Построение вычисляемых полей

В общем случае для создания *вычисляемого (производного) поля* в списке `SELECT` следует указать некоторое *выражение* языка *SQL*. В этих выражениях применяются арифметические *операции* сложения, вычитания, умножения и деления, а также встроенные функции языка *SQL*. Можно указать имя любого столбца (поля) таблицы или запроса, но использовать имя столбца только той таблицы или запроса, которые указаны в списке предложения `FROM` соответствующей инструкции. При построении сложных выражений могут понадобиться скобки.

Стандарты *SQL* позволяют явным образом задавать имена столбцов результирующей таблицы, для чего применяется фраза `AS` .

Пример 6.1. Рассчитать общую *стоимость* для каждой сделки. Этот *запрос* использует расчет результирующих столбцов на основе арифметических выражений.

```
SELECT Товар.Название, Товар.Цена,
    Сделка.Количество,
    Товар.Цена*Сделка.Количество AS Стоимость
FROM Товар INNER JOIN Сделка
    ON Товар.КодТовара=Сделка.КодТовара

6.1. Расчет общей стоимости для каждой сделки.
```

Пример 6.2. Получить *список* фирм с указанием фамилии и инициалов клиентов.

```
SELECT Фирма, Фамилия+" "+
    Left(Имя,1)+"."+Left(Отчество,1)+"."AS ФИО
FROM Клиент

6.2. Получение списка фирм с указанием фамилии и инициалов клиентов.
```

В запросе использована *встроенная функция* `Left` , позволяющая вырезать в текстовой переменной один символ слева в данном случае.

Пример 6.3. Получить *список* товаров с указанием года и месяца продажи.

```
SELECT Товар.Название, Year(Сделка.Дата)
    AS Год, Month(Сделка.Дата) AS Месяц
FROM Товар INNER JOIN Сделка
    ON Товар.КодТовара=Сделка.КодТовара

6.3. Получение списка товаров с указанием года и месяца продажи.
```

В запросе использованы встроенные функции `Year` и `Month` для выделения года и месяца из даты.

Использование итоговых функций

С помощью *итоговых (агрегатных) функций* в рамках *SQL*-запроса можно получить ряд обобщающих статистических сведений о множестве отобранных значений выходного набора.

Пользователю доступны следующие основные *итоговые функции*:

- `Count` (Выражение) - определяет количество записей в выходном наборе SQL-запроса;
- `Min/Max` (Выражение) - определяют наименьшее и наибольшее из множества значений в некотором поле запроса;
- `Avg` (Выражение) - эта функция позволяет рассчитать среднее значение множества значений, хранящихся в определенном поле отобранных запросом записей. Оно является арифметическим средним значением, т.е. суммой значений, деленной на их количество.
- `Sum` (Выражение) - вычисляет сумму множества значений, содержащихся в определенном поле отобранных запросом записей.

Чаще всего в качестве выражения выступают имена столбцов. *Выражение* может вычисляться и по значениям нескольких таблиц.

Все эти функции оперируют со значениями в единственном столбце таблицы или с арифметическим выражением и возвращают единственное *значение*. Функции `COUNT` , `MIN` и `MAX` применимы как к числовым, так и к нечисловым полям, тогда как функции `SUM` и `AVG` могут использоваться только в случае числовых полей, за исключением `COUNT(*)` . При вычислении результатов любых функций сначала исключаются все пустые значения, после чего требуемая операция применяется только к оставшимся конкретным значениям столбца. Вариант `COUNT(*)` - особый случай использования функции `COUNT` , его назначение состоит в подсчете всех строк в результирующей таблице, независимо от того, содержатся там пустые, дублирующиеся или любые другие значения.

Если до применения обобщающей функции необходимо исключить дублирующиеся значения, следует перед именем столбца в определении функции поместить *ключевое слово* `DISTINCT` . Оно не имеет смысла для функций `MIN` и `MAX` , однако его использование может повлиять на результаты выполнения функций `SUM` и `AVG` , поэтому необходимо заранее обдумать, должно ли оно присутствовать в каждом конкретном случае. Кроме того, *ключевое слово* `DISTINCT` может быть указано в любом запросе не более одного раза.

