Объединение

Объединение результатов запросов применяется в том случае если необходимо получить совокупность строк двух и более запросов, которые выполняются независимо друг от друга. Для того чтобы результаты запросов можно было объединить они должны соответствовать определенным условиям совместимости:

• **количество столбцов** в каждом запросе должно быть одинаковым;

• **типы данных** соответствующих столбцов во всех запросах должны быть совместимы.

Также при объединении запросов существует ряд особенностей:

• в результирующем наборе, полученном при объединении, будут использоваться **имена столбцов**, которые были указаны в первом запросе;

• нельзя сортировать каждый запрос по отдельности, осуществить сортировку можно только всего составного запроса, указав **по его окончанию оператор ORDER BY**.

Ключевое слово **UNION** (объединение по вертикали) позволяет объединить результаты запросов и применяется в том случае, когда в результирующем наборе необходимо исключить повторяющиеся строки.

```
SELECT FirstName + ` ` + LastName AS FullName, BirthDate
FROM Students
WHERE MONTH(BirthDate) > 5 AND MONTH(BirthDate) < 9
UNION
SELECT FirstName + ` ` + LastName, BirthDate
FROM Teachers
WHERE  MONTH(BirthDate) > 5 AND MONTH(BirthDate) < 9
ORDER BY BirthDate;
```

Данный **SQL**-запрос будет выполняться следующим образом: вначале выполнятся два запроса, которые вернут полное имя и дату рождения студентов и преподавателей, родившихся летом, соответственно. После этого результаты запросов объединяются вместе без повторяющихся записей при помощи ключевого слова **UNION**, полученный вследствие этого результирующий набор будет отсортирован по возрастанию даты рождения с использованием оператора **ORDER BY**.

FullName	BirthDate
Daniel Williams	1979-07-30
Grace Evans	1997-06-24
Oliver Moore	1997-07-05
Jessica Brown	1997-07-17

Саманчук Елена Дмитриевна
samancuk_e@itstep.academy

В свою очередь оператор **JOIN** (объединение по горизонтали) позволяет объединять таблицы в пределах одного **SQL**-запроса. Существуют различные виды объединений, которые можно выполнить при помощи оператора **JOIN**: внутреннее (**INNER**) и внешние (**LEFT, RIGHT, FULL**).

Внутреннее объединение двух таблиц возможно при помощи оператора **INNER JOIN** с указанием условия их объединения (предиката) после ключевого слова **ON**. При выполнении оператора **INNER JOIN** каждая запись из первой таблицы сопоставляется с каждой записью из другой таблицы по условию, указанному после оператора **ON**, если условие выполняется, тогда строки записываются в результирующую таблицу, которая формируется путем конкатенации строк первой и второй таблиц.

```
SELECT *
FROM Groups INNER JOIN Students
ON Groups.Id = Students.GroupId;
```

В данном **SQL**-запросе после оператора **SELECT** специально прописан символ *, для того чтобы мы смогли увидеть все полученные столбцы.

Groups

Id	GroupName	FacultyId
1	29PR21	1
1	29PR21	1
2	30PR11	2
2	30PR11	2
3	31PPS11	1
4	32PR31	2
4	32PR31	2
4	32PR31	2
5	32PPS11	3

Students

Id	FirstName	LastName	BirthDate	Grants	Email	GroupId
1	Jack	Jones	1997-11-05	1256.00	jj@net.eu	1
2	Harry	Miller	1998-02-11	1100.00	hm@net.eu	1
3	Grace	Evans	1997-06-24	NULL	eg@net.eu	2
4	Lily	Wilson	1998-09-12	NULL	lw@net.eu	2
5	Joshua	Johnson	1997-05-23	1100.00	jo@net.eu	3
6	Emily	Taylor	1997-12-27	1100.00	et@net.eu	4
7	Charlie	Thomas	1998-01-31	1256.00	ct@net.eu	4
8	Oliver	Moore	1997-07-05	NULL	om@net.eu	4
9	Jessica	Brown	1997-07-17	1100.00	jb@net.eu	5

