ВЫСШИЙ КОЛЛЕДЖ ИМЕНИ КУМАША НУРГАЛИЕВА

Учебно-методическое пособие по учебной практике по базам данных

«SQL в задачах и примерах»

составитель Литвинова О.Ю.

Усть-Каменогорск, 2021

УДК
ББК

Литвинова О.Ю.
Учебно-методическое пособие по учебной практике по базам данных «SQL в задачах и примерах»/Литвинова О.Ю. – Усть-Каменогорск; Учреждение «Высший колледж имени Кумаша Нургалиева», 2021 – 29 стр.

ISBN

Учебно-методическое пособие содержит основные правила построения синтаксиса языка SQL, а также задания для закрепления и отработки навыков владения языком структурированных запросов. Может быть использовано в качестве вспомогательного пособия для использования студентами специальности 1304000 «Вычислительная техника и программное обеспечение» при прохождении производственного обучения по базам данных.

УДК
ББК

ISBN

СОДЕРЖАНИЕ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Настоящее учебно-методическое пособие разработано в качестве вспомогательного пособия для использования студентами специальности 1304000 «Вычислительная техника и программное обеспечение» при прохождении производственного обучения по базам данных. Общий объем – 36 часов.

Цель учебно-методического пособия: оказание помощи обучающимся в изучении SQL-команд и применения их на практике при выполнении самостоятельной работы по созданию запросов в проектах по дисциплинам «Базы данных», «Объектно-ориентированное программирование», «Программирование в офисных приложениях».

Учебно-методическое пособие содержит основные правила построения синтаксиса языка SQL, а также задания для закрепления и отработки навыков владения языком структурированных запросов, которые позволят обучающимся самостоятельно овладеть фундаментальными знаниями, профессиональными умениями и навыками деятельности по специальности, опытом творческой и исследовательской деятельности и направлены на формирование следующих компетенций:

-создавать, сохранять, видоизменять, удалять запрос;

-формировать условия запроса;

-строить запросы на несколько таблиц, и использовать внешние связи;

-использовать итоговые запросы для суммирования и других итоговых операций.;

-создавать модифицирующие запросы всех типов.

Цель работы: изучить SQL-команды и применять их на практике.

Язык SQL (Structured Query Language) предназначен для выполнения операций над таблицами (создание, удаление, изменение структуры) и над данными таблиц (выборка, изменение, добавление и удаление), а также некоторых сопутствующих операций, и не содержит операторов управления, организации подпрограмм, ввода-вывода и т. п. Таким образом, SQL не обладает функциями полноценного языка разработки, т.е. является непроцедурным языком. В связи с этим SQL автономно не используется, обычно его включают в состав средств разработки программ СУБД (например, FoxPro СУБД Visual FoxPro, ObjectPAL СУБД Paradox, Visual Basic for Applications СУБД Access), причем в различных СУБД состав операторов SQL может несколько отличаться .

Язык SQL является первым и пока единственным стандартным языком

работы с базами данных, который получил достаточно широкое распространение. Практически все крупнейшие разработчики СУБД в настоящее время создают свои продукты с использованием языка SQL либо интерфейса SQL, и большинство таких компаний участвуют в работе, по меньшей мере, одной организации, которая занимается разработкой стандартов этого языка.

Язык SQL также принят в качестве Федерального стандарта обработки информации (Federal Information Processing Standard – FIPS), который должен соблюдаться в СУБД для получения разрешения продавать ее на территории США. Хотя исходные концепции языка SQL были разработаны корпорацией IBM, его важность очень скоро подтолкнула и других разработчиков к созданию собственных реализаций. В настоящее время на рынке доступны буквально сотни продуктов, построенных на использовании языка SQL,причем постоянно приходится слышать о выпуске все новых и новых версий

В современных СУБД с интерактивным интерфейсом можно создавать запросы, используя другие средства, например QBE (Query By Example – запрос по образцу)

Однако по сравнению с QBE применение SQL зачастую позволяет повысить эффективность обработки данных в базе.

