Итоговые функции

Функции в SQL представляются ключевыми словами и используются для математических преобразований данных в столбце с целью соответствующего представления данных при выводе.

Функция — это команда, всегда используемая в связи с именем столбца или выражением. В SQL имеется несколько типов функций. В ходе этого урока мы рассмотрим итоговые функции. *Итоговая функция* — это функция, используемая в операторе SQL для получения итоговой информации типа общего числа строк, сумм или среднего значения.

- COUNT
- SUM
- MAX
- MIN
- AVG

Функция COUNT используется для подсчета строк или значений в столбце, отличных от значения NULL При использовании в запросах функция COUNT возвращает числовое значение. При использовании с опцией DISTINCT функция COUNT посчитает только разные строки (т. е строки без учета повторений). По умолчанию используется опция ALL (противоположность DISTINCT), поэтому указывать ALL не обязательно Повторяющиеся значения считаются, когда DISTINCT не указано Другой опцией функции COUNT является звездочка (*) При использовании со звездочкой функция COUNT возвращает число всех строк в таблице, без исключения повторяющихся, не обращая внимания на возможно имеющиеся в столбце значения NULL Синтаксис оператора функции COUNT следующий

COUNT [(*) | (DISTINCT | ALL) J (имя_столбца)

Команда DISTINCT не используется с COUNT(*), а используется только с COUNT(имя_столбца)

Пример	*Значение*
SELECT COUNT (EMP_ID) FROM EMPLOYEE__PAY_TBL;	Подсчет числа табельных номеров всех служащих
SELECT COUNT (DISTINCT SALARY) FROM EMPLOYEE_PAY_TBL;	Подсчет только разных строк
SELECT COUNT (ALL SALARY); FROM EMPLOYEE_PAY_TBL;	Подсчет всех строк для SALARY
SELECT COUNT (*) FROM EMPLOYEE_TBL;	Подсчет всех строк таблицы EMPLOYEE_TBL

Ввиду того, что функция COUNT подсчитывает строки, тип содержащихся в столбце данных роли не играет, т. е. данные в строке могут быть любого типа

Функция SUM используется для подсчета суммы значений в столбце для заданной группы строк. Функцию SUM можно использовать вместе с ключевым словом DISTINCT. При использовании ключевого слова DISTINCT повторно встречающиеся значения в сумму не включаются. В этом случае итог не будет полной суммой, поскольку некоторые строки могут быть при суммировании пропущены.

Синтаксис оператора функции SUM следующий

SUM ([DISTINCT] имя_столбца)

При использовании функции SUM тип значения в столбце предполагается числовым. Функцию нельзя использовать по отношению к столбцам с символьными значениями или значениями дат и времени

Пример _____ *Значение* _____

SELECT SUM (SALARY) Подсчет суммы зарплат всех служащих
FROM EMPLOYEE_PAY_TBL;

SELECT SUM (DISTINCT SALARY) Подсчет суммы зарплат всех
FROM EMPLOYEE_PAY_TBL; служащих без учета
повторяющихся - значений

Подсчитаем сумму всех значений стоимости товаров из таблицы PRODUCT S_TBL.

Ввод:

SELECT SUM(COST)
FROM PRODUCTS_TBL;

Функция AVG используется для подсчета среднего для значений заданной группы строк. При использовании с ключевым словом DISTINCT повторно встречающиеся значения в среднем не учитываются.

Синтаксис оператора функции AVG следующий.

AVG([DISTINCT] имя_столбца)

Для использования функции AVG тип значения в столбце должен быть числовым.

Пример _____ *Значение* _____

SELECT AVG (SALARY) Подсчет средней зарплаты всех служащих
FROM EMPLOYEE_PAY_TBL;

SELECT AVG (DISTINCT SALARY) Подсчет среднего значения для
FROM EMPLOYEE_PAY_TBL ; зарплат всех служащих без
_ учета повторяющихся значений

Подсчитаем среднее для всех значений стоимости товаров из таблицы PRODUCTS_TBL.

Ввод:

```
SELECT AVG(COST)
FROM PRODUCTS_TBL;
```

В некоторых реализациях SQL результат будет округлен до точности, заданной типом значений в столбце.

В следующем примере в одном запросе используются две функции. Поскольку одним служащим платят ставку, а другим — почасово, можно подсчитать средние значения и для столбца PAY_RATE, и для столбца SALARY.