Очень важно отметить, что *итоговые функции* могут использоваться только в списке предложения `SELECT` и в составе предложения `HAVING` . Во всех других случаях это недопустимо. Если *список* в предложении `SELECT` содержит *итоговые функции*, а в тексте запроса отсутствует фраза `GROUP BY` , обеспечивающая *объединение* данных в группы, то ни один из элементов списка предложения `SELECT` не может включать каких-либо ссылок на поля, за исключением ситуации, когда поля выступают в качестве аргументов *итоговых функций*.

Пример 6.4. Определить первое по алфавиту название товара.

```
SELECT Min(Товар.Название) AS Min_Название
FROM Товар

6.4. Определение первого по алфавиту названия товара.
```

Пример 6.5. Определить количество сделок.

```
SELECT Count(*) AS Количество_сделок
FROM Сделка

6.5. Определить количество сделок.
```

Пример 6.6. Определить суммарное количество проданного товара.

```
SELECT Sum(Сделка.Количество)
    AS Количество_товара
FROM Сделка

6.6. Определение суммарного количества проданного товара.
```

Пример 6.7. Определить среднюю цену проданного товара.

```
SELECT Avg(Товар.Цена) AS Avg_Цена
FROM Товар INNER JOIN Сделка
    ON Товар.КодТовара=Сделка.КодТовара;

6.7. Определение средней цены проданного товара.
```

Пример 6.8. Подсчитать общую *стоимость* проданных товаров.

```
SELECT Sum(Товар.Цена*Сделка.Количество)
    AS Стоимость
FROM Товар INNER JOIN Сделка
ON Товар.КодТовара=Сделка.КодТовара

6.8. Подсчет общей стоимости проданных товаров.
```

Предложение GROUP BY

Часто в запросах требуется формировать промежуточные итоги, что обычно отображается появлением в запросе фразы "для каждого...". Для этой цели в операторе `SELECT` используется предложение `GROUP BY` . *Запрос*, в котором присутствует `GROUP BY` , называется группирующим запросом, поскольку в нем группируются данные, полученные в результате выполнения *операции* `SELECT` , после чего для каждой отдельной группы создается единственная суммарная строка. Стандарт *SQL* требует, чтобы предложение `SELECT` и фраза `GROUP BY` были тесно связаны между собой. При наличии в операторе `SELECT` фразы `GROUP BY` каждый *элемент списка* в предложении `SELECT` должен иметь единственное *значение* для всей группы. Более того, предложение `SELECT` может включать только следующие типы элементов: имена полей, *итоговые функции*, *константы* и выражения, включающие комбинации перечисленных выше элементов.

Все имена полей, приведенные в списке предложения `SELECT` , должны присутствовать и во фразе `GROUP BY` - за исключением случаев, когда имя столбца используется в *итоговой функции*. Обратное правило не является справедливым - во фразе `GROUP BY` могут быть имена столбцов, отсутствующие в списке предложения `SELECT` .

Если совместно с `GROUP BY` используется предложение `WHERE` , то оно обрабатывается первым, а *группированию* подвергаются только те строки, которые удовлетворяют условию поиска.

Стандартом *SQL* определено, что при проведении *группирования* все отсутствующие значения рассматриваются как равные. Если две строки таблицы в одном и том же группируемом столбце содержат *значение* `NULL` и идентичные значения во всех остальных непустых группируемых столбцах, они помещаются в одну и ту же группу.

Пример 6.9. Вычислить средний объем покупок, совершенных каждым покупателем.

```
SELECT Клиент.Фамилия, Avg(Сделка.Количество)
    AS Среднее_количество
FROM Клиент INNER JOIN Сделка
    ON Клиент.КодКлиента=Сделка.КодКлиента
GROUP BY Клиент.Фамилия
```

```
6.9. Вычисление среднего объема покупок, совершенных каждым покупателем.
```

Фраза "каждым покупателем" нашла свое отражение в *SQL*-запросе в виде предложения `GROUP BY Клиент.Фамилия` .