Основным назначением языка SQL (как и других языков для работы с базами данных) является подготовка и выполнение запросов. В результате выборки данных из одной или нескольких таблиц может быть получено множество записей, называемое представлением.

Представление по существу является таблицей, формируемой в результате выполнения запроса. Можно сказать, что оно является разновидностью хранимого запроса. По одним и тем же таблицам можно построить несколько представлений. Само представление описывается путем указания идентификатора представления и запроса, который должен быть выполнен для его получения.

SQL-Урок 1. Язык SQL. Основные понятия

Для того, чтобы начать изучать **SQL** нам нужно сначала понять, что такое база данных.

1. Что такое База Данных

База данных (БД) - упорядоченный набор логически взаимосвязанных данных, используемых совместно, и которые хранятся в одном месте. Если коротко, то простейшая **БД** это обычная таблица со строками и столбцами в которой хранится разного рода информация (примером может служить таблица в **Excel**). Так, часто, с **БД** нераздельно связывают **Системы управления базами данных (СУБД)**, которые предоставляют функционал для работы с **БД**. Язык **SQL** как раз и является частью **СУБД**, которая осуществляет управление информацией в **БД**. Мы будем считать **БД** набором обычных таблиц, которые хранятся в отдельных файлах.

2. Что такое SQL

Итак, переходим к **SQL**.

SQL - простой язык программирования, который имеет немного команд и которой может научиться любой желающий. Расшифровывается как **Structured Query Language** - язык структурированных запросов, который был разработан для работы с **БД**, а именно, чтобы получать /добавлять /изменять данные, иметь возможность обрабатывать большие массивы информации и быстро получать структурированную и сгруппированную информацию. Есть много вариантов языка **SQL**, но у них всех основные команды почти одинаковы. Также существует и много **СУБД**, но основными из них являются: **Microsoft Access, Microsoft SQL Server, MySQL, Oracle SQL, IBM DB2 SQL, PostgreSQL та Sybase Adaptive Server SQL**. Чтобы работать с **SQL** кодом, нам понадобится одна из вышеперечисленных **СУБД**. Для обучения мы будем использовать **СУБД Microsoft Access** .

SQL как и другие языки программирования имеет свои команды (операторы), с помощью которых отдаются инструкции для выборки данных. Чтобы рассмотреть как работают операторы **SQL**, мы будем использовать мнимую **БД** с информацией о реализованной продукции:

ID	Month	Product	City	Quantity	Amount
2	April	Bikes	Montreal	12	$4 500,00
3	April	Bikes	Montreal	56	$21 000,00
4	April	Bikes	San Francisco	854	$320 250,00
5	April	Bikes	New York	25	$9 375,00
6	April	Skates	Montreal	56	$5 544,00
7	April	Skates	Toronto	854	$84 546,00
8	April	Skates	San Francisco	25	$2 475,00
9	April	Skates	New York	663	$65 637,00
10	April	Skis Long	Montreal	854	$209 230,00
11	April	Skis Long	Toronto	25	$6 125,00
12	April	Skis Long	San Francisco	663	$162 435,00
13	April	Skis Long	New York	21	$5 145,00
14	April	Skis Short	Montreal	21	$4 389,00
15	April	Skis Short	Toronto	4	$836,00
16	April	Skis Short	San Francisco	522	$109 098,00

Рисунок 1 –Образец базы данных

Самым первым и главным оператором в **SQL** является ***SELECT***. С его помощью мы можем отбирать необходимые нам поля данных в таблице.

1. Выборка отдельных полей.

SELECT Product FROM Sumproduct

Product
Bikes
Bikes
Bikes
Bikes
Skates
Skates
Skates
Skates
Skis Long
Skis Long

Рисунок 2 –Результат запроса

Видим, что наш **SQL запрос** отобрал колонку **Product** из таблицы **Sumproduct** .