Ввод:

```
SELECT AVG(PAY_RATE), AVG(SALARY)
FROM PRODUCTS_TBL;
```

Функция MAX

Функция MAX используется для подсчета максимума для значений заданной группы строк. Значения NULL при этом игнорируются. Можно использовать также ключевое слово DISTINCT, но поскольку повторно встречающиеся значения на значение максимума не влияют, это ключевое слово оказывается в данном случае бесполезным.

MAX([DISTINCT] имя_столбца)

Пример _____ *Значение* _____

SELECT MAX (SALARY) Нахождение максимальной зарплаты
FROM EMPLOYEE_PAY_TBL;

SELECT MAX (DISTINCT SALARY) Нахождение максимальной
FROM EMPLOYEE_PAY_TBL; зарплаты без учета
повторяющихся значений

_

Подсчитаем максимум всех значений стоимости товаров из таблицы PRODUCT S_TBL.

Ввод:

SELECT MAX(COST)
FROM PRODUCTS_TBL;

Функция MIN используется для подсчета минимума для значений заданной группы строк. Значения NULL при этом игнорируются. Можно использовать также ключевое слово DISTINCT, но поскольку повторно встречающиеся значения на значение минимума не влияют, это ключевое слово оказывается в данном случае бесполезным.

MIN([DISTINCT] имя_столбца)

Пример _____ *Значение* _____

SELECT MIN (SALARY) Нахождение минимальной зарплаты
FROM EMPLOYEE_PAY_TBL;

SELECT MIN (DISTINCT SALARY) Нахождение минимальной
FROM EMPLOYEE_PAY_TBL; зарплаты без учета повторяющих-
_ ся значений

Подсчитаем минимум всех значений стоимости товаров из таблицы PRODUCTS_TBL.

Ввод:

SELECT MIN(COST)
FROM PRODUCTS_TBL;

Комбинирование итоговых функций с арифметическими операциями.

Ввод:

```
SELECT COUNT(ORD_NUM), SUM(QTY),
SUM(QTY) / COUNT(ORD_NUM) AVG_QTY
FROM ORDERS_TBL;
```

Здесь подсчитано число заказов, указана общая сумма стоимости всех заказов и с помощью деления второй величины на первую вычислена средняя стоимость заказа. Для представления последней создан псевдоним столбца — AVG_QTY.

Резюме

Итоговые функции несложно использовать и они могут оказаться весьма полезными. Помните о том, что при использовании итоговых функций значение NULL не учитывается — исключением является функция COUNT в формате COUNT (*).

Итоговые функции являются первыми из функций SQL, но они не единственные и существует множество других. Итоговые функции используются также для группирования значений. По мере изучения других функций, вы обнаружите, что в основном они имеют похожий синтаксис и что лежащие в их основе концепции достаточно просты.

Группирование данных — это размещение данных в столбцах с повторяющимися значениями в определенном логичном порядке. Например, в базе данных содержится информация о служащих. Служащие могут жить в разных городах, но многие из них живут в одном городе. Вполне вероятно, что вам может понадобиться информация по каждому конкретному городу и живущих там служащих. Для этого вы группируете информацию о служащих по городам — и соответствующий отчет готов!

Предположим, что вам необходимо найти среднюю зарплату служащих по каждому из городов. Это можно сделать, применив к столбцу SALARY итоговую функцию AVG, а затем использовав GROUP BY для группирования выводимых данных по городам.

Группирование данных осуществляется с помощью выражения GROUP BY в операторе SELECT. На предыдущем уроке были рассмотрены итоговые функции. В ходе данного урока мы с вами научимся использовать итоговые функции в совокупности с выражением GROUP BY, чтобы лучше организовать представляемые данные.

Ключевое слово GROUP BY используется в операторе SELECT для того, чтобы объединять повторяющиеся значения в группы. Ключевое слово GROUP BY должно следовать за выражением WHERE и предшествовать ключевому слову ORDER BY.

Вот какая должна быть последовательность ключевых слов в операторе, выполняющем запрос:

SELECT FROM WHERE GROUP BY ORDER BY

Ключевое слово GROUP BY должно следовать за условиями в выражении ключевого слова WHERE и предшествовать ключевому слову ORDER BY, если последнее имеется.

