ТЕМА 11. ОПЕРАТОР SELECT

Оператор SELECT позволяет получить строки из таблицы или представления

Синтаксис

```
SELECT [ ALL | DISTINCT [ ON ( выражение [, ...] ) ] ]
    [ * | выражение [ [ AS ] имя_результата ] [, ...] ]
    [ FROM элемент_FROM [, ...] ]
    [ WHERE условие ]
    [ GROUP BY элемент_группирования [, ...] ]
    [ HAVING условие [, ...] ]
    [ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] выборка ]
    [ ORDER BY выражение [ ASC | DESC ] [ NULLS { FIRST | LAST } ] [, ...] ]
    [ LIMIT { число | ALL } ]
    [ OFFSET начало [ ROW | ROWS ] ]


    элемент_FROM [ NATURAL ] тип_соединения элемент_FROM [ ON условие_соединения | USING
( столбец_соединения [, ...] ) ]
```

и *элемент_группирования* может быть следующим:

```
    ( )
    выражение
    ( выражение [, ...] )
```

Порядок выполнения оператора

SELECT получает строки из множества таблиц (возможно, пустого). Общая процедура выполнения SELECT следующая:

1. Вычисляются все элементы в списке FROM. (Каждый элемент в списке FROM представляет собой реальную или виртуальную таблицу.) Если список FROM содержит несколько элементов, они объединяются перекрёстным соединением.
2. Если указано предложение WHERE, все строки, не удовлетворяющие условию, исключаются из результата.
3. Если присутствует указание GROUP BY, либо в запросе вызываются агрегатные функции, вывод разделяется по группам строк, соответствующим одному или нескольким значениям, а затем вычисляются результаты агрегатных функций. Если добавлено предложение HAVING, оно исключает группы, не удовлетворяющие заданному условию.
4. Вычисляются фактические выходные строки по заданным в SELECT выражениям для каждой выбранной строки или группы строк.
5. SELECT DISTINCT исключает из результата повторяющиеся строки. SELECT DISTINCT ON исключает строки, совпадающие по всем указанным выражениям. SELECT ALL (по умолчанию) возвращает все строки результата, включая дубликаты.
6. Операторы UNION, INTERSECT и EXCEPT объединяют вывод нескольких команд SELECT в один результирующий набор. Оператор UNION возвращает все строки, представленные в одном, либо обоих наборах результатов. Оператор INTERSECT возвращает все строки, представленные строго в обоих наборах. Оператор EXCEPT возвращает все строки, представленные в первом наборе, но не во втором. Во всех трёх случаях повторяющиеся строки исключаются из результата, если явно не указано ALL. Чтобы явно обозначить, что выдаваться должны только неповторяющиеся строки, можно добавить избыточное слово DISTINCT.
7. Если присутствует предложение ORDER BY, возвращаемые строки сортируются в указанном порядке. В отсутствие ORDER BY строки возвращаются в том порядке, в каком системе будет проще их выдать. (См. Предложение ORDER BY ниже.)
8. Если указано предложение LIMIT (или FETCH FIRST) либо OFFSET, оператор SELECT возвращает только подмножество строк результата.

Параметры

Предложение FROM

В предложении FROM перечисляются одна или несколько таблиц, служащих источниками данных для SELECT. Если указано несколько источников, результатом будет декартово произведение (перекрёстное соединение) всех их строк. Но обычно в запрос добавляются уточняющие условия (в предложении WHERE), которые ограничивают набор строк небольшим подмножеством этого произведения.

Предложение FROM может содержать следующие элементы:

имя_таблицы

Имя (возможно, дополненное схемой) существующей таблицы или представления.

псевдоним

Альтернативное имя для элемента списка FROM. Этот псевдоним используется для краткости или для исключения неоднозначности с замкнутыми соединениями (когда одна таблица читается неоднократно). Когда задаётся псевдоним, он полностью скрывает настоящее имя таблицы или функции; например, при записи FROM foo AS f, в продолжении запроса SELECT к этому элементу FROM нужно обращаться по имени f, а не foo. Если задан псевдоним таблицы, за ним можно также написать список псевдонимов столбцов, который определит альтернативные имена для столбцов таблицы.