2. Выборка нескольких полей.

Допустим, нам необходимо выбрать название и количество реализованного товара. Для этого просто перечисляем необходимые поля через запятую:

SELECT Product, Quantity FROM Sumproduct

Product	Quantity
Bikes	12
Bikes	56
Bikes	854
Bikes	25
Skates	56
Skates	854
Skates	25
Skates	663
Skis Long	854
Skis Long	25
Skis Long	663

Рисунок 3 –Запрос в выборкой нескольких полей

3. Выборка всех столбцов.

Если же нам необходимо получить всю таблицу со всеми полями, тогда просто ставим знак звездочка (*):

SELECT * FROM Sumproduct

ID	Month	Product	City	Quantity	Amount
2	April	Bikes	Montreal	12	$4 500,00
3	April	Bikes	Montreal	56	$21 000,00
4	April	Bikes	San Francisco	854	$320 250,00
5	April	Bikes	New York	25	$9 375,00
6	April	Skates	Montreal	56	$5 544,00
7	April	Skates	Toronto	854	$84 546,00
8	April	Skates	San Francisco	25	$2 475,00
9	April	Skates	New York	663	$65 637,00
10	April	Skis Long	Montreal	854	$209 230,00
11	April	Skis Long	Toronto	25	$6 125,00
12	April	Skis Long	San Francisco	663	$162 435,00
13	April	Skis Long	New York	21	$5 145,00
14	April	Skis Short	Montreal	21	$4 389,00
15	April	Skis Short	Toronto	4	$836,00
16	April	Skis Short	San Francisco	522	$109 098,00

Рисунок 4–Запрос в выборкой всех полей

PS. Все операторы в **SQL** нечувствительны к регистру, поэтому вы можете писать как большими буквами, так и маленькими (как правило, их принято писать большими буквами, чтобы различать от названий полей и таблиц). Названия же таблиц и полей является наоборот чувствительными к регистру и должны писаться точно как в **БД**.

В будущем нам может понадобиться сортировать нашу выборку - в алфавитном порядке для текста или по возрастанию/убыванию - для цифровых значений. Для таких целей в **SQL** есть специальный оператор ***ORDER BY*** .

1. Сортировка выбранных данных.

Давайте всю нашу таблицу отсортируем по сумме реализации продукции, а именно по столбцу **Amount**.

SELECT * FROM Sumproduct ORDER BY Amount

ID	Month	Product	City	Quantity	Amount
47	January	Skates	New York	4	$396,00
15	April	Skis Short	Toronto	4	$836,00
35	February	Skis Short	Toronto	4	$836,00
74	March	Skis Short	Montreal	4	$836,00
52	January	Skis Long	San Francisco	4	$980,00
41	February	Snow Board	New York	4	$1 232,00
18	April	Snow Board	Montreal	4	$1 232,00
79	March	Snow Board	Toronto	4	$1 232,00
62	March	Bikes	Montreal	4	$1 500,00
28	February	Skates	San Francisco	21	$2 079,00
26	February	Skates	Montreal	21	$2 079,00
46	January	Skates	Montreal	21	$2 079,00
8	April	Skates	San Francisco	25	$2 475,00

Рисунок 5–Результат сортировки

Видим, что запрос отсортировал записи по возрастанию в поле **Amount**. Обязательно нужно соблюдать последовательность расположения операторов, т.е. оператор ***ORDER BY*** должен идти в самом конце запроса. В противном случае будет получено сообщение об ошибке. Также особенностью оператора ***ORDER BY*** является то, что он может сортировать данные по полю, которого мы не выбирали в запросе, то есть достаточно, чтобы оно вообще было в **БД**.

2. Сортировка по нескольким полям.

Теперь отсортируем наш пример дополнительно за еще одним полем. Пусть это будет поле **City**, которое отображает место реализации продукции.

