Module 4 : Tri et limitation des résultats

ACADYS

Objectifs pédagogiques

À la fin de ce module, vous serez capable de :

- Trier les résultats d'une requête SQL avec ORDER BY
- Utiliser l'ordre de tri ascendant et descendant (ASC, DESC)
- Trier les résultats selon plusieurs colonnes
- Limiter le nombre de résultats renvoyés par une requête
- Utiliser LIMIT/TOP pour l'implémentation de la pagination

1. Tri des résultats avec ORDER BY

1.1 Rôle et importance du tri

Le tri des résultats d'une requête SQL est essentiel pour plusieurs raisons :

- Améliorer la lisibilité des données
- Faciliter l'analyse en regroupant les valeurs similaires
- Présenter les données dans un ordre logique pour les utilisateurs
- Préparer les données pour des rapports ou des affichages paginés

Sans un tri explicite, l'ordre des résultats d'une requête SQL n'est pas garanti et peut varier selon le SGBD et le plan d'exécution choisi.

1.2 Syntaxe de base d'ORDER BY

La clause ORDER BY se place après les clauses SELECT, FROM et WHERE (si présente) :

```
SELECT colonne1, colonne2, ...
FROM table
WHERE condition
ORDER BY colonne1;
```

Par défaut, le tri est effectué en ordre ascendant (du plus petit au plus grand, de A à Z).

1.3 Exemples simples

```
-- Tri des clients par nom de famille
SELECT client_id, nom, prenom
FROM Clients
ORDER BY nom;

-- Tri des produits par prix
SELECT produit_id, nom_produit, prix
FROM Produits
ORDER BY prix;
```

```
-- Tri des commandes par date (de la plus ancienne à la plus récente)
SELECT commande_id, date_commande
FROM Commandes
ORDER BY date_commande;
```

2. Direction du tri : ASC et DESC

2.1 Tri ascendant (ASC)

Le tri ascendant est le comportement par défaut d'ORDER BY. Il peut être spécifié explicitement avec le mot-clé ASC :

```
-- Tri des clients par nom en ordre ascendant (A à Z)
SELECT client_id, nom, prenom
FROM Clients
ORDER BY nom ASC;
```

Caractéristiques du tri ascendant :

- Valeurs numériques : du plus petit au plus grand
- Chaînes de caractères : ordre alphabétique (A à Z)
- Dates : de la plus ancienne à la plus récente
- NULL : généralement placés au début (dépend du SGBD)

2.2 Tri descendant (DESC)

Pour trier en ordre inverse, utilisez le mot-clé DESC :

```
-- Tri des produits par prix du plus cher au moins cher
SELECT produit_id, nom_produit, prix
FROM Produits
ORDER BY prix DESC;

-- Tri des commandes par date (de la plus récente à la plus ancienne)
SELECT commande_id, date_commande
FROM Commandes
ORDER BY date_commande DESC;
```

Caractéristiques du tri descendant :

- Valeurs numériques : du plus grand au plus petit
- Chaînes de caractères : ordre alphabétique inverse (Z à A)
- Dates : de la plus récente à la plus ancienne
- NULL : généralement placés à la fin (dépend du SGBD)

2.3 Mélange d'ordres de tri dans une même requête

Vous pouvez spécifier ASC ou DESC pour chaque colonne indépendamment :

```
-- Clients classés par ville (A-Z) puis par nom (Z-A)
SELECT client_id, nom, prenom, ville
FROM Clients
ORDER BY ville ASC, nom DESC;
```

3. Tri sur plusieurs colonnes

3.1 Principe du tri multi-colonnes

Le tri sur plusieurs colonnes permet d'ordonner les résultats selon une hiérarchie de critères. Lorsque deux enregistrements ont la même valeur pour le premier critère de tri, le second critère est utilisé pour les départager, et ainsi de suite.