SELECT столбец1, столбец2

FROM таблица1, таблица2 WHERE условия

GROUP BY столбец1, столбец2

ORDER BY столбец1, столбец2

Группирование выбранных данных

Группировать данные просто. В выражении ключевого слова GROUP BY могут использоваться только выбранные столбцы (т. е. столбцы из списка ключевого слова SELECT в операторе запроса). Если имя столбца не указано в списке ключевого слова SELECT, то имя этого столбца в выражении ключевого слова GROUP BY использовать нельзя. Это логично — как группировать в отчете данные, которых в нем нет?

Но если столбец выбран, то его имя должно быть включено в выражение ключевого слова GROUP BY. Имя столбца можно представить и его номером, о чем мы поговорим немного позже.

При группировании данных порядок группирования столбцов не обязан совпадать с порядком, заданным в выражении ключевого слова SELECT.

К функциям группирования — функциям, используемым в выражении ключевого слова GROUP BY для объединения данных в группы, — относятся AVG, MAX, MIN, SUM и COUNT. Это итоговые функции, но ранее итоговые функции использовались по отношению ко всем данным столбца, а здесь мы рассмотрим использование итоговых функций для группирования повторяющихся значений

Создание групп и использование итоговых функций

При использовании в операторе SELECT ключевого слова GROUP BY должны соблюдаться некоторые правила.

В частности, имена выбранных для отображения столбцов должны присутствовать и в выражении ключевого слова GROUP BY, за исключением тех, к которым применены итоговые функции. Столбцы в выражении ключевого слова GROUP BY не обязательно должны быть представлены в том же порядке, что и в выражении ключевого слова SELECT. Но если имя столбца указано в выражении ключевого слова SELECT, имя этого столбца должно присутствовать и в выражении ключевого слова GROUP BY. Вот несколько примеров использования оператора SELECT с ключевым словом GROUP BY.

```
SELECT EMP_ID, CITY
FROM EMPLOYEE_TBL
GROUP BY CITY, EMP_ID;
```

В этом операторе SQL из таблицы EMPLOYEE_TBL выбираются столбцы EMP_ID и CITY, а данные последних выводятся сгруппированными сначала по CITY, а затем по EMP_ID.

Обратите внимание на порядок выбора столбцов и на порядок столбцов в выражении ключевого слова GROUP BY

```
SELECT EMP_ID, SUM(SALARY)
FROM EMPLOYEE_PAY_TBL
GROUP BY SALARY, EMP_ID;
```

Этот оператор SQL возвращает данные столбца EMP_ID и сумму по группам зарплат, созданным по величине зарплаты (SALARY) и табельному номеру (EMP_ID).

В следующем примере для группирования данных комбинируется использование нескольких компонентов запроса. Необходимо получить средние значения для почасовой оплаты и зарплаты, но только для городов Псков и Уайтленд. Для этого данные группируются по полю CITY — другого выбора здесь нет, поскольку иначе из выбранных столбцов используется итоговая функция. Наконец, отчет упорядочивается сначала по столбцу 2, а затем по столбцу 3, т. е. по средней почасовой оплате и средней зарплате.

Ввод:

```
SELECT CITY, AVG(PAY_RATE), AVG(SALARY)
FROM EMP_PAY_TMP
WHERE CITY IN (‘PSKOV','WHITELAND')
GROUP BY CITY
ORDER BY 2,3;
```

Рассмотрим использование в выражении ключевого слова ORDER BY итоговых функций MAX и MIN.

Ввод:

```
SELECT CITY, MAX(PAY_RATE), MIN(SALARY)
FROM EMP_PAY_TMP
GROUP BY CITY;
```

Вывод:

CITY	MAX(PAY_RATE)	MIN(SALARY)
GREENWOOD		30000
PSKOV	15	20000
WHITELAND		40000

Представление имен столбцов числами

В отличие от выражения ключевого слова ORDER BY, в выражении ключевого слова GROUP BY указать порядок столбцов с помощью их номеров нельзя — за исключением того случая, когда используется ключевое слово UNION и имена всех столбцов разные. Вот пример использования номеров вместо имен столбцов.

```
SELECT EMP_ID, SUM(SALARY)
FROM EMPLOYEE_PAY_TBL
UNION
SELECT EMP_ID, SUM(PAY_RATE)
FROM EMPLOYEE_PAY_TBL
GROUP BY 2, 1;
```

Этот оператор SQL возвращает табельный номер служащего (EMP_ID) и группирует суммы по значениям зарплаты. При использовании ключевого слова UNION результаты двух операторов SELECT объединяются. Группирование выполняется сначала по столбцу 2, представляющем зарплату (SALARY), а затем по столбцу 1, представляющем табельный номер служащего (EMP_ID).

