IT CAREER HUB

Урок 10.2. Оконные функции смещения и выбора

Оконные функции смещения и выбора

 **Важно!**
База данных с доступом на чтение:
hostname: ich-db.edu.itcareerhub.de
username: ich1
password: password

 **Оконные функции смещения и выбора — это функции, позволяющие выполнять вычисления, которые зависят от значений в других строках в пределах определенного окна.**

Эти функции полезны для сравнения значений строк с их предыдущими или последующими значениями, а также для выполнения более сложных аналитических задач.

Рассмотрим следующие оконные функции смещения и выбора:

1. `LEAD()`
2. `LAG()`
3. `FIRST_VALUE()`
4. `LAST_VALUE()`
5. `NTH_VALUE()`

IT CAREER HUB

Общий синтаксис

```
Unset
SELECT column1,
       window_function(column2) OVER (
           PARTITION BY column3
           ORDER BY column4 ) AS result_column
FROM table_name;
```

- `window_function(column2)`: Оконная функция смещения или выбора, которая применяется.
- `PARTITION BY column3`: Разделяет данные на группы, если требуется.
- `ORDER BY column4`: Устанавливает порядок строк в рамках окна.

IT CAREER HUB

Основные оконные функции смещения и выбора

LEAD() — это функция, которая возвращает значение из следующей строки в пределах окна, которое определяется ORDER BY.

Пример использования функции LEAD()

Задача: Найти значение `SaleAmount` следующей продажи по дате.

```
Unset
SELECT SaleID, SaleDate, SaleAmount,
       LEAD(SaleAmount, 1) OVER (ORDER BY SaleDate) AS NextSaleAmount
FROM Sales;
```

Входные данные

SaleID	SaleDate	SaleAmount
1	2024-01-01	100
2	2024-01-02	150
3	2024-01-03	200

Результат

SaleID	SaleDate	SaleAmount	NextSaleAmount
1	2024-01-01	100	150
2	2024-01-02	150	200
3	2024-01-03	200	NULL

`LEAD(SaleAmount, 1)` возвращает значение `SaleAmount` из следующей строки, упорядоченной по `SaleDate`.

LAG() — это функция, которая возвращает значение из предыдущей строки в пределах окна.

Пример использования функции LAG()

Задача: Найти значение `SaleAmount` предыдущей продажи по дате.

```
Unset
SELECT SaleID, SaleDate, SaleAmount,
       LAG(SaleAmount, 1) OVER (ORDER BY SaleDate) AS PrevSaleAmount
FROM Sales;
```

Входные данные

SaleID	SaleDate	SaleAmount
1	2024-01-01	100
2	2024-01-02	150
3	2024-01-03	200

Результат

SaleID	SaleDate	SaleAmount	PrevSaleAmount
1	2024-01-01	100	NULL
2	2024-01-02	150	100
3	2024-01-03	200	150

`LAG(SaleAmount, 1)` возвращает значение `SaleAmount` из предыдущей строки, упорядоченной по `SaleDate`.

FIRST_VALUE() — это функция, которая возвращает первое значение в рамках окна.

IT CAREER HUB

Пример использования функции FIRST_VALUE()

Задача: Найти первую цену продажи для каждого месяца.

```
Unset
SELECT OrderID, OrderDate, UnitPrice,
        FIRST_VALUE(UnitPrice) OVER (PARTITION BY EXTRACT(YEAR FROM OrderDate),
EXTRACT(MONTH FROM OrderDate) ORDER BY OrderDate) AS FirstPriceOfMonth
FROM OrderDetails;
```

Входные данные

OrderID	OrderDate	UnitPrice
1	2024-01-01	10
2	2024-01-05	20
3	2024-02-01	15
4	2024-02-15	25

Результат

OrderID	OrderDate	UnitPrice	FirstPriceOfMonth
1	2024-01-01	10	10
2	2024-01-05	20	10
3	2024-02-01	15	15
4	2024-02-15	25	15

`FIRST_VALUE(UnitPrice)` возвращает первое значение `UnitPrice` в рамках окна, упорядоченного по дате заказа.

LAST_VALUE() — это функция, которая возвращает последнее значение в рамках окна.