выборка

Предложение FROM может содержать вложенный запрос SELECT. Можно считать, что из его результата создаётся временная таблица на время выполнения основной команды SELECT. Заметьте, что вложенный запрос SELECT должен заключаться в скобки и для него ***должен*** задаваться псевдоним. Здесь также можно использовать команду VALUES.

имя_функции

В предложении FROM могут содержаться вызовы функций. (Это особенно полезно для функций, возвращающих множества, но в принципе можно использовать любые функции.) Псевдоним для функции можно задать так же, как и для таблицы

тип_соединения

Один из следующих вариантов:

- `[ INNER ] JOIN`
- `LEFT [ OUTER ] JOIN`
- `RIGHT [ OUTER ] JOIN`
- `FULL [ OUTER ] JOIN`
- `CROSS JOIN`

Для типов соединений INNER и OUTER необходимо указать условие соединения, а именно одно из предложений NATURAL, ON *условие_соединения*или USING (*столбец_соединения* [, ...]). Эти предложения описываются ниже. Для CROSS JOIN ни одно из этих предложений не допускается.

Предложение JOIN объединяет два элемента списка FROM, которые мы для простоты дальше будем называть «таблицами», хотя на самом деле это может быть любой объект, допустимый в качестве элемента FROM. Для определения порядка вложенности при необходимости следует использовать скобки. В отсутствие скобок предложения JOIN обрабатывается слева направо. В любом случае, JOIN связывает элементы сильнее, чем запятые, разделяющие элементы в списке FROM.

CROSS JOIN и INNER JOIN формируют простое декартово произведение, то же, что можно получить, указав две таблицы на верхнем уровне FROM, но ограниченное возможным условием соединения. Предложение CROSS JOIN равнозначно INNER JOIN ON (TRUE), то есть, никакие строки по условию не удаляются. Эти типы соединений введены исключительно для удобства записи, они не дают ничего такого, что нельзя было бы получить, используя просто FROM и WHERE.

LEFT OUTER JOIN возвращает все строки ограниченного декартова произведения (т. е. все объединённые строки, удовлетворяющие условию соединения) плюс все строки в таблице слева, для которых не находится строк в таблице справа, удовлетворяющих условию. Строка, взятая из таблицы слева, дополняется до полной ширины объединённой таблицы значениями NULL в столбцах таблицы справа. Заметьте, что для определения, какие строки двух таблиц соответствуют друг другу, проверяется только условие самого предложения JOIN. Внешние условия проверяются позже.

RIGHT OUTER JOIN, напротив, возвращает все соединённые строки плюс одну строку для каждой строки справа, не имеющей соответствия слева (эта строка дополняется значениями NULL

влево). Это предложение введено исключительно для удобства записи, так как его можно легко свести к LEFT OUTER JOIN, поменяв левую и правую таблицы местами.

FULL OUTER JOIN возвращает все соединённые строки плюс все строки слева, не имеющие соответствия справа, (дополненные значениями NULL вправо) плюс все строки справа, не имеющие соответствия слева (дополненные значениями NULL влево).

ON *условие_соединения*

Задаваемое *условие_соединения* представляет собой выражение, выдающее значение типа boolean (как в предложении WHERE), которое определяет, какие строки считаются соответствующими при соединении.

USING (*столбец_соединения* [, ...])

Предложение вида USING (a, b, ...) представляет собой сокращённую форму записи ON таблица_слева.a = таблица_справа.a AND таблица_слева.b = таблица_справа.b .... Кроме того, USING подразумевает, что в результат соединения будет включён только один из пары равных столбцов, но не оба.

NATURAL

NATURAL представляет собой краткую запись USING со списком, в котором перечисляются все столбцы двух таблиц, имеющие одинаковые имена. Если одинаковых имён нет, указание NATURAL равнозначно ON TRUE.

Предложение WHERE

Необязательное предложение WHERE имеет общую форму

```
WHERE условие
```

, где *условие* — любое выражение, выдающее результат типа boolean. Любая строка, не удовлетворяющая этому условию, исключается из результата. Строка удовлетворяет условию, если оно возвращает true при подстановке вместо ссылок на переменные фактических значений из этой строки.

