Основы базы данных
SQL
Урок 3

2020 г.

Summary of second week

1. SELECT ALL
2. SELECT COLUMN_NAME
3. DISTINCT
4. WHERE (CONDITION)
5. AND, OR, NOT OPERATORS
6. NULL VALUES
7. LIKE
8. MIN(), MAX(), SUM(), AVG() AND COUNT() FUNCTIONS
9. IN AND BETWEEN OPERATORS
10. ORDER BY
11. GROUP BY
12. ALIAS

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

HAVING

Оператор **HAVING** является указателем результата выполнения агрегатных функций. Агрегатная функция в языке SQL называется функцией, возвращающей какое-либо одно значение по набору значений столбца. Такими функциями являются: COUNT (), MIN (), MAX (), AVG (), SUM ().

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

HAVING аналогичный оператору SQL WHERE за тем исключением, что применяется не для всего набора столбцов таблицы, а для набора созданного оператором SQL GROUP BY и применяется всегда строго после него.

> Оператор **HAVING** было добавлено в SQL, поскольку ключевое слово WHERE нельзя было использовать с агрегатными функциями.

Как HAVING работает в SQL

```sql
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

SELECT определяет столбцы.
FROM предоставляет набор потенциальных строк для результата.
WHERE дает фильтр для этих потенциальных строк.
GROUP BY делит строки в таблице на более мелкие группы.
HAVING дает фильтр для этих групповых строк.

Нужно различать между Where и Having в одной команде.
Where это команда которая фильтрует данные перед группировкой (Group by)
Having это команда которая фильтрует данные после группировки (Group by)

SQL JOINS

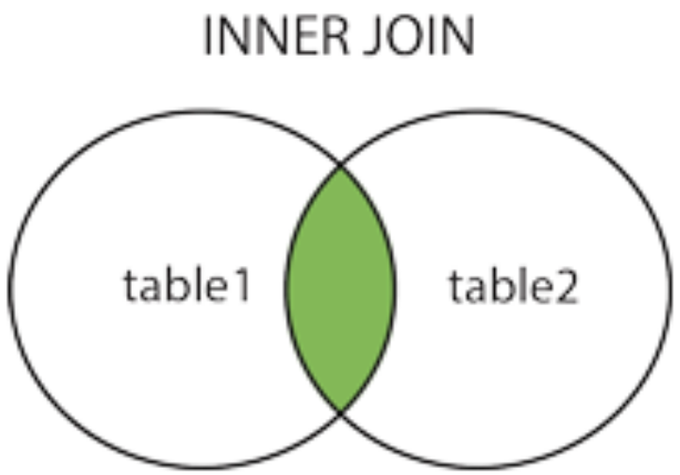
JOIN используется для объединения строк из двух или более таблиц на основе связанного столбца между ними.

Различные типы SQL JOIN

I**NNER JOIN:** возвращает записи, которые имеют совпадающие значения в обеих таблицах.
LEFT (OUTER) JOIN: возвращает все записи из левой таблицы и соответствующие записи из правой таблицы.
RIGHT (OUTER) JOIN: возвращает все записи из правой таблицы и соответствующие записи из левой таблицы.
FULL (OUTER) JOIN: возвращает все записи, если есть совпадение в левой или правой таблице

INNER JOIN

INNER JOIN выбирает записи, которые имеют совпадающие значения в обеих таблицах.



Синтаксис

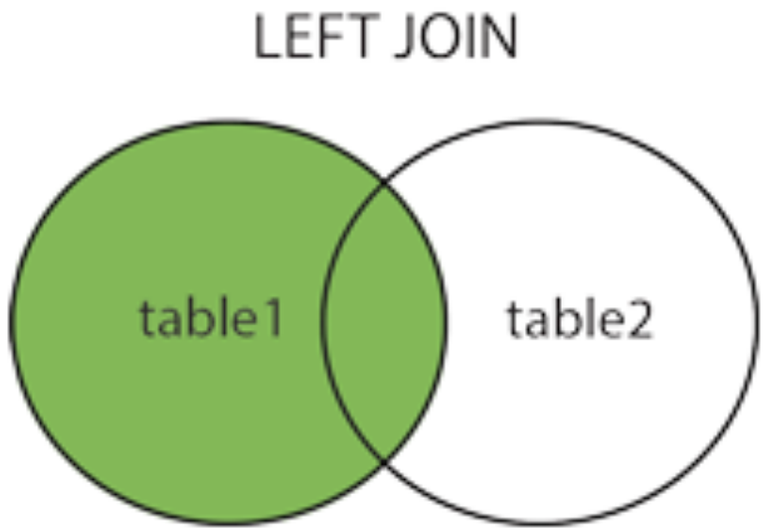
```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