3.2 Syntaxe du tri multi-colonnes

Les colonnes sont spécifiées dans l'ordre de priorité, séparées par des virgules :

```
SELECT colonne1, colonne2, ...
FROM table
ORDER BY colonne1, colonne2, colonne3;
```

3.3 Exemples de tri multi-colonnes

```
-- Clients triés par ville, puis par nom, puis par prénom
SELECT client_id, nom, prenom, ville
FROM Clients
ORDER BY ville, nom, prenom;

-- Produits triés par catégorie, puis par prix décroissant
SELECT produit_id, nom_produit, categorie_id, prix
FROM Produits
ORDER BY categorie_id, prix DESC;

-- Commandes triées par client, puis par date de commande (récentes d'abord)
SELECT commande_id, client_id, date_commande
FROM Commandes
ORDER BY client_id, date_commande DESC;
```

3.4 Cas d'usage typiques

Le tri multi-colonnes est particulièrement utile pour :

- Présenter des données groupées logiquement (ex: produits par catégorie)
- Créer des listes alphabétiques hiérarchisées (ex: annuaire par ville, nom, prénom)
- Organiser des données chronologiques par entité (ex: commandes par client et date)

- Générer des rapports complexes avec plusieurs niveaux de tri

4. Tri sur expressions et colonnes calculées

4.1 Tri sur expressions

Vous pouvez trier les résultats selon des expressions ou des fonctions, pas seulement selon des colonnes :

```
-- Tri des produits selon la valeur en stock (prix × quantité)
SELECT produit_id, nom_produit, prix, stock, prix * stock AS valeur_stock
FROM Produits
ORDER BY prix * stock DESC;

-- Tri des clients par longueur du nom
SELECT client_id, nom, prenom
FROM Clients
ORDER BY LENGTH(nom);
```

4.2 Tri sur colonnes calculées

Vous pouvez aussi trier selon une colonne calculée définie dans la clause SELECT :

```
-- Tri selon la colonne calculée (avec alias)
SELECT produit_id, nom_produit, prix, stock, prix * stock AS valeur_stock
FROM Produits
ORDER BY valeur_stock DESC;
```

4.3 Tri selon la position de la colonne

Dans certains SGBD, vous pouvez trier selon la position de la colonne dans la liste SELECT :

```
-- Tri selon la 3ème colonne (prix)
SELECT produit_id, nom_produit, prix, stock
FROM Produits
ORDER BY 3 DESC;
```

Note : Cette syntaxe est déconseillée car elle rend le code moins lisible et plus fragile face aux modifications de la requête.

5. Limitation des résultats

5.1 Pourquoi limiter les résultats ?

Limiter le nombre de résultats d'une requête présente plusieurs avantages :

- Amélioration des performances (moins de données à transférer)

- Implémentation de la pagination dans les applications
- Affichage des N premiers/derniers éléments (Top N)
- Optimisation des ressources système et réseau

5.2 Syntaxe LIMIT (MySQL, PostgreSQL, SQLite)

La clause LIMIT spécifie le nombre maximal de lignes à retourner :

```
 -- Retourner les 10 premiers produits
SELECT *
FROM Produits
LIMIT 10;
```

Pour la pagination, LIMIT est souvent combiné avec OFFSET :

```
 -- Retourner les produits 11 à 20 (page 2 avec 10 éléments par page)
SELECT *
FROM Produits
LIMIT 10 OFFSET 10;

-- Syntaxe alternative pour MySQL
SELECT *
FROM Produits
LIMIT 10, 10;  -- LIMIT offset, count
```

5.3 Syntaxe TOP (SQL Server)

SQL Server utilise la clause TOP dans la partie SELECT :

```
 -- Retourner les 10 premiers produits
SELECT TOP 10 *
FROM Produits;

-- Avec pourcentage (retourne 10% des lignes)
SELECT TOP 10 PERCENT *
FROM Produits;

-- Avec égalité (inclut les lignes avec des valeurs égales au seuil)
SELECT TOP 10 WITH TIES *
FROM Produits
ORDER BY prix DESC;
```

5.4 Autres syntaxes selon les SGBD

Oracle (avant 12c) :

```
-- Retourner les 10 premiers produits
SELECT *
FROM Produits
WHERE ROWNUM <= 10;
```

