**ROLL UP**

ROLL UP - это конструкция, которая используется в Microsoft SQL для выполнения иерархической агрегации данных. Она позволяет вычислять агрегаты не только для каждого значения в столбцах GROUP BY, но также для иерархически связанных значений. ROLL UP выполняет последовательную агрегацию, начиная с самого мелкого уровня, затем переходит к более крупному и более крупному, пока не достигнет наивысшего уровня иерархии.

Пример использования конструкции ROLL UP:

SELECT region, country, city, SUM(sales)

FROM sales

GROUP BY ROLL UP(region, country, city);

Этот запрос вычисляет сумму продаж для каждого города в каждой стране в каждом регионе. Однако с помощью конструкции ROLL UP мы можем также вычислить суммы продаж для каждой страны в каждом регионе и общую сумму продаж для каждого региона. То есть, в итоге мы получим иерархически связанные данные, начиная с наименьшего уровня, а заканчивая наивысшим уровнем:

region country city sales

North America Canada Toronto $500,000

North America USA New York $1,000,000

North America USA Los Angeles $800,000

North America USA NULL $1,800,000

North America NULL NULL $2,300,000

Europe France Paris $600,000

Europe Germany Berlin $700,000

Europe Germany Munich $500,000

Europe NULL NULL $1,800,000

NULL NULL NULL $4,100,000

Обратите внимание, что в итоговых строках, где уровень иерархии меньше, чем указанный в ROLL UP, значения будут равны NULL.

**CUBE**

CUBE - это конструкция, которая используется в Microsoft SQL для выполнения многомерной агрегации данных. Она позволяет вычислять агрегаты для всех возможных комбинаций значений в столбцах GROUP BY, что делает ее более гибкой, чем конструкция ROLL UP.

Пример использования конструкции CUBE:

SELECT region, country, city, SUM(sales)

FROM sales

GROUP BY CUBE(region, country, city);

Этот запрос вычисляет сумму продаж для каждого города в каждой стране в каждом регионе, а также для каждой комбинации значений, например для каждой страны в каждом регионе, для каждого города в каждой стране, для каждого региона и т.д. То есть, в итоге мы получим все возможные комбинации значений:

region country city sales

North America Canada Toronto $500,000

North America Canada Vancouver $400,000

North America USA New York $1,000,000

North America USA Los Angeles $800,000

North America USA NULL $1,800,000

North America NULL NULL $2,700,000

Europe France Paris $600,000

Europe Germany Berlin $700,000

Europe Germany Munich $500,000

Europe NULL NULL $1,800,000

NULL NULL NULL $4,500,000

Обратите внимание, что конструкция CUBE создает значительно больше строк, чем конструкция ROLL UP. Также, в итоговых строках, где уровень иерархии меньше, чем указанный в CUBE, значения будут равны NULL.

**UNION и UNION ALL**

UNION - это оператор, который используется в Microsoft SQL для объединения результатов двух и более SELECT запросов в один набор результатов. Он позволяет объединять результаты из разных таблиц или представлений с одинаковой структурой.

Пример использования оператора UNION:

SELECT column1, column2, column3

FROM table1

UNION

SELECT column1, column2, column3

FROM table2;

В этом примере оператор UNION объединяет результаты двух SELECT запросов и выводит один набор результатов, содержащий все строки из обоих таблиц. Обратите внимание, что для того, чтобы использовать оператор UNION, структура и количество столбцов в обоих SELECT запросах должны быть одинаковыми. В противном случае, оператор UNION выдаст ошибку.

Также можно использовать оператор UNION ALL, который объединяет результаты SELECT запросов без удаления дубликатов строк. Он может быть более быстрым, чем обычный UNION, но требует большего внимания при выборе данных, чтобы избежать дублирования строк.

Пример использования оператора UNION ALL:

SELECT column1, column2, column3

FROM table1

UNION ALL

SELECT column1, column2, column3

FROM table2;

В этом примере оператор UNION ALL также объединяет результаты двух SELECT запросов, но не удаляет дублирующиеся строки. В результате, набор результатов может содержать дублирующиеся строки.

**INTERSECT**

INTERSECT - это оператор, который используется в Microsoft SQL для получения пересечения двух и более наборов строк. Он возвращает только те строки, которые встречаются в обоих или нескольких наборах строк.

Пример использования оператора INTERSECT:

SELECT column1, column2, column3

FROM table1

INTERSECT

SELECT column1, column2, column3

FROM table2;

В этом примере оператор INTERSECT возвращает только те строки, которые присутствуют в обоих таблицах. Если какой-либо столбец в таблицах содержит разные значения для одинаковых строк, то эти строки будут исключены из результата.

Обратите внимание, что для использования оператора INTERSECT, наборы строк, которые вы хотите пересечь, должны иметь одинаковое количество столбцов и те же имена столбцов. Если это не так, вы получите ошибку.

Также стоит отметить, что оператор INTERSECT может быть менее эффективным, чем операторы UNION и JOIN, особенно если в таблицах много данных. Поэтому, перед использованием оператора INTERSECT, стоит оценить необходимость его использования и возможность использования более эффективных операторов.

**EXCEPT**

EXCEPT - это оператор, который используется в Microsoft SQL для получения разности двух наборов строк. Он возвращает только те строки, которые присутствуют в первом наборе строк, но отсутствуют во втором наборе.

Пример использования оператора EXCEPT:

SELECT column1, column2, column3

FROM table1

EXCEPT

SELECT column1, column2, column3

FROM table2;

В этом примере оператор EXCEPT возвращает только те строки, которые присутствуют в таблице table1, но отсутствуют в таблице table2.

Обратите внимание, что для использования оператора EXCEPT, наборы строк, которые вы хотите вычесть, должны иметь одинаковое количество столбцов и те же имена столбцов. Если это не так, вы получите ошибку.

Также стоит отметить, что оператор EXCEPT может быть менее эффективным, чем операторы UNION и JOIN, особенно если в таблицах много данных. Поэтому, перед использованием оператора EXCEPT, стоит оценить необходимость его использования и возможность использования более эффективных операторов.