> *INNER JOIN выбирает все строки из обеих таблиц до тех пор, пока существует соответствие между столбцами.*

LEFT JOIN

LEFT JOIN возвращает все записи из левой таблицы (table1) и соответствующие записи из правой таблицы (table2). Результатом будет NULL с правой стороны, если совпадения нет.

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```



Алгоритм работы LEFT JOIN следующий:

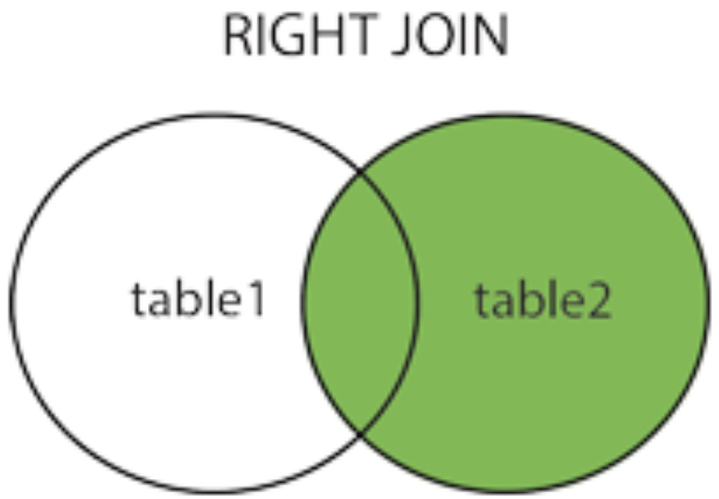
- Процесс происходит формирование таблицы внутренним соединением (оператор SQL INNER JOIN) левой и правой таблиц
- Затем в результате добавляются записи левой таблицы. Для них соответствующие записи из правой таблицы заполняются значениями NULL.

> *В некоторых базах данных LEFT JOIN называется LEFT OUTER JOIN.*

RIGHT JOIN

RIGHT JOIN возвращает все записи из правой таблицы (table2) и соответствующие записи из левой таблицы (table1). Результатом является NULL с левой стороны, когда нет совпадения.

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```



Алгоритм работы **RIGHT JOIN** следующий:

- Сначала происходит формирование таблицы внутренним соединением (оператор SQL INNER JOIN) левой и правой таблиц
- Затем, в результат добавляются записи левой таблицы не вошедшие в результат формирования таблицы внутренним соединением. Для них, соответствующие записи из левой таблицы заполняются значениями NULL.

> *В некоторых базах данных RIGHT JOIN называется RIGHT OUTER JOIN.*

Пример LEFT JOIN & RIGHT JOIN

Authors *— содержит в себе информацию об авторах книг*
Books *— содержит в себе информацию о названии книг:*

Authors

AuthorID	AuthorName
1	Bruce Eckel
2	Robert Lafore
3	Andrew

Books

BookID	BookName
3	Modern Operating
1	Thinking in Java
3	Computer Architecture
4	Programming in Scala

LEFT JOIN

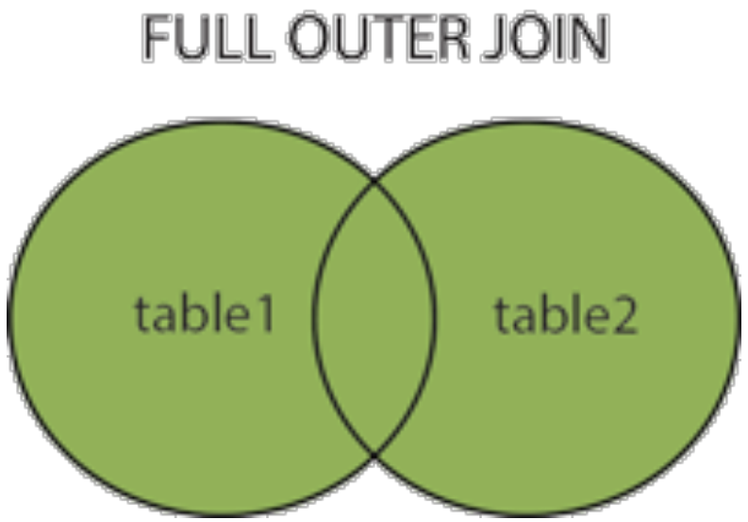
Authors.AuthorID	Authors.AuthorName	Books.BookID	Books.BookName
1	Bruce Eckel	1	Thinking in Java
2	Robert Lafore	NULL	NULL
3	Andrew Tanenbaum	3	Modern Operating System
3	Andrew Tanenbaum	3	Computer Architecture