SELECT * FROM Sumproduct ORDER BY Amount, City

ID	Month	Product	City	Quantity	Amount
47	January	Skates	New York	4	$396,00
74	March	Skis Short	Montreal	4	$836,00
35	February	Skis Short	Toronto	4	$836,00
15	April	Skis Short	Toronto	4	$836,00
52	January	Skis Long	San Francisco	4	$980,00
18	April	Snow Board	Montreal	4	$1 232,00
41	February	Snow Board	New York	4	$1 232,00
79	March	Snow Board	Toronto	4	$1 232,00
62	March	Bikes	Montreal	4	$1 500,00
26	February	Skates	Montreal	21	$2 079,00
46	January	Skates	Montreal	21	$2 079,00
28	February	Skates	San Francisco	21	$2 079,00

Рисунок 6–Сортировка по нескольким полям

Очередность сортировки будет зависеть от порядка расположения полей в запросе. То есть, в нашем случае сначала данные будут рассортированы по колонке **Amount** , а затем по **City**.

3. Направление сортировки

Несмотря на то, что по умолчанию оператор ***ORDER BY*** сортирует по возрастанию, мы можем также прописать сортировки значений по убыванию. Для этого в конце каждого поля проставляем оператор ***DESC*** (что является сокращением от слова ***DESCENDING).***

SELECT * FROM Sumproduct ORDER BY Amount DESC, City

ID	Month	Product	City	Quantity	Amount
4	April	Bikes	San Francisco	854	$320 250,00
22	February	Bikes	Montreal	663	$248 625,00
25	February	Bikes	New York	658	$246 750,00
70	March	Skis Long	Montreal	854	$209 230,00
10	April	Skis Long	Montreal	854	$209 230,00
21	April	Snow Board	New York	663	$204 204,00
59	January	Snow Board	Toronto	663	$204 204,00
63	March	Bikes	Montreal	522	$195 750,00
12	April	Skis Long	San Francisco	663	$162 435,00
32	February	Skis Long	San Francisco	663	$162 435,00
71	March	Skis Long	Toronto	663	$162 435,00
58	January	Snow Board	Montreal	522	$160 776,00
80	March	Snow Board	San Francisco	522	$160 776,00

Рисунок 7–Запрос ORDER BY

В данном примере, значение в поле **Amount** были отсортированы по убыванию, а в поле **City** - по возрастанию. Оператор ***DESC*** применяется только для одного столбца, поэтому при необходимости его нужно прописывать после каждого поля, которое принимает участие в сортировке.

В большинстве случаев необходимо получать не все записи, а только те, которые соответствуют определенным критериям. Поэтому для осуществления фильтрации выборки в **SQL** есть специальный оператор **WHERE**.

1. Простое фильтрование оператором WHERE.

Давайте из нашей таблицы, например, отберем записи, относящиеся только к определенному товару. Для этого мы укажем дополнительный параметр отбора, который будет фильтровать значение по колонке **Product**.

Пример запроса для отбора текстовых значений:
SELECT * FROM Sumproduct WHERE Product = *'Bikes'*

ID	Month	Product	City	Quantity	Amount
2	April	Bikes	Montreal	12	$4 500,00
3	April	Bikes	Montreal	56	$21 000,00
4	April	Bikes	San Francisco	854	$320 250,00
5	April	Bikes	New York	25	$9 375,00
22	February	Bikes	Montreal	663	$248 625,00
23	February	Bikes	San Francisco	21	$7 875,00
24	February	Bikes	San Francisco	54	$20 250,00
25	February	Bikes	New York	658	$246 750,00
42	January	Bikes	Montreal	75	$28 125,00
43	January	Bikes	Toronto	12	$4 500,00
44	January	Bikes	San Francisco	136	$51 000,00
45	January	Bikes	New York	21	$7 875,00
62	March	Bikes	Montreal	4	$1 500,00
63	March	Bikes	Montreal	522	$195 750,00
64	March	Bikes	San Francisco	125	$46 875,00
65	March	Bikes	New York	212	$79 500,00

Рисунок 8– Использование фильтров

Как видим, условие отбора взято в одинарные кавычки, что является обязательным при фильтровании текстовых значений. При фильтровании числовых значений кавычки не нужны.