Предложение GROUP BY

Необязательное предложение GROUP BY имеет общую форму

```
GROUP BY элемент_группирования [, ...]
```

GROUP BY собирает в одну строку все выбранные строки, выдающие одинаковые значения для выражений группировки. В качестве *выражения* внутри *элемента_группирования* может выступать имя входного столбца, либо имя или порядковый номер выходного столбца (из списка элементов SELECT), либо произвольное значение, вычисляемое по значениям входных столбцов. В случае неоднозначности имя в GROUP BY будет восприниматься как имя входного, а не выходного столбца.

Агрегатные функции, если они используются, вычисляются по всем строкам, составляющим каждую группу, и в итоге выдают отдельное значение для каждой группы. (Если агрегатные функции используются без предложения GROUP BY, запрос выполняется как с одной группой, включающей все выбранные строки

Когда в запросе присутствует предложение GROUP BY или какая-либо агрегатная функция, выражения в списке SELECT не могут обращаться к негруппируемым столбцам, кроме как в агрегатных функциях или в случае функциональной зависимости, так как иначе в негруппируемом столбце нужно было бы вернуть более одного возможного значения. Функциональная зависимость образуется, если группируемые столбцы (или их подмножество) составляют первичный ключ таблицы, содержащей негруппируемый столбец.

Предложение HAVING

Необязательное предложение HAVING имеет общую форму

```
HAVING условие
```

Здесь *условие* задаётся так же, как и для предложения WHERE.

HAVING исключает из результата строки групп, не удовлетворяющих условию. HAVING отличается от WHERE: WHERE фильтрует отдельные строки до применения GROUP BY, а HAVING фильтрует строки групп, созданных предложением GROUP BY. Каждый столбец, фигурирующий в *условии*, должен однозначно ссылаться на группируемый столбец, за исключением случаев, когда эта ссылка находится внутри агрегатной функции или негруппируемый столбец функционально зависит от группируемых.

В присутствие HAVING запрос превращается в группируемый, даже если GROUP BY отсутствует. То же самое происходит, когда запрос содержит агрегатные функции, но не предложение GROUP BY. Все выбранные строки считаются формирующими одну группу, а в списке SELECT и предложении HAVING можно обращаться к столбцам таблицы только из агрегатных функций. Такой запрос будет выдавать единственную строку, если результат условия HAVING — true, и ноль строк в противном случае.

Список SELECT

Список SELECT (между ключевыми словами SELECT и FROM) содержит выражения, которые формируют выходные строки оператора SELECT. Эти выражения могут обращаться (и обычно обращаются) к столбцам, вычисленным в предложении FROM.

Так же, как в таблице, каждый выходной столбец SELECT имеет имя. В простом предложении SELECT это имя просто помечает столбец при выводе, но когда SELECT представляет собой подзапрос большого запроса, это имя большой запрос видит как имя столбца виртуальной таблицы, созданной подзапросом. Чтобы задать имя для выходного столбца, нужно написать AS *выходное_имя* после выражения столбца. (Слово AS можно опустить, но только если желаемое выходное имя не совпадает с каким-либо ключевым словом PostgreSQL Чтобы не зависеть от появления новых ключевых слов в будущем, рекомендуется всегда писать AS, либо заключать имя в двойные кавычки.) Если имя столбца не задать, PostgreSQLвыберет его автоматически.

По имени выходного столбца можно обратиться к его значению в предложениях ORDER BY и GROUP BY, но не в WHERE или HAVING; в них вместо имени надо записывать всё выражение.

Вместо выражения в выходном списке можно указать *, что будет обозначать все столбцы выбранных строк. Кроме того, можно записать *имя_таблицы*.* как краткое обозначение всех столбцов, получаемых из данной таблицы. В этих случаях нельзя задать новые имена столбцов с помощью AS; именами выходных столбцов будут имена столбцов в таблице.

Предложение DISTINCT

Если указано SELECT DISTINCT, все повторяющиеся строки исключаются из результирующего набора (из каждой группы дубликатов остаётся одна строка). SELECT ALL делает противоположное: сохраняет все строки; это поведение по умолчанию.

Предложение UNION+

Предложение UNION имеет следующую общую форму:

оператор_SELECT UNION [ALL | DISTINCT] *оператор_SELECT*

Здесь *оператор_SELECT* — это любой подзапрос SELECT без предложений ORDER BY, LIMIT . (ORDER BY и LIMIT можно добавить к вложенному выражению, если оно заключено в скобки. Без скобок эти предложения будут восприняты как применяемые к результату UNION, а не к выражению в его правой части.)