RIGHT JOIN

Authors.AuthorID	Authors.AuthorName	Books.BookID	Books.BookName
3	Andrew Tanenbaum	3	Modern Operating System
1	Bruce Eckel	1	Thinking in Java
3	Andrew Tanenbaum	3	Computer Architecture
NULL	NULL	4	Programming in Scala

FULL JOIN

FULL JOIN осуществляет формирование таблицы из записей двух или нескольких таблиц. В операторе SQL FULL JOIN не важен порядок следования таблиц, он никак не влияет на окончательный результат, так как оператор является симметричным.



Синтаксис

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```

> *FULL JOIN можно воспринимать как сочетание операторов INNER JOIN + LEFT JOIN + RIGHT JOIN.*

Пример FULL JOIN

Authors *— содержит в себе информацию об авторах книг*
Books *— содержит в себе информацию о названии книг:*

Authors

AuthorID	AuthorName
1	Bruce Eckel
2	Robert Lafore
3	Andrew

Books

BookID	BookName
3	Modern Operating
1	Thinking in Java
3	Computer Architecture
4	Programming in Scala

FULL JOIN

Authors.AuthorID	Authors.AuthorName	Books.BookID	Books.BookName
1	Bruce Eckel	1	Thinking in Java
2	Robert Lafore	NULL	NULL
3	Andrew Tanenbaum	3	Modern Operating System
3	Andrew Tanenbaum	3	Computer Architecture
NULL	NULL	4	Programming in Scala

SELF JOIN

SELF JOIN - это обычное соединение, но таблица соединяется сама с собой.

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

T1 и T2 - разные псевдонимы для одной и той же таблицы.

SELF JOIN - можно рассматривать как соединение двух копий одной и той же таблицы. Таблица на самом деле не копируется, но SQL выполняет команду, как если бы она была.

Синтаксис команды присоединения таблицы к самой себе почти такой же, как у команды соединения двух разных таблиц. Чтобы отличать имена столбцов друг от друга, используются псевдонимы фактического имени таблицы, поскольку обе таблицы имеют одинаковое имя.

UNION

Оператор **UNION** используется для объединения набора результатов из двух или более SELECT.

Каждый SELECT в UNION должен иметь одинаковое количество столбцов. Столбцы также должны иметь похожие типы данных, также столбцы должны быть в одном порядке.

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

Оператор UNION по умолчанию выбирает только отдельные значения. Чтобы разрешить повторяющиеся значения, используется **UNION ALL**:

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

JOIN vs UNION

1) Столбцы объединяемых таблиц могут быть разными в **JOIN**, но в **UNION** количество столбцов и порядок столбцов во всех запросах должны быть одинаковыми.

2) **UNION** помещает строки из запросов друг за другом (ставит вертикально), но **JOIN** помещает столбцы из запросов один за другим (ставит по горизонтали)

3) Основное различие между **UNION** и **UNION ALL** заключается в том, что **UNION** удаляет повторяющиеся записи, а **UNION ALL** - нет.

SUBQUERY

Подзапрос(subquery) - это запрос, вложенный в другой оператор, в такие как SELECT, INSERT, UPDATE или DELETE.

```
SELECT    select_list
FROM      table
WHERE     expr operator
                (SELECT    select_list
                 FROM      table);
```

Подзапрос должен быть заключен в круглые скобки ( ).
Подзапросы не могут управлять своими результатами внутри, поэтому **ORDER BY** нельзя добавить в подзапрос. Можно использовать ORDER BY в основном операторе SELECT (внешний запрос)

Questions?