

L'opérateur de concaténation s'écrit `||`.

Il s'utilise de la façon suivante :

```
chaîne_1 || chaîne_2
```

Exemple de concaténation

```
SELECT prenom || ' ' || nom AS nom_complet, adresse  
FROM Patient;
```

Les fonctions UPPER et LOWER

Changement de casse de caractères

Les fonctions UPPER et LOWER permettent de changer la casse de caractères de l'argument.

UPPER L'argument est retourné en **majuscules**.

LOWER L'argument est retourné en **minuscules**.

Exemple de fonction UPPER et LOWER

```
SELECT UPPER(nom) AS nom_majuscule,  
       LOWER(prenom) AS prenom_minuscule  
FROM Patient;
```

La fonction POSITION

La fonction POSITION retourne la position de la première occurrence d'une sous-chaîne de caractères dans une autre chaîne de caractères. Si la sous-chaîne n'est pas présente, on retourne 0.

Sa syntaxe est la suivante :

```
POSITION(sous-chaîne IN chaîne)
```

Exemple de fonction POSITION

```
SELECT POSITION("lut" IN "turlututu") AS position  
FROM Patient;
```

La fonction SUBSTRING

La fonction SUBSTRING retourne la sous-chaîne d'une chaîne de caractères passée en argument commençant à la position *d* et de longueur *l*.

Sa syntaxe est la suivante :

```
SUBSTRING(chaîne FROM d FOR l)
```

Exemple de fonction SUBSTRING

```
SELECT SUBSTRING(code_postal FROM 1 FOR 2) AS departement  
FROM Patient;
```

La fonction TRIM

La fonction TRIM supprime toutes les occurrences d'un caractère en début et/ou en fin d'une chaîne de caractères passée en argument.

Sa syntaxe est la suivante :

`TRIM(position caractere FROM chaîne)`

Par défaut, le caractère supprimé est l'espace.

Le paramètre *position* peut prendre les valeurs suivantes :

LEADING suppression en début de chaîne.

TRAILING suppression en fin de chaîne.

BOTH suppression en début et en fin de chaîne (valeur par défaut).

Exemple de fonction TRIM

```
SELECT TRIM(" SQL Tutorial ") AS TrimmedString;
```

La fonction CHARACTER_LENGTH

La fonction CHARACTER_LENGTH retourne le nombre de caractères d'une chaîne de caractères passée en argument.

Sa syntaxe est la suivante :

```
CHARACTER_LENGTH(chaîne)
```

Exemple de fonction CHARACTER_LENGTH

```
SELECT CHARACTER_LENGTH(adresse) AS longueur;
```

Les fonctions OCTET_LENGTH et BIT_LENGTH

La fonction OCTET_LENGTH et BIT_LENGTH retournent la taille d'une valeur passée en argument.

OCTET_LENGTH retourne la taille en **octets**.

BIT_LENGTH retourne la taille en **bits**.

La syntaxe est la suivante :

`OCTET_LENGTH(valeur)BIT_LENGTH(valeur)`

Exemple de fonction CHARACTER_LENGTH

```
SELECT OCTET_LENGTH(adresse) AS taille ;
```

Les fonctions CURRENT_DATE, CURRENT_TIME et CURRENT_TIMESTAMP

Les fonctions CURRENT_DATE, CURRENT_TIME et CURRENT_TIMESTAMP retournent la date et l'heure courante (de la machine).

CURRENT_DATE retourne la date courante;

CURRENT_TIME retourne l'heure courante;

CURRENT_TIMESTAMP retourne la date et l'heure courantes.

Exemple

```
SELECT CURRENT_DATE(),  
CURRENT_TIME(),  
CURRENT_TIMESTAMP();
```


La fonction EXTRACT

La fonction EXTRACT permet d'extraire une partie de la valeur d'une expression temporelle.

Sa syntaxe est la suivante :

```
EXTRACT(partie FROM expression)
```

Le paramètre *partie* prend une des valeurs suivantes : YEAR, MONTH, DAY, HOUR, MINUTE, SECOND.

Exemple de la fonction EXTRACT

```
SELECT EXTRACT(YEAR FROM date_naissance);
```

La fonction DATEDIFF

La fonction DATEDIFF donne le nombre de jours entre deux dates.

Sa syntaxe est la suivante :

`DATEDIFF(date_1, date_2)`

prérequis : *date_1* est antérieure à *date_2*.

Exemple de la fonction DATEDIFF

```
SELECT DATEDIFF(CURRENT_DATE(), date_naissance) AS age  
FROM Patient;
```

La fonction OVERLAPS

La fonction OVERLAPS teste le recouvrement de deux périodes temporelles.