Оператор UNION вычисляет объединение множеств всех строк, возвращённых заданными запросами SELECT. Строка оказывается в объединении двух наборов результатов, если она присутствует минимум в одном наборе. Два оператора SELECT, представляющие прямые операнды UNION, должны выдавать одинаковое число столбцов, а типы соответствующих столбцов должны быть совместимыми.

Результат UNION не будет содержать повторяющихся строк, если не указан параметр ALL. ALL предотвращает исключение дубликатов. (Таким образом, UNION ALL обычно работает значительно быстрее, чем UNION; поэтому, везде, где возможно, следует указывать ALL.) DISTINCT можно записать явно, чтобы обозначить, что дублирующиеся строки должны удаляться (это поведение по умолчанию).

При использовании в одном запросе SELECT нескольких операторов UNION они вычисляются слева направо, если иной порядок не определяется скобками.

Предложение INTERSECT

Предложение INTERSECT имеет следующую общую форму:

```
оператор_SELECT INTERSECT [ ALL | DISTINCT ] оператор_SELECT
```

Здесь *оператор_SELECT* — это любой подзапрос SELECT без предложений ORDER BY, LIMIT.

Оператор INTERSECT вычисляет пересечение множеств всех строк, возвращённых заданными запросами SELECT. Строка оказывается в пересечении двух наборов результатов, если она присутствует в обоих наборах.

Результат INTERSECT не будет содержать повторяющихся строк, если не указан параметр ALL. С параметром ALL строка, повторяющаяся *m* раз в левой таблице и *n* раз в правой, будет выдана в результирующем наборе min(*m*,*n*) раз. DISTINCT можно записать явно, чтобы обозначить, что дублирующиеся строки должны удаляться (это поведение по умолчанию).

При использовании в одном запросе SELECT нескольких операторов INTERSECT они вычисляются слева направо, если иной порядок не диктуется скобками. INTERSECT связывает свои подзапросы сильнее, чем UNION. Другими словами, A UNION B INTERSECT C будет восприниматься как A UNION (B INTERSECT C).

Предложение EXCEPT

Предложение EXCEPT имеет следующую общую форму:

```
оператор_SELECT EXCEPT [ ALL | DISTINCT ] оператор_SELECT
```

Здесь *оператор_SELECT* — это любой подзапрос SELECT без предложений ORDER BY, LIMIT.

Оператор EXCEPT вычисляет набор строк, которые присутствуют в результате левого запроса SELECT, но отсутствуют в результате правого.

Результат EXCEPT не будет содержать повторяющихся строк, если не указан параметр ALL. С параметром ALL строка, повторяющаяся *m* раз в левой таблице и *n* раз в правой, будет выдана в результирующем наборе max(*m*-*n*,0) раз. DISTINCT можно записать явно, чтобы обозначить, что дублирующиеся строки должны удаляться (это поведение по умолчанию).

При использовании в одном запросе SELECT нескольких операторов EXCEPT они вычисляются слева направо, если иной порядок не диктуется скобками. EXCEPT связывает свои подзапросы так же сильно, как UNION.

Предложение ORDER BY

Необязательное предложение ORDER BY имеет следующую общую форму:

```
ORDER BY выражение [ ASC | DESC] [ NULLS { FIRST | LAST } ] [, ...]
```

Предложение ORDER BY указывает, что строки результата должны сортироваться согласно заданным выражениям. Если две строки дают равные значения для самого левого выражения, проверяется следующее выражение и т. д. Если их значения оказываются равными для всех заданных выражений, строки возвращаются в порядке, определяемом реализацией.

В качестве *выражения* может задаваться имя или порядковый номер выходного столбца (элемента списка SELECT), либо произвольное выражение со значениями входных столбцов.

Порядковым номером в данном случае считается последовательный номер (при нумерации слева направо) позиции выходного столбца. Возможность указать порядковый номер позволяет выполнить сортировку по столбцу, не имеющему уникального имени. В принципе это не абсолютно необходимо, так как выходному столбцу всегда можно присвоить имя, воспользовавшись предложением AS.