GROUP BY И ORDER BY

Оба эти ключевые слова задают сортировку данных. В выражении ключевого слова ORDER BY задается сортировка данных запроса, а в выражении ключевого слова GROUP BY — сортировка этих данных по группам. Поэтому ключевое слово GROUP BY можно использовать для сортировки точно так же, как и ORDER BY.

Вот несколько особенностей использования ключевого слова GROUP BY для сортировки.

Все выбранные столбцы, к которым не применяются итоговые функции, должны быть указаны в списке ключевого слова GROUP BY.

В отличие от выражения ключевого слова ORDER BY, в выражении ключевого слова GROUP BY имена столбцов нельзя заменить числами.

Использовать ключевое слово GROUP BY вообще нет необходимости, если не используются итоговые функции.

Вот пример использования для сортировки данных ключевого слова GROUP BY вместо ключевого слова ORDER BY:

Ввод:

```
SELECT LAST_NAME, FIRST_NAME, CITY
FROM EMPLOYEE_TBL
GROUP BY LAST_NAME;
```

В этом примере сервер базы данных сообщает об ошибке из-за того, что имя столбца FIRST_NAME не указано в выражении ключевого слова GROUP BY. Все столбцы из списка ключевого слова SELECT должны быть указаны в выражении ключевого слова GROUP BY, за исключением тех столбцов, к которым применяются итоговые функции.

В следующем примере проблема предыдущего оператора решена путем добавления в список ключевого слова GROUP BY недостающих имен из списка ключевого слова SELECT.

Ввод:

```
SELECT IAST_NAME, FIRST_NAME, CITY
FROM EMPLOYEE_TBL
GROUP BY LAST_NAME, FIRST_NAME, CITY;
```

Хотя ключевые слова GROUP BY и ORDER BY и функционируют подобным образом, между ними имеется существенное отличие. Ключевое слово GROUP BY предназначено для группирования одинаковых значений, а задачей ORDER BY является представление данных просто в определенном порядке. Ключевые слова GROUP BY и ORDER BY можно использовать в одном операторе SELECT, но каждое из них должно выполнять свою задачу. В одном операторе SELECT ключевое слово GROUP BY должно предшествовать ключевому слову ORDER BY.

Ключевое слово GROUP BY можно использовать для сортировки данных в операторе CREATE VIEW, а вот ключевое слово ORDER BY использовать в операторе CREATE VIEW нельзя

Ключевое слово HAVING используется в операторе SELECT вместе с ключевым словом GROUP BY, чтобы указать какие из групп должны быть представлены в выводе. Для GROUP BY ключевое слово HAVING играет ту же роль, что и WHERE для ORDER BY. Другими словами, WHERE задает условия для значений из выбранных столбцов, а HAVING задает условия для групп, создаваемых с помощью GROUP BY.

Ключевое слово HAVING в операторе SELECT должно следовать за выражением ключевого слова GROUP BY и тоже предшествовать ключевому слову ORDER BY, если последнее используется.

Синтаксис оператора SELECT, в котором используется ключевое слово HAVING, следующий.

```
SELECT столбец1, столбец2
FROM таблица1, таблица2
WHERE условия
GROUP BY столбец1, столбец2
HAVING условия
ORDER BY столбец1, столбец2
```

Ключевое слово GROUP BY используется, главным образом, с итоговыми функциями SQL типа SUM, AVG, MAX, MIN и COUNT. Ключевое слово GROUP BY подобно ключевому слову ORDER BY в том смысле, что оба они сортируют выводимые данные. Ключевое слово GROUP BY предназначено для сортировки данных по группам, но может использоваться и для обычной сортировки данных, хотя последнее проще сделать с помощью ключевого слова ORDER BY.

Ключевое слово HAVING используется в операторе SELECT вместе с ключевым словом GROUP BY, чтобы задать условия отбора для создаваемых групп. Ключевое слово WHERE используется для задания условий отбора для данных столбцов из списка ключевого слова SELECT.