Sa syntaxe est la suivante :

$(D1, F1) \text{ OVERLAPS } (D2, F2)$

- $D1, F1, D2, F2$ sont de même type (DATE, TIME ou TIMESTAMP)
- $D1 < F1, D2 < F2$

Exemple de la fonction OVERLAPS

```
SELECT (DATE '2018-01-01', DATE '2018-12-31') OVERLAPS (DATE  
'2018-09-01', DATE '2019-06-30');
```

La fonction COALESCE

La fonction COALESCE prend deux arguments : *argument_1* et *argument_2*.

Si *argument_1* vaut NULL ALORS la fonction retourne *argument_2*
SINON la fonction retourne NULL.

Sa syntaxe est la suivante :

COALESCE(*argument_1*, *argument_2*)

Exemple de la fonction COALESCE

```
SELECT COALESCE(SaO2, SpO2)  
FROM Resultat;
```

La fonction NULLIF

La fonction NULLIF prend deux arguments : *argument_1* et *argument_2*.

Si *argument_1* est égal à *argument_2* ALORS la fonction retourne NULL
SINON elle retourne *argument_1*.

Sa syntaxe est la suivante :

NULLIF(*argument_1*, *argument_2*)

Exemple de la fonction NULLIF

```
SELECT NULLIF(pouls_1, pouls_2)
FROM Resultat;
```

La fonction CONVERT

La fonction CONVERT permet de convertir une valeur dans un autre type de données.

Sa syntaxe est la suivante :

CONVERT(*type_final*, *valeur*, *format*)

La valeur de *format* permet de spécifier un format de retour (par exemple pour une date). Plus d'informations à l'adresse

https://www.w3schools.com/SQL/func_sqlserver_convert.asp

Exemple de la fonction CONVERT

```
SELECT CONVERT(SMALLINT, code_postal)
FROM Patient;
```

La fonction CAST

La fonction CAST permet de convertir une valeur dans un autre type de données.

Sa syntaxe est la suivante :

CAST(*expression* AS *type*)

Exemple de la fonction CAST

```
SELECT CAST(code_postal AS SMALLINT)  
FROM Patient;
```

Plan

- 1 Requête élémentaire
- 2 Jointures
- 3 Expressions et Fonctions**
 - Expressions
 - Fonctions
 - Fonctions pour les valeurs numériques
 - **Opérateur CASE**
- 4 Agrégation et statistiques
- 5 Vues

Le prédicat CASE

Le prédicat CASE permet de retourner un choix conformément à une alternative.

Sa syntaxe est la suivante :

```
CASE
  WHEN condition_1 THEN expression_1
  [WHEN condition_2 THEN expression_2]
  [...]
  [WHEN condition_n THEN expression_n]
  [ELSE expression_else]
END
```

Le prédicat CASE

Exemple

Exemple avec le prédicat CASE

```
SELECT CASE  
    WHEN departement=75 THEN "Ile-de-France"  
    WHEN departement=93 THEN "Ile-de-France"  
    WHEN departement=69 THEN "Auvergne-Rhône-Alpes"  
    WHEN departement=971 THEN "Guadeloupe"  
    ...  
END
```

Plan

- 1 Requête élémentaire
- 2 Jointures
- 3 Expressions et Fonctions
- 4 Agrégation et statistiques**
 - Agrégation
 - Statistiques
- 5 Vues
- 6 Ajout, Modification et suppression de valeurs

Plan

- 1 Requête élémentaire
- 2 Jointures
- 3 Expressions et Fonctions
- 4 Agrégation et statistiques**
 - Agrégation
 - Statistiques
- 5 Vues
- 6 Ajout, Modification et suppression de valeurs

Fonctions d'agrégat

On appelle fonction d'agrégat une fonction qui retourne une valeur après avoir agrégé un ensemble de résultats.

Le langage SQL propose 5 fonctions d'agrégat :

COUNT retourne le *décompte* des valeurs d'un attribut.

SUM retourne la *somme* des valeurs d'un attribut.

AVG retourne la *moyenne* des valeurs d'un attribut.

MIN retourne la valeur *minimum* des valeurs d'un attribut.

MAX retourne la valeur *maximum* des valeurs d'un attribut.

Fonctions d'agrégat

Exemples

Exemple de fonction COUNT

```
SELECT COUNT(*)  
FROM Patient;
```

Exemple de fonction AVG

```
SELECT AVG(age)  
FROM Patient  
WHERE diagnostic="J11.1";
```

Exemple de fonction MAX

```
SELECT MAX(PAS), MAX(PAD)  
FROM Examen_Clinique  
WHERE id_patient=14;
```

Plan

- 1 Requête élémentaire
- 2 Jointures
- 3 Expressions et Fonctions
- 4 Agrégation et statistiques**
 - Agrégation
 - **Statistiques**
- 5 Vues
- 6 Ajout, Modification et suppression de valeurs

Regrouper par catégories

Le langage SQL permet d'utiliser les fonctions d'agrégat non pas sur toute la table, mais sur des sous-catégories d'un attribut donné.

Le prédicat à utiliser est GROUP BY.