В предложении ORDER BY также можно использовать произвольные выражения, в том числе, и со столбцами, отсутствующими в списке результатов SELECT. Таким образом, следующий оператор вполне корректен:

```
SELECT name FROM distributors ORDER BY code;
```

Однако, если ORDER BY применяется к результату UNION, INTERSECT или EXCEPT, в нём можно задать только имя или номер выходного столбца, но не выражение.

Если в качестве выражения ORDER BY задано простое имя, которому соответствует и выходной, и входной столбец, то ORDER BY будет воспринимать его как имя выходного столбца. Этот выбор противоположен тому, что делает GROUP BY в такой же ситуации. Такая несогласованность допущена для соответствия стандарту SQL.

Дополнительно после любого выражения в предложении ORDER BY можно добавить ключевое слово ASC (по возрастанию) или DESC (по убыванию). По умолчанию подразумевается ASC.

Если указано NULLS LAST, значения NULL при сортировке оказываются после значений не NULL; с указанием NULLS FIRST значения NULL оказываются перед значениями не NULL. Если не указано ни то, ни другое, по умолчанию подразумевается NULLS LAST при явно или неявно выбранном порядке ASC, либо NULLS FIRST при порядке DESC (то есть по умолчанию считается, что значения NULL больше значений не NULL).

Заметьте, что параметры сортировки применяются только к тому выражению, за которым они следуют; в частности, ORDER BY x, y DESC означает не то же самое, что ORDER BY x DESC, y DESC.

Предложение LIMIT

Предложение LIMIT состоит из двух независимых вложенных предложений:

```
LIMIT { число | ALL }
OFFSET начало
```

Здесь *число* определяет максимальное количество строк, которое должно быть выдано, тогда как *начало* определяет, сколько строк нужно пропустить, прежде чем начать выдавать строки. Когда указаны оба значения, сначала строки пропускаются в количестве, заданном значением *начало*, а затем следующие строки выдаются в количестве, не превышающем значения *число*.

Если результатом выражения *число* оказывается NULL, предложение воспринимается как LIMIT ALL, т. е. число строк не ограничивается. Если *начало* принимает значение NULL, предложение воспринимается как OFFSET 0.

SQL:2008 вводит другой синтаксис для получения того же результата, и его так же поддерживает PostgreSQL. Он выглядит так:

```
OFFSET начало { ROW | ROWS }
FETCH { FIRST | NEXT } [ число ] { ROW | ROWS } ONLY
```

В этом синтаксисе, чтобы подставить в *начало* или *число* что-либо сложнее простой целочисленной константы, необходимо заключить это выражение в скобки. Если *число* опускается в предложении FETCH, оно принимает значение 1. ROW и ROWS так же, как и FIRST и NEXT являются избыточными словами, которые не влияют не действие этих предложений. Согласно стандарту, предложение OFFSET должно предшествовать предложению FETCH, если присутствуют они оба; но PostgreSQL менее строг и допускает любой порядок.

Применяя LIMIT, имеет смысл использовать также предложение ORDER BY, чтобы строки результата выдавались в определённом порядке. Иначе будут возвращаться непредсказуемые подмножества строк запроса — вы можете запросить строки с десятой по двадцатую, но какой порядок вы имеете в виду? Порядок будет неизвестен, если не добавить ORDER BY.

Команда TABLE

Команда

```
TABLE имя
```

равнозначна

```
SELECT * FROM имя
```

Её можно применять в качестве команды верхнего уровня или как более краткую запись внутри сложных запросов. С командой TABLE могут использоваться только предложения WITH, UNION, INTERSECT, EXCEPT, ORDER BY, LIMIT, OFFSET, FETCH ; предложение WHERE и какие-либо формы агрегирования не поддерживаются.

Необязательное предложение FROM

PostgreSQL разрешает опустить предложение FROM. Это позволяет очень легко вычислять результаты простых выражений:

```
SELECT 2+2;

 ?column?
----------
        4
```

Некоторые другие базы данных SQL не допускают этого, требуя задействовать в SELECT фиктивную таблицу с одной строкой.

Заметьте, что если предложение FROM не указано, запрос не может обращаться ни к каким таблицам базы данных.