Подзапросы (вложенные запросы)

В языке SQL реализована возможность создания **подзапросов** или другими словами вложенных запросов, то есть возможность создания такого запроса, который будет помещен внутрь другого запроса.
Обсудим необходимость использования подзапросов на простом примере, допустим нам необходимо вывести **фамилию, имя и номер группы** студентов, которые получают **максимальную стипендию**. Казалось бы, что написать такой SQL-запрос не составит особого труда, например, так:

```
SELECT LastName, FirstName, GroupName, Grants
FROM Students AS S, Groups AS G
WHERE G.Id = S.GroupId
GROUP BY LastName, FirstName, GroupName, Grants
HAVING Grants = MAX(Grants);
```

LastName	FirstName	GroupName	Grants
Brown	Jessica	32PPS11	1100.00
Johnson	Joshua	31PPS11	1100.00
Jones	Jack	29PR21	1256.00
Miller	Harry	29PR21	1100.00
Taylor	Emily	32PR31	1100.00
Thomas	Charlie	32PR31	1256.00

Однако полученный результат будет далек от ожидаемого, в данном случае мы получили максимальное значение стипендии для каждой группы данных. Такой результат выполнения SQL-запроса легко объясним. В данном случае данные сгруппированы таким образом, что формируют уникальные группы на основании фамилии и имени студента, названии группы и стипендии, а в операторе HAVING сравнивается значение стипендии для каждой группы данных с максимальным значением стипендии той же группы, то есть само с собой, тем самым возвращая истину. Группы, у которых максимальная стипендия имеет неопределенное значение (Grants = NULL) отбрасываются, так как сравнение на NULL-значение возможно только с использованием ключевого слова IS NULL.
Для того чтобы наш SQL-запрос вернул требуемые результаты, необходимо сравнивать стипендии студентов с максимальным значением стипендии (в нашем случае оно равно 1256). Однако, как вы знаете из предыдущего раздела, использование функций агрегирования в операторе WHERE запрещено, поэтому запрос можно написать следующим образом:

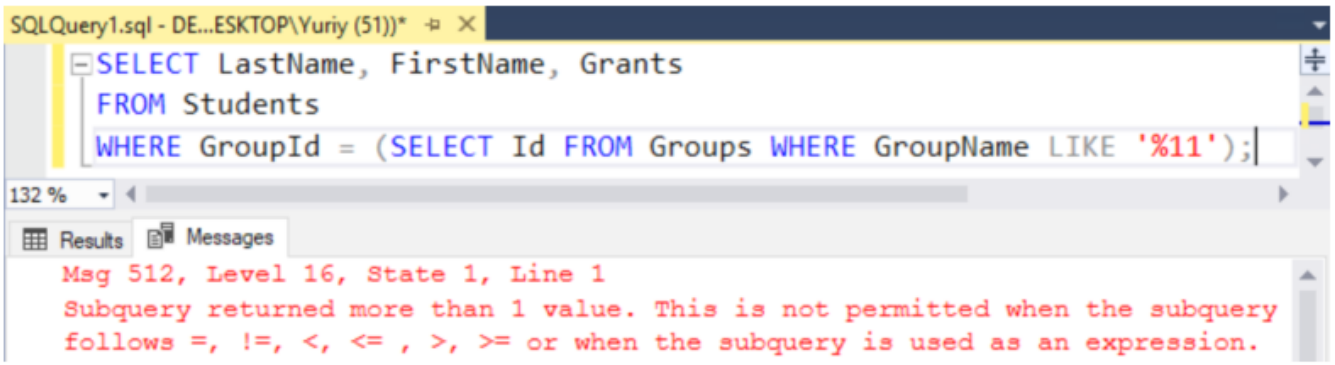
```
SELECT LastName, FirstName, GroupName, Grants
FROM Students AS S, Groups AS G
WHERE G.Id = S.GroupId AND Grants = 1256;
```

LastName	FirstName	GroupName	Grants
Jones	Jack	29PR21	1256.00
Thomas	Charlie	32PR31	1256.00

Несмотря на то, что в результате выполнения этого SQL-запроса мы получили правильный результирующий набор, такой подход не выдерживает никакой критики, ведь максимальное значение стипендии может меняться, что в свою очередь приведет к необходимости изменения «магического числа». Поэтому для получения максимального значения стипендии целесообразно использовать соответствующий подзапрос (SELECT MAX(Grants) FROM Students). Полученное в результате его выполнения значение будет сравниваться со стипендией в каждой записи студентов и в случае совпадения значений, именно эта запись будет добавляться в результирующую таблицу. Перепишем предыдущий SQL-запрос с использованием подзапроса, результат его выполнения будет таким же:

```
SELECT LastName, FirstName, GroupName, Grants
FROM Students AS S, Groups AS G
WHERE G.Id = S.GroupId AND Grants = (SELECT MAX(Grants) FROM Students);
```

Дополняя комментарии к данному запросу, следует добавить, что в соответствии с синтаксисом языка SQL подзапросы необходимо помещать в круглые скобки. В предыдущем примере мы использовали подзапрос, который возвращал скалярное значение, как результат выполнения функции агрегирования MAX(). Однако SQL- запросы в основном возвращают множество строк и в этом случае при использовании их в качестве подзапросов существует своя особенность. Например, нам необходимо получить информацию обо всех студентах, которые учатся в группах с номером 11 при этом нам неважно на каком потоке. В этом случае подзапрос выполнится правильно и вернет неопределенное количество значений одного столбца (Id), но неправильное сравнение возвращаемых им результатов приведет к ошибке.