IT CAREER HUB

Пример использования функции LAST_VALUE()

Задача: Найти последнюю цену продажи для каждого месяца.

```
Unset
SELECT OrderID, OrderDate, UnitPrice,
        LAST_VALUE(UnitPrice) OVER (PARTITION BY EXTRACT(YEAR FROM OrderDate),
EXTRACT(MONTH FROM OrderDate) ORDER BY OrderDate) AS LastPriceOfMonth
FROM OrderDetails;
```

Входные данные

OrderID	OrderDate	UnitPrice
1	2024-01-01	10
2	2024-01-05	20
3	2024-02-01	15
4	2024-02-15	25

Результат

OrderID	OrderDate	UnitPrice	LastPriceOfMonth
1	2024-01-01	10	20
2	2024-01-05	20	20
3	2024-02-01	15	25
4	2024-02-15	25	25

`LAST_VALUE(UnitPrice)` возвращает последнее значение `UnitPrice` в пределах окна.

NTH_VALUE() — это функция, которая возвращает n-е значение в рамках окна.

IT CAREER HUB

Пример использования функции NTH_VALUE()

Задача: Найти третью цену продажи для каждого месяца.

```
Unset
SELECT OrderID, OrderDate, UnitPrice,

        NTH_VALUE(UnitPrice, 3) OVER (PARTITION BY EXTRACT(YEAR FROM OrderDate),
EXTRACT(MONTH FROM OrderDate) ORDER BY OrderDate) AS ThirdPriceOfMonth

FROM OrderDetails;
```

Входные данные

OrderID	OrderDate	UnitPrice
1	2024-01-01	10
2	2024-01-05	20
3	2024-01-10	30
4	2024-02-01	15
5	2024-02-15	25
6	2024-02-20	35

Результат

OrderID	OrderDate	UnitPrice	ThirdPriceOfMonth
1	2024-01-01	10	30
2	2024-01-05	20	30
3	2024-01-10	30	30
4	2024-02-01	15	35
5	2024-02-15	25	35
6	2024-02-20	35	35

`NTH_VALUE(UnitPrice, 3)` возвращает третье значение `UnitPrice` в рамках окна, упорядоченного по дате заказа.

IT CAREER HUB

Практическая работа

1. Из таблицы `purchase_orders` для каждого поставщика `supplier_id` выведите дату создания заказа, а также дату создания предыдущего заказа. Посчитайте разницу между этим датами.

2. Измените предыдущий запрос таким образом, чтобы узнать среднее время между двумя заказами.

3. Напишите аналогичный второму задания запрос, но с использованием функции `LEAD`. Сравните результаты.

4. Найдите самую раннюю дату `submitted_date` для каждого менеджера `created_by`. Решите данное задание использую оконные функции `MIN` и `FIRST_VALUE`. Сравните результаты.

Решение

1. Из таблицы `purchase_orders` для каждого поставщика `supplier_id` выведите дату создания заказа, а также дату создания предыдущего заказа. Посчитайте разницу между этим датами.

```
Unset
SELECT supplier_id, creation_date,
LAG(creation_date) OVER (partition by supplier_id ORDER BY creation_date) as previous_created_date,
datediff(creation_date, LAG(creation_date) OVER (partition by supplier_id ORDER BY creation_date)) as date_dif
FROM purchase_orders
```

2. Измените предыдущий запрос таким образом, чтобы узнать среднее время между двумя заказами.

```
Unset
SELECT AVG(date_dif)
FROM
(SELECT supplier_id, creation_date, LAG(creation_date) OVER (partition by supplier_id ORDER BY creation_date) as previous_created_date,
datediff(creation_date, LAG(creation_date) OVER (partition by supplier_id ORDER BY creation_date)) as date_dif
FROM purchase_orders) a
```

3. Напишите аналогичный второму задания запрос, но с использованием функции LEAD. Сравните результаты.

```
Unset
SELECT AVG(date_dif)
FROM
(SELECT supplier_id, creation_date, LAG(creation_date) OVER (partition by supplier_id ORDER BY creation_date) as previous_created_date,
datediff(creation_date, LEAD(creation_date) OVER (partition by supplier_id ORDER BY creation_date)) as date_dif
FROM purchase_orders) a
```

4. Найдите самую раннюю дату `submitted_date` для каждого менеджера `created_by`. Решите данное задание используя оконные функции `MIN` и `FIRST VALUE`. Сравните результаты.

```
Unset
SELECT created_by,
MIN(submitted_date) OVER (PARTITION BY created_by),
FIRST_VALUE (submitted_date) OVER (PARTITION BY created_by ORDER BY
submitted_date)
FROM purchase_orders
```