Как вы можете заметить, мы получили результирующую таблицу, которая фактически состоит из значений двух таблиц **Groups** и **Students**, связанных между собой уникальным идентификатором группы (**Groups.Id**). Естественно, что использовать результаты запроса в таком виде не имеет никакого смысла, однако мы можем указать в операторе **SELECT** только необходимые нам столбцы, и в результирующей таблице будут именно их значения в указанной нами последовательности:

```
SELECT LastName, FirstName, Email, GroupName
FROM Groups INNER JOIN Students
ON Groups.Id = Students.GroupId;
```

LastName	FirstName	Email	GroupName
Jones	Jack	jj@net.eu	29PR21
Miller	Harry	hm@net.eu	29PR21
Evans	Grace	eg@net.eu	30PR11
Wilson	Lily	lw@net.eu	30PR11
Johnson	Joshua	jo@net.eu	31PPS11
Taylor	Emily	et@net.eu	32PR31
Thomas	Charlie	ct@net.eu	32PR31
Moore	Oliver	om@net.eu	32PR31
Brown	Jessica	jb@net.eu	32PPS11

При использовании оператора **INNER JOIN** допускается не писать служебное слово **INNER**, в этом случае JOIN будет расцениваться как внутреннее объединение. Как вы, наверное, догадались при помощи оператора **INNER JOIN** можно осуществлять объединение нескольких таблиц.

```
SELECT FirstName, LastName, Name AS Subject, Assesment, GroupName
FROM Groups AS G JOIN Students AS S ON G.Id = S.GroupId
        JOIN Achievements AS A ON S.Id = A.StudentId
        JOIN Subjects AS Sb ON Sb.Id = A.SubjectId;
```

При выполнении данного **SQL**-запроса нам потребовалось объединить таблицы **Groups** и **Students** по идентификатору группы (**G.Id = S.GroupId**), **Achievements** и **Students** связав их по идентификатору студента (**S.Id = A.StudentId**) и таблицы **Subjects** и **Achievements** – по идентификатору предмета (**Sb.Id = A.SubjectId**).

FirstName	LastName	Subject	Assesment	GroupName
Jack	Jones	ADO.NET	11	29PR21
Harry	Miller	C#	10	29PR21
Jack	Jones	SQL Server	10	29PR21
Grace	Evans	C#	8	30PR11
Harry	Miller	ADO.NET	9	29PR21
Grace	Evans	SQL Server	7	30PR11

При осуществлении внутреннего объединения вы, как и раньше, можете использовать все операторы языка **SQL**. Например, если потребуется получить оценки по предметам только у студентов **29** потока и отсортировать полученную информацию по фамилиям студентов, тогда нам достаточно применить операторы **WHERE** и **ORDER BY** к написанному ранее запросу.

```
SELECT FirstName, LastName, Name AS Subject, Assesment, GroupName
FROM Groups AS G JOIN Students AS S ON G.Id = S.GroupId
        JOIN Achievements AS A ON S.Id = A.StudentId
        JOIN Subjects AS Sb ON Sb.Id = A.SubjectId
WHERE GroupName LIKE '29%'
ORDER BY LastName;
```

FirstName	LastName	Subject	Assesment	GroupName
Jack	Jones	ADO.NET	11	29PR21
Jack	Jones	SQL Server	10	29PR21
Harry	Miller	C#	10	29PR21
Harry	Miller	ADO.NET	9	29PR21

При выполнении **SQL**-запросов, которые связаны с получением полной информации из таблиц, использование внутреннего объединения не даст желаемого результата. Например, необходимо вывести информацию обо всех студентах и оценках, которые ими были получены.

```
SELECT FirstName + ' ' + LastName AS FullName, Assesment
FROM Students AS S INNER JOIN Achievements AS A ON S.Id = A.StudentId;
```

FullName	Assesment
Jack Jones	11
Harry Miller	10
Jack Jones	10
Grace Evans	8
Harry Miller	9
Grace Evans	7

Однако полученный нами результат не удовлетворяет требуемому условию, ведь нам необходимо было получить данные **обо всех студентах**, а не только о тех из них кто получил оценки. Такое поведение характерно при использовании оператора **INNER JOIN**, потому что в результирующую таблицу будут помещены только те записи, которые удовлетворяют условию, указанному после ключевого слова **ON**, в нашем случае соответствие уникального идентификатора студента (**S.Id = A.StudentId**).