Пример запроса для отбора числовых значений:

SELECT * FROM Sumproduct WHERE Amount > *40000* **ORDER BY Amount**

ID	Month	Product	City	Quantity	Amount
39	February	Snow Board	Toronto	136	$41 888,00
61	January	Snow Board	New York	136	$41 888,00
64	March	Bikes	San Francisco	125	$46 875,00
44	January	Bikes	San Francisco	136	$51 000,00
48	January	Skates	San Francisco	522	$51 678,00
9	April	Skates	New York	663	$65 637,00
27	February	Skates	Toronto	663	$65 637,00
65	March	Bikes	New York	212	$79 500,00
7	April	Skates	Toronto	854	$84 546,00
67	March	Skates	Toronto	854	$84 546,00
36	February	Skis Short	San Francisco	522	$109 098,00
16	April	Skis Short	San Francisco	522	$109 098,00

Рисунок 9–Запрос ORDER BY

В этом примере мы отобрали записи, в которых выручка от реализации составила более **40 тыс. $** и, дополнительно, все записи отсортировали по возрастанию по полю **Amount.**

В таблице ниже указан перечень условных операторов, поддерживаемых **SQL**:

Таблица 1 Условные операторы

Знак операции	Значение
=	Равно
<>	Не равно
<	Меньше
<=	Меньше или равно
>	Больше
>=	Больше или равно
BETWEEN	Между двумя значениями
IS NULL	Отсутствует запись

2. Фильтрация по диапазону значений (BETWEEN).

Для отбора данных, которые лежат в определенном диапазоне, используется оператор **BETWEEN**. В следующем запросе будут отобраны все значения, лежащие в пределах от ***1000 $*** в ***2000 $*** включительно, в поле **Amount**.

SELECT * FROM Sumproduct WHERE Amount BETWEEN *1000* **AND** *2000*

ID	Month	Product	City	Quantity	Amount
18	April	Snow Board	Montreal	4	$1 232,00
41	February	Snow Board	New York	4	$1 232,00
62	March	Bikes	Montreal	4	$1 500,00
79	March	Snow Board	Toronto	4	$1 232,00

Рисунок 10–Запрос с использованием оператора BETWEEN

Очередность сортировки будет зависеть от порядка расположения полей в запросе. То есть, в нашем случае сначала данные будут отсортированы по колонке **Amount**, а затем по **City**.

3. Выборка пустых записей (IS NULL).

В **SQL** существует специальный оператор для выборки пустых записей (называеьбся **NULL**). Пустой записью считается любая ячейка в таблице, в которую не введены какие-либо символы. Если в ячейку введен **0** или **пробел**, то считается, что поле заполнено.

SELECT * FROM Sumproduct WHERE Amount IS NULL

ID	Month	Product	City	Quantity	Amount
5	April	Bikes	New York	25	
19	April	Snow Board	Toronto	522	

Рисунок 11–Запрос с использованием оператора NULL

В примере выше, мы нарочно удалили два значения в поле **Amount**, чтобы продемонстрировать работу оператора **NULL**.

4. Расширенные фильтрации (AND, OR).

Язык **SQL** не ограничивается фильтрацией по одному условию, для собственных целей вы можете использовать достаточно сложные конструкции для выборки данных одновременно по многим критериям. Для этого в **SQL** есть дополнительные операторы, которые расширяют возможности оператора **WHERE**. Такими операторами являются: **AND, OR, IN, NOT**. Приведем несколько примеров работы данных операторов.