Sa syntaxe est la suivante :

`GROUP BY attribut_1, attribut_2, ..., attribut_n`

Les résultats sont regroupés par combinaison des valeurs des attributs *attribut_1*, *attribut_2*, ..., *attribut_n*.

Filtrer les groupes d'agrégats

Après que les groupes d'agrégats ont été créés, il est possible d'appliquer des filtres pour en supprimer certains. Cels se fait avec le prédicat HAVING.

Sa syntaxe est la suivante :

```
HAVING condition_agregat
```

Filtrer les groupes d'agrégats

Exemple

Afficher la pression artérielle diastolique maximum par âge ayant un IMC moyen entre 20 et 25

```
SELECT MAX(pression__arterielle__diastolique)
FROM Mesure
GROUP BY age
HAVING AVG(IMC) BETWEEN 20 AND 25 ;
```

Format général d'une requête SELECT

La requête SELECT a donc pour format général :

```
SELECT [DISTINCT] liste_attributs  
FROM liste_tables  
[WHERE expression_logique]  
[GROUP BY liste_attributs]  
[HAVING expression_logique]  
[ORDER BY liste_attributs]
```

Plan

- 1 Requête élémentaire
- 2 Jointures
- 3 Expressions et Fonctions
- 4 Agrégation et statistiques
- 5 Vues**
- 6 Ajout, Modification et suppression de valeurs

Principe des vues

Définition

Une **vue** est une requête sauvegardée. Elle s'utilise comme une table.

La création d'une vue se fait avec la requête suivante :

```
CREATE VIEW nom_vue[liste_attributs]  
AS requete;
```

Vues

Exemple

Vue qui donne le nombre de consultations par patient

```
CREATE VIEW nombre_visites  
AS SELECT IPP, count(*) AS qte  
FROM Consultation  
GROUP BY IPP;
```

Vue qui donne la pression artérielle diastolique maximum par sexe, par age et par ville

```
CREATE VIEW evolution_PAD  
AS SELECT sexe, age, code_postal, max(PAD) AS PAD_max  
FROM Consultation  
GROUP BY sexe, age, code_postal;
```

Plan

- 1 Requête élémentaire
- 2 Jointures
- 3 Expressions et Fonctions
- 4 Agrégation et statistiques
- 5 Vues
- 6 Ajout, Modification et suppression de valeurs**
 - Ajout de données
 - Modification
 - Suppression

Plan

- 1 Requête élémentaire
- 2 Jointures
- 3 Expressions et Fonctions
- 4 Agrégation et statistiques
- 5 Vues
- 6 Ajout, Modification et suppression de valeurs**
 - Ajout de données
 - Modification
 - Suppression

Ajout de données

La requête INSERT permet d'ajouter des tuples à la base de données.

Sa syntaxe est la suivante :

```
INSERT INTO nom_table(attribut_1, attribut_2, ...,
attribut_n)
VALUES (valeur11, valeur21, ..., valeurn1)
[, (valeur12, valeur22, ..., valeurn2)]
...
(valeur1m, valeur2m, ..., valeurnm)
```

Ajout de données

Exemple

Exemple d'ajout de données

```
INSERT INTO Patient(IPP, nom, prenom, date_naissance)  
VALUES (1482442, "BEAULIEU", "Marguerite", "1963-12-06"),  
(162429684, "KLAINY", "Jacques", "1972-01-27");
```

Plan

- 1 Requête élémentaire
- 2 Jointures
- 3 Expressions et Fonctions
- 4 Agrégation et statistiques
- 5 Vues
- 6 Ajout, Modification et suppression de valeurs**
 - Ajout de données
 - **Modification**
 - Suppression

Requête de modification de valeur

La requête UPDATE permet de mettre à jour la valeur d'un ou plusieurs attributs de certains tuples de la base de données.

Sa syntaxe est la suivante :

```
UPDATE nom_table
SET attribut_1=valeur_1,
  [attribut_2=valeur_2,
  ...
  attribut_n=valeur_n]
[WHERE expression_logique];
```

Requête de modification de valeur

Exemple

Ajout d'un diagnostic d'hypertension artérielle

```
UPDATE Diagnostic  
SET HTA=TRUE  
WHERE PAS>=140 OR PAD>=90 ;
```

Plan

- 1 Requête élémentaire
- 2 Jointures
- 3 Expressions et Fonctions
- 4 Agrégation et statistiques
- 5 Vues
- 6 Ajout, Modification et suppression de valeurs**
 - Ajout de données
 - Modification
 - **Suppression**

Requête de suppression de valeur

La requête DELETE permet de supprimer des tuples depuis une table de la base de données.

Sa syntaxe est la suivante :

```
DELETE FROM nom_table  
[WHERE expression_logique];
```

Requête de suppression de valeur

Exemple

Suppression des patients nés avant le 1^{er} Janvier 1920

```
DELETE FROM Patient  
WHERE date_naissance < '1920-01-01' ;
```