Для того, чтобы получать правильные результаты при написании подобных **SQL**-запросов необходимо применять внешние объединения (**OUTER JOIN**), которые позволяют получить все данные из одной таблицы независимо от того существует ли для них совпадения строк в другой таблице. Существует три типа внешних объединений: левое (**LEFT**), правое (**RIGHT**) и полное (**FULL**). При написании запросов с использованием внешних объединений ключевое слово **OUTER** можно опустить, потому что ключевые слова **LEFT**, **RIGHT** и **FULL** однозначным образом определят его тип.

Оператор **LEFT JOIN (LEFT OUTER JOIN)** позволяет создать, так называемое левое внешнее объединение, при выполнении которого в результирующий набор помимо строк, удовлетворяющих условию после ключевого слова **ON**, будут записаны даже те строки из таблицы, расположенной слева от оператора, которые не удовлетворяют указанному условию.

Попробуем решить задачу обо всех студентах и их оценках при помощи оператора **LEFT JOIN**.

```
SELECT FirstName + ` ` + LastName AS FullName, Assesment
FROM Students AS S LEFT JOIN Achievements AS A ON S.Id = A.StudentId;
```

В данном случае в результирующий набор попадут данные обо всех студентах из таблицы Students, независимо от того получали они оценки или нет. В случае если в таблице Achievements отсутствует оценка для соответствующего студента, результат будет равен **NULL**.

FullName	Assesment
Jack Jones	11
Jack Jones	10
Harry Miller	10
Harry Miller	9
Grace Evans	8
Grace Evans	7
Lily Wilson	NULL
Joshua Johnson	NULL
Emily Taylor	NULL
Charlie Thomas	NULL
Oliver Moore	NULL
Jessica Brown	NULL

Допустим, нам необходимо получить информацию о студентах, которые пока не получали оценок. Одним из вариантов получения требуемых данных является **SQL**-запрос с обязательным использованием подзапроса.

```
SELECT FirstName + ' ' + LastName AS FullName
FROM Students
WHERE Id NOT IN (SELECT StudentId FROM Achievements);
```

В данном случае подзапрос возвращает уникальные идентификаторы всех студентов, которые получили оценки, и информация о которых находится в таблице **Achievements**. В основном запросе выводятся данные только о тех студентах, уникальный идентификатор которых отсутствует во множестве, сформированном подзапросом, то есть запись о них отсутствует в таблице **Achievements**, а значит они еще не получили ни одной оценки.

FullName
Lily Wilson
Joshua Johnson
Emily Taylor
Charlie Thomas
Oliver Moore
Jessica Brown

При написании подзапросов многие студенты сталкиваются с проблемой понимания их выполнения. Поэтому более изящным решением поставленной задачи без использования подзапроса является возможность применения в данном **SQL**-запросе оператора **LEFT JOIN**, который образует более понятную синтаксическую конструкцию. К тому же такой запрос выполнится быстрее, потому что для каждого **LEFT JOIN** создается отдельной поток и его выполнение происходит параллельно в отличие от подзапроса, выполнение которого осуществляется последовательно с основным запросом в одном потоке. Для того чтобы понять, как решить поставленную задачу при помощи внешнего объединения, вам необходимо внимательно присмотреться к результату выполнения **SQL**-запроса. Как вы заметили, у некоторых студентов оценка имеет неопределенное значение (**NULL**), то есть если отфильтровать записи по этому признаку, мы получим требуемый результат, который полностью совпадает с результатом, который был получен при выполнении предыдущего запроса.

```
SELECT FirstName + ' ' + LastName AS FullName
FROM Students AS S LEFT JOIN Achievements AS A ON S.Id = A.StudentId
WHERE Assesment IS NULL;
```

Саманчук Елена Дмитриевна
samancuk_e@itstep.academy

При помощи оператора **RIGHT JOIN (RIGHT OUTER JOIN)** возможно создать, так называемое правое внешнее объединение, при выполнении которого в результирующий набор помимо строк, удовлетворяющих условию после ключевого слова **ON**, будут записаны все строки из таблицы, расположенной справа от оператора, независимо от того удовлетворяют они указанному условию или нет. В качестве первого примера использования оператора **RIGHT JOIN** продемонстрируем вам решение уже известной вам задачи — вывод информации обо всех студентах и полученных ими оценках.

```
SELECT FirstName + ' ' + LastName AS FullName, Assesment
FROM Achievements AS A RIGHT JOIN Students AS S ON S.Id = A.StudentId;
```

В данном случае мы поменяли местами таблицы **Achievements** и **Students** и в результате выполнения данного запроса мы получили результат идентичный результат. Таким образом, при помощи правого внешнего объединения можно получить результаты аналогичные результатам левого внешнего объединения, если изменить порядок следования таблиц в **SQL**-запросе.