Oracle (12c et plus récent) :

```
-- Retourner les 10 premiers produits
SELECT *
FROM Produits
FETCH FIRST 10 ROWS ONLY;

-- Avec OFFSET pour la pagination
SELECT *
FROM Produits
OFFSET 10 ROWS FETCH NEXT 10 ROWS ONLY;
```

DB2 :

```
-- Retourner les 10 premiers produits
SELECT *
FROM Produits
FETCH FIRST 10 ROWS ONLY;
```

6. Combinaison du tri et de la limitation

6.1 Importance de l'ordre des clauses

Lorsque vous combinez ORDER BY et LIMIT, l'ordre des clauses est crucial :

1. SELECT : sélection des colonnes
2. FROM : source des données
3. WHERE : filtrage des lignes
4. ORDER BY : tri des résultats
5. LIMIT : limitation du nombre de résultats

Si vous inversez cet ordre, la requête générera une erreur de syntaxe.

6.2 Exemples de tri et limitation combinés

```
-- Les 5 produits les plus chers
SELECT produit_id, nom_produit, prix
FROM Produits
ORDER BY prix DESC
LIMIT 5;
```

```
-- Les 10 commandes les plus récentes
SELECT commande_id, client_id, date_commande
FROM Commandes
ORDER BY date_commande DESC
LIMIT 10;

-- Les 3 clients ayant le plus de commandes
SELECT c.client_id, c.nom, c.prenom, COUNT(commande_id) AS nombre_commandes
FROM Clients c
JOIN Commandes co ON c.client_id = co.client_id
GROUP BY c.client_id, c.nom, c.prenom
ORDER BY nombre_commandes DESC
LIMIT 3;
```

6.3 Application à la pagination

La pagination est une technique courante dans les applications web pour afficher de grandes quantités de données par "pages" :

```
 -- Page 1 (éléments 1-10)
SELECT * FROM Produits ORDER BY nom_produit LIMIT 10 OFFSET 0;

-- Page 2 (éléments 11-20)
SELECT * FROM Produits ORDER BY nom_produit LIMIT 10 OFFSET 10;

-- Page 3 (éléments 21-30)
SELECT * FROM Produits ORDER BY nom_produit LIMIT 10 OFFSET 20;

-- Formule générale : LIMIT nombre_par_page OFFSET (page - 1) * nombre_par_page
```

6.4 Bonnes pratiques pour la pagination

- Toujours combiner LIMIT/OFFSET avec ORDER BY pour garantir un ordre cohérent entre les pages
- Utiliser un tri sur une colonne unique ou une combinaison de colonnes qui garantit un ordre total
- Pour les grandes tables, optimiser les requêtes de pagination avec des index appropriés
- Envisager des techniques de "pagination par curseur" pour les très grands ensembles de données

7. Considérations de performance

7.1 Impact du tri sur les performances

Le tri peut être coûteux en termes de ressources, particulièrement sur de grandes tables :

- Les tris peuvent nécessiter des opérations de tri en mémoire ou sur disque

- L'utilisation d'index peut améliorer significativement les performances de tri
- Le tri sur plusieurs colonnes peut être plus lent qu'un tri simple

7.2 Optimisation du tri avec les index

Les index peuvent accélérer considérablement les opérations de tri :

```
 -- Création d'un index qui accélère le tri par prix
CREATE INDEX idx_produits_prix ON Produits(prix);

-- Création d'un index composite pour le tri multi-colonnes
CREATE INDEX idx_clients_ville_nom ON Clients(ville, nom);
```

7.3 Efficacité de la limitation des résultats

La limitation des résultats améliore généralement les performances :

- Moins de données à traiter et à transférer
- Possibilité pour certains SGBD d'optimiser l'exécution (early termination)
- Réduction de la charge mémoire côté client et serveur

Prochaine étape

Dans le prochain module, nous explorerons les fonctions d'agrégation basiques (COUNT, SUM, AVG, MIN, MAX) qui vous permettront d'effectuer des calculs statistiques simples sur vos données.