Данная ошибка связана с тем, что наш подзапрос возвращает уникальные идентификаторы нескольких групп, которые соответствуют заданному условию (GroupName LIKE ‘%11’), а в операторе WHERE возвращаемые результаты сравниваются с единственным значением столбца GroupId, что в принципе невозможно (подобная ситуация уже описывалась в текущем уроке при использовании функций агрегирования). Поэтому во всех случаях, когда запрос может вернуть несколько значений, для их сравнения вместо логических операторов следует использовать ключевое слово IN:

```
SELECT LastName, FirstName, GroupId
FROM Students
WHERE GroupId IN (SELECT Id FROM Groups WHERE GroupName LIKE '%11');
```

LastName	FirstName	GroupId
Evans	Grace	2
Wilson	Lily	2
Johnson	Joshua	3
Brown	Jessica	5

Как многотабличные запросы, так и подзапросы можно отнести к категории «сложных» запросов. Однако если многотабличные запросы возвращают данные, полученные в результате соединения нескольких реально существующих таблиц, то подзапросы формируют виртуальные значения, которые используются для сравнения с реально существующими данными. Подзапросы позволяют обеспечить значительную гибкость при выполнении запросов, потому что их можно использовать не только в операторе WHERE (что было уже продемонстрировано), но и в операторах SELECT, FROM и HAVING.
В следующем SQL-запросе мы продемонстрируем возможность использования подзапросов в операторе HAVING. При помощи этого запроса мы получим список преподавателей, среднее значение месяца рождения которых больше среднего значения месяца рождения студентов:

```
SELECT LastName, FirstName
FROM Teachers
GROUP BY LastName, FirstName
HAVING AVG(MONTH(BirthDate))>(SELECT AVG(MONTH(BirthDate)) FROM Students);
```

LastName	FirstName
Williams	Daniel
Cooper	Michael
Nelson	Sophia

В этом случае подзапрос возвращает среднее значение месяцев рождения студентов, которое в операторе HAVING сравнивается со средним значением месяца рождения каждого преподавателя. Принцип работы подзапросов В том случае если в SQL-запросе применяются подзапросы, то выполнение SQL-операторов в первую очередь происходит в подзапросе, а уже потом полученные результаты используются в основном запросе. В одном SQL-запросе может быть несколько подзапросов при этом они могут быть вложены друг в друга, в этом случае выполнение начинается с подзапроса, который имеет наиболее глубокое вложение. Продемонстрируем это при помощи следующего SQL-запроса, который выведет название групп, студенты которых получают максимальную стипендию:

```
SELECT GroupName
FROM Groups
WHERE Id IN (SELECT GroupId FROM Students WHERE Grants = (SELECT MAX(Grants) FROM
Students));
```

GroupName
29PR21
32PR31

В данном случае первым выполнится подзапрос (SELECT MAX(Grants) FROM Students), который вернет размер максимальной стипендии. Следует заметить, что существует еще один вид подзапросов, которые называются связанными или коррелированными подзапросами. Особенностью выполнения таких подзапросов является их зависимость от значений в основном запросе и поэтому они не могут быть обработаны раньше, чем основной запрос, для лучшего понимания принципа их использования приведем пример. Допустим нам необходимо вывести максимальную оценку, полученную студентами по каждому предмету. Такую информацию можно получить, написав SQL-запрос и по-другому, но мы в данном случае решили продемонстрировать использование связанного подзапроса в операторе SELECT. Необходимо заметить, что обязательным условием такого использования является возврат подзапросом одного значения, для этих целей идеально подходят функции агрегирования:

```
SELECT Subjects.Name,(SELECT MAX(A.Assesment)
FROM Achievements AS A
WHERE Subjects.Id = A.SubjectId) AS Maximum FROM Subjects;
```

Name	Maximum
C#	10
Discrete Math	NULL
SQL Server	10
ADO.NET	11
ITE1	NULL
JavaScript	NULL
WIN10	NULL

Выполнение текущего связанного подзапроса зависит от сравнения уникального идентификатора предмета (Subjects.Id = A.SubjectId), значение которого изменяется по мере построчного прочтения записей в таблице Subjects, поэтому количество строк в полученной виртуальной таблице соответствует количеству записей в таблице Subjects.

Практическая работа

1. Вывести названия отделений, что находятся в том же корпусе, что и отделение “Cardiology”.
2. Вывести названия отделений, что находятся в том же корпусе, что и отделения “Gastroenterology” и “General Surgery”.
3. Вывести название отделения, которое получило меньше всего пожертвований.
4. Вывести фамилии врачей, ставка которых больше, чем у врача “Thomas Gerada”.
5. Вывести названия палат, вместимость которых больше, чем средняя вместимость в палатах отделения “Microbiology”.
6. Вывести полные имена врачей, зарплаты которых (сумма ставки и надбавки) превышают более чем на 100 зарплату врача “Anthony Davis”.
7. Вывести названия отделений, в которых проводит обследования врач “Joshua Bell”.
8. Вывести названия спонсоров, которые не делали пожертвования отделениям “Neurology” и “Oncology”.
9. Вывести фамилии врачей, которые проводят обследования в период с 12:00 до 15:00.

Саманчук Елена Дмитриевна
samancuk_e@itstep.academy