В следующем примере мы продемонстрируем возможность совместного использования правого и левого объединений. Предположим, что нам необходимо получить список всех предметов и информацию о преподавателях, которые их читают.

```
SELECT [Name] AS [Subject], LastName, FirstName
FROM TeachersSubjects AS TS RIGHT JOIN Subjects AS S ON S.Id = TS.SubjectId LEFT JOIN
Teachers AS T ON T.Id = TS.TeacherId
ORDER BY [Name];
```

Таблица **TeachersSubjects** содержит информацию о том, кто из преподавателей какой предмет читает. В таблице **Subjects** находится список всех предметов, то есть именно строки из этой таблицы нам необходимо получить, независимого от того, имеются ли соответствия в других связанных с ней таблицах. Поэтому мы объединили таблицы **TeachersSubjects** и **Subjects** при помощи правого внешнего соединения. Для того чтобы сохранились результаты первого объединения мы, при помощи левого внешнего объединения, связываем их с таблицей Teachers, в которой содержится информация о преподавателях. Из полученного результата видно, что предмет **Unity** на данный момент не читает ни один из преподавателей.

Subject	LastName	FirstName
ADO.NET	Cooper	Michael
C#	Nelson	Sophia
Discrete Math	Kirk	Emma
ITE1	Williams	Daniel
JavaScript	Nelson	Sophia
SQL Server	MacAlister	Henry
Unity	NULL	NULL
WIN10	Williams	Daniel

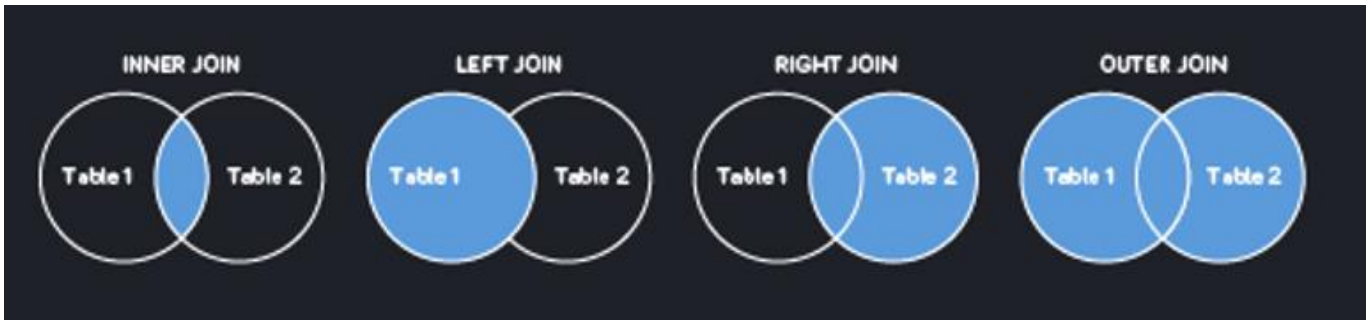
Саманчук Елена Дмитриевна
samancuk_e@itstep.academy