Пример 6.10. Определить, на какую сумму был продан *товар* каждого наименования.

```
SELECT Товар.Название,
    Sum(Товар.Цена*Сделка.Количество)
    AS Стоимость
FROM Товар INNER JOIN Сделка
    ON Товар.КодТовара=Сделка.КодТовара
GROUP BY Товар.Название
```

```
6.10. Определение, на какую сумму был продан товар каждого наименования.
```

Пример 6.11. Подсчитать количество сделок, осуществленных каждой фирмой.

```
SELECT Клиент.Фирма, Count(Сделка.КодСделки)
    AS Количество_сделок
FROM Клиент INNER JOIN Сделка
    ON Клиент.КодКлиента=Сделка.КодКлиента
GROUP BY Клиент.Фирма
```

```
6.11. Подсчет количества сделок, осуществленных каждой фирмой.
```

Пример 6.12. Подсчитать общее количество купленного для каждой фирмы товара и его *стоимость*.

```
SELECT Клиент.Фирма, Sum(Сделка.Количество)
    AS Общее_Количество,
    Sum(Товар.Цена*Сделка.Количество)
    AS Стоимость
```

```
FROM Товар INNER JOIN
    (Клиент INNER JOIN Сделка
    ON Клиент.КодКлиента=Сделка.КодКлиента)
    ON Товар.КодТовара=Сделка.КодТовара
GROUP BY Клиент.Фирма
```

6.12. Подсчет общего количества купленного для каждой фирмы товара и его стоимости.

Пример 6.13. Определить суммарную *стоимость* каждого товара за каждый месяц.

```
SELECT Товар.Название, Month(Сделка.Дата)
    AS Месяц,
    Sum(Товар.Цена*Сделка.Количество)
    AS Стоимость
FROM Товар INNER JOIN Сделка
    ON Товар.КодТовара=Сделка.КодТовара
GROUP BY Товар.Название, Month(Сделка.Дата)
```

6.13. Определение суммарной стоимости каждого товара за каждый месяц.

Пример 6.14. Определить суммарную *стоимость* каждого товара первого сорта за каждый месяц.

```
SELECT Товар.Название, Month(Сделка.Дата)
    AS Месяц,
    Sum(Товар.Цена*Сделка.Количество)
    AS Стоимость
FROM Товар INNER JOIN Сделка
    ON Товар.КодТовара=Сделка.КодТовара
WHERE Товар.Сорт="Первый"
GROUP BY Товар.Название, Month(Сделка.Дата)
```

6.14. Определение суммарной стоимости каждого товара первого сорта за каждый месяц.

Предложение HAVING

При помощи `HAVING` отражаются все предварительно сгруппированные посредством `GROUP BY` блоки данных, удовлетворяющие заданным в `HAVING` условиям. Это дополнительная возможность "профильтровать" выходной набор.

Условия в `HAVING` отличаются от условий в `WHERE` :

- `HAVING` исключает из результирующего набора данных группы с результатами агрегированных значений;
- `WHERE` исключает из расчета агрегатных значений по группировке записи, не удовлетворяющие условию;
- в условии поиска `WHERE` нельзя задавать агрегатные функции.

Пример 6.15. Определить фирмы, у которых общее количество сделок превысило три.

```
SELECT Клиент.Фирма, Count(Сделка.Количество)
    AS Количество_сделок
FROM Клиент INNER JOIN Сделка
    ON Клиент.КодКлиента=Сделка.КодКлиента
GROUP BY Клиент.Фирма
HAVING Count(Сделка.Количество)>3
```

6.15. Определение фирм, у которых общее количество сделок превысило три.

Пример 6.16. Вывести *список* товаров, проданных на сумму более 10000 руб.

```
SELECT Товар.Название,
    Sum(Товар.Цена*Сделка.Количество)
```

```
    AS Стоимость
FROM Товар INNER JOIN Сделка
    ON Товар.КодТовара=Сделка.КодТовара
GROUP BY Товар.Название
HAVING Sum(Товар.Цена*Сделка.Количество)>10000
```

6.16. Вывод списка товаров, проданных на сумму более 10000 руб.

Пример 6.17. Вывести *список* товаров, проданных на сумму более 10000 без указания суммы.

```
SELECT Товар.Название
FROM Товар INNER JOIN Сделка
    ON Товар.КодТовара=Сделка.КодТовара
GROUP BY Товар.Название
HAVING Sum(Товар.Цена*Сделка.Количество)>10000
```

6.17. Вывод списка товаров, проданных на сумму более 10000 без указания суммы.