Result Control with LIMIT and OFFSET

LEARN

1. Introduction to LIMIT Clause

The LIMIT clause in SQL is used to constrain the number of rows returned in the result set. It's especially useful when you need only a subset of data for reporting, previewing, or debugging purposes. This clause enhances performance by reducing the volume of data transferred or processed.

Why Use LIMIT?

- To preview a specific number of records.
- To restrict large result sets for performance optimization.
- To extract top N rows based on a condition or order.

Syntax:

```
SELECT column1, column2
FROM table_name
LIMIT number;
```

Example:

```
SELECT *
FROM Students
LIMIT 5;
```

This fetches the first 5 records from the Students table.

> Note: The order of rows without an ORDER BY clause is not guaranteed.

2. Using OFFSET to Skip Rows

The OFFSET clause is used in conjunction with LIMIT to skip a defined number of rows before starting to return the result set. OFFSET is most commonly used in data pagination.

Syntax:

```
SELECT column1, column2
FROM table_name
LIMIT number OFFSET offset_value;
```

- number: Total number of rows to fetch.
- offset_value: Number of rows to skip from the beginning.

Example:

```
SELECT *
FROM Students
LIMIT 5 OFFSET 10;
```

This retrieves 5 rows starting from the 11th row (since OFFSET is zero-based).

> OFFSET works well with ORDER BY to maintain consistency across pages.

---

3. Pagination with LIMIT and OFFSET

Pagination allows users to navigate large datasets efficiently by dividing data into manageable "pages." It's commonly used in web applications to show results page-by-page.

Use Case:

Assume 10 rows per page. Here's how to fetch different pages:

- Page 1: OFFSET = 0
- Page 2: OFFSET = 10
- Page 3: OFFSET = 20

General Pagination Formula:

```
OFFSET = (PageNumber - 1) * RowsPerPage
```

SQL Example (Page 3 with 10 rows per page):

```
SELECT *
FROM Students
ORDER BY StudentID
LIMIT 10 OFFSET 20;
```

> Ensuring ORDER BY on a unique column like StudentID guarantees consistent results across pages.

Advantages of Using LIMIT and OFFSET

- Enhances application responsiveness.
- Supports UI pagination logic.
- Optimizes network traffic and server load.

Limitations:

- OFFSET can become inefficient for large offsets, as the database must still scan and skip many rows.
- Some databases have alternative mechanisms (like keyset pagination or cursors) for large-scale performance.

PRACTISE

Task 1: Fetch Top 3 Highest-Paid Employees

```
SELECT *
FROM Employees
ORDER BY Salary DESC
LIMIT 3;
```

Fetches the top 3 earners.

Task 2: Retrieve Records 11 to 20 from Products

```
SELECT *
FROM Products
ORDER BY ProductID
LIMIT 10 OFFSET 10;
```

This returns the second set of 10 rows (page 2) using sorting for consistency.

Task 3: Simulate Page 3 of Customer List (5 rows per page)

```
SELECT *
FROM Customers
ORDER BY CustomerID
LIMIT 5 OFFSET 10;
```

Shows rows 11 to 15.

Task 4: Show the Last 5 Students (Reverse Order)

```
SELECT *
FROM Students
```

```
ORDER BY EnrollmentDate DESC
LIMIT 5;
```

Displays the latest 5 students enrolled.

FAQ

- **Q:** What is the default starting point when using LIMIT?
  - **A:** OFFSET is 0 by default, meaning retrieval starts from the first row.
- **Q:** What happens if OFFSET is beyond the number of rows?
  - **A:** The query returns zero rows as there are no more rows to return.
- **Q:** Can LIMIT be used without OFFSET?
  - **A:** Yes. OFFSET is optional and is only needed if skipping initial rows is required.
- **Q:** Can I combine LIMIT and OFFSET with ORDER BY?
  - **A:** Yes, and it's highly recommended to ensure deterministic row selection.
- **Q:** Is LIMIT supported by all databases?
  - **A:** Most relational databases support LIMIT (MySQL, PostgreSQL). SQL Server uses TOP or OFFSET FETCH.
- **Q:** Are there performance considerations?
  - **A:** Yes. Large OFFSET values can degrade performance. Use indexed columns in ORDER BY and consider keyset pagination for large datasets.