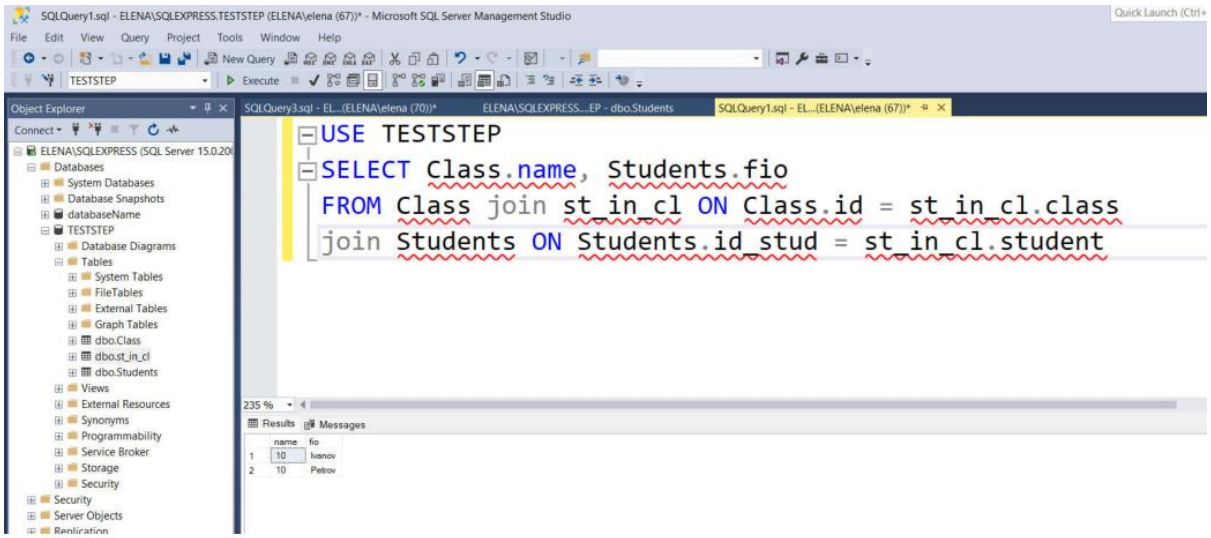
Оператор **FULL JOIN (FULL OUTER JOIN)** позволяет создать полное внешнее объединение, при котором в результирующий набор включаются записи из левой таблицы даже в том случае, если для них отсутствуют соответствующие записи в правой таблице, а записи из правой таблицы будут помещены в результирующую таблицу даже тогда, когда для них нет записей соответствия в левой таблице. Таким образом, полное внешнее соединение, по сути, является комбинацией левого и правого внешних объединений. Приведем пример выполнения полного внешнего объединения для получения информации обо всех студентах и группах.

```
SELECT FirstName, LastName, GroupName
FROM Students AS S FULL JOIN Groups AS G ON G.Id = S.GroupId
ORDER BY FirstName;
```

FirstName	LastName	GroupName
NULL	NULL	33GR12
Charlie	Thomas	32PR31
Charlotte	Becker	NULL
Emily	Taylor	32PR31
Grace	Evans	30PR11
Harry	Miller	29PR21
Jack	Jones	29PR21
Jessica	Brown	32PPS11
Joshua	Johnson	31PPS11
Lily	Wilson	30PR11
Oliver	Moore	32PR31

В полученной результирующей таблице находятся все записи из таблицы **Students** и все записи из таблицы **Groups** там, где отсутствует соответствующая запись из другой таблицы, возвращается неопределенное значение (**NULL**). В данном случае в группе **33GR12** нет ни одного студента и студентка **Charlotte Becker** пока не зачислена ни в одну из групп.





Практическая работа

1. Вывести названия и вместимости палат, расположенных в 3-м корпусе, вместимостью 3 и более мест, если в этом корпусе есть хотя бы одна палата вместимостью более 3 мест.

2. Вывести названия отделений в которых проводилось хотя бы одно обследование 12.00 до 14.00.

3. Вывести полные имена врачей, которые проводят обследования.

4. Вывести названия отделений, в которых проводятся обследования.

5. Вывести фамилии врачей, которые являются хирургами.

6. Вывести фамилию врача, премия которого больше, чем ставка остальных врачей.

7. Вывести названия палат в 3-м корпусе, чья вместимость больше.

8. Вывести фамилии врачей, проводящих обследования в отделениях хирургии и терапии.

9. Вывести названия отделений, в которых работают врачи.

10. Вывести полные имена врачей и отделения в которых они проводят обследования, если отделение имеет сумму пожертвований более 2000.

11. Вывести название отделения, в котором проводит обследования врач с наибольшей ставкой.

12. Вывести названия заболеваний и количество проводимых по ним обследований.

Саманчук Елена Дмитриевна
samancuk_e@itstep.academy