SELECT * FROM Sumproduct WHERE Amount > *40000* **AND City** = *'Toronto'*

ID	Month	Product	City	Quantity	Amount
7	April	Skates	Toronto	854	$84 546,00
19	April	Snow Board	Toronto	522	$160 776,00
27	February	Skates	Toronto	663	$65 637,00
39	February	Snow Board	Toronto	136	$41 888,00
59	January	Snow Board	Toronto	663	$204 204,00
67	March	Skates	Toronto	854	$84 546,00
71	March	Skis Long	Toronto	663	$162 435,00
75	March	Skis Short	Toronto	522	$109 098,00

Рисунок 12–Запрос с использованием фильтров

SELECT * FROM Sumproduct WHERE Month= *'April'* **OR Month**= *'March'*

ID	Month	Product	City	Quantity	Amount
15	April	Skis Short	Toronto	4	$836,00
74	March	Skis Short	Montreal	4	$836,00
18	April	Snow Board	Montreal	4	$1 232,00
79	March	Snow Board	Toronto	4	$1 232,00
62	March	Bikes	Montreal	4	$1 500,00
8	April	Skates	San Francisco	25	$2 475,00
77	March	Skis Short	San Francisco	21	$4 389,00
14	April	Skis Short	Montreal	21	$4 389,00
2	April	Bikes	Montreal	12	$4 500,00
13	April	Skis Long	New York	21	$5 145,00
72	March	Skis Long	San Francisco	21	$5 145,00

Рисунок 13–Запрос с использованием фильтров

Давайте объединим операторы **AND** и **OR**. Для этого сделаем выборку велосипедов (***Bikes***) и коньков (***Skates***), которые были проданы в марте ***(March***).

SELECT * FROM Sumproduct WHERE Product = *'Bikes'* **OR Product** = *'Skates'* **AND Month**= *'March'*

ID	Month	Product	City	Quantity	Amount
2	April	Bikes	Montreal	12	$4 500,00
3	April	Bikes	Montreal	56	$21 000,00
4	April	Bikes	San Francisco	854	$320 250,00
5	April	Bikes	New York	25	$9 375,00
22	February	Bikes	Montreal	663	$248 625,00
23	February	Bikes	San Francisco	21	$7 875,00
24	February	Bikes	San Francisco	54	$20 250,00
25	February	Bikes	New York	658	$246 750,00
42	January	Bikes	Montreal	75	$28 125,00
43	January	Bikes	Toronto	12	$4 500,00
44	January	Bikes	San Francisco	136	$51 000,00
45	January	Bikes	New York	21	$7 875,00
62	March	Bikes	Montreal	4	$1 500,00
63	March	Bikes	Montreal	522	$195 750,00
64	March	Bikes	San Francisco	125	$46 875,00
65	March	Bikes	New York	212	$79 500,00
66	March	Skates	Montreal	56	$5 544,00
67	March	Skates	Toronto	854	$84 546,00
68	March	Skates	San Francisco	212	$20 988,00
69	March	Skates	New York	56	$5 544,00

Рисунок 14–Запрос с использованием оператора AND и OR

Видим, что в нашу выборку попало за много значений (кроме марта (**March**), также январь (**January**), февраль (**February**) и апрель (**April**)). В чем же причина? А в том, что **SQL** имеет приоритеты выполнения команд. То есть оператор **AND** имеет более высокий приоритет, чем оператор **OR**, поэтому сначала были отобраны записи с коньками, которые проданные в марте, а потом все записи, касающиеся велосипедов.

Итак, чтобы получить правильную выборку, нам нужно изменить приоритеты выполнения команд. Для этого используем **скобки**, как в математике. Тогда, сначала будут обработаны операторы в скобках, а затем - все остальные.

SELECT * FROM Sumproduct WHERE (Product = *'Bikes'* **OR Product** = *'Skates'***) AND Month**= *'March'*

ID	Month	Product	City	Quantity	Amount
62	March	Bikes	Montreal	4	$1 500,00
63	March	Bikes	Montreal	522	$195 750,00
64	March	Bikes	San Francisco	125	$46 875,00
65	March	Bikes	New York	212	$79 500,00
66	March	Skates	Montreal	56	$5 544,00
67	March	Skates	Toronto	854	$84 546,00
68	March	Skates	San Francisco	212	$20 988,00
69	March	Skates	New York	56	$5 544,00

Рисунок 15–Результаты запроса

5. Расширенная фильтрация (оператор IN).

SELECT * FROM Sumproduct WHERE ID IN (4, 12, 58, 67)

ID	Month	Product	City	Quantity	Amount
4	April	Bikes	San Francisco	854	$320 250,00
12	April	Skis Long	San Francisco	663	$162 435,00
58	January	Snow Board	Montreal	522	$160 776,00
67	March	Skates	Toronto	854	$84 546,00

Рисунок 16–Запрос с использованием расширенной фильтрации

Оператор **IN** выполняет ту же функцию, что и **OR**, однако имеет ряд преимуществ:

- При работе с длинными списками, предложение с **IN** легче читать;
- Используется меньшее количество операторов, что ускоряет обработку запроса;
- Самое важное преимущество **IN** в том, что в его конструкции можно использовать дополнительную конструкцию **SELECT**, что открывает большие возможности для создания сложных подзапросов.

6. Расширенная фильтрация (оператор NOT).

SELECT * FROM Sumproduct WHERE NOT City IN ('Toronto', 'Montreal')

ID	Month	Product	City	Quantity	Amount
47	January	Skates	New York	4	$396,00
52	January	Skis Long	San Francisco	4	$980,00
41	February	Snow Board	New York	4	$1 232,00
28	February	Skates	San Francisco	21	$2 079,00
8	April	Skates	San Francisco	25	$2 475,00
77	March	Skis Short	San Francisco	21	$4 389,00
57	January	Skis Short	New York	21	$4 389,00
13	April	Skis Long	New York	21	$5 145,00
72	March	Skis Long	San Francisco	21	$5 145,00

Рисунок 17–Запрос с использованием расширенной фильтрации

Ключевое слово **NOT** позволяет убрать ненужные значения из выборки. Также его особенностью является то, что оно проставляется перед названием столбца, участвующего в фильтровании, а не после.

SQL-Урок 5. Символы подстановки и регулярные выражения (LIKE)

Часто, для фильтрации данных, нам нужно будет осуществить выборку не по точному совпадении условия, а по приближенному значению. То есть когда, например, мы ищем товар, название которого соответствует определенному шаблону или содержит определенные символы или слова. Для таких целей в **SQL** существует оператор **LIKE**, который ищет приближенные значения. Для конструирования такого шаблона используются **метасимволы** (специальные символы для поиска части значения), а именно: "*знак процента*" (**%**) или *звездочка* (*****), "*символ подчеркивания*" (**_**) или "*знак вопроса*" (**?**), "*квадратные скобки*" (**[ ]**).

1. Метасимвол знак процента (%) или звездочка (*)

Давайте из нашей таблицы, например, отберем записи, относящиеся только к товарам, содержащим в своем названии слово ***Skis*** *(лыжи)*. Для этого составим соответствующий шаблон:

SELECT * FROM Sumproduct WHERE Product LIKE *'*Skis*'*

ID	Month	Product	City	Quantity	Amount
10	April	Skis Long	Montreal	854	$209 230,00
11	April	Skis Long	Toronto	25	$6 125,00
12	April	Skis Long	San Francisco	663	$162 435,00
13	April	Skis Long	New York	21	$5 145,00
14	April	Skis Short	Montreal	21	$4 389,00
15	April	Skis Short	Toronto	4	$836,00
16	April	Skis Short	San Francisco	522	$109 098,00
17	April	Skis Short	New York	136	$28 424,00
30	February	Skis Long	Montreal	522	$127 890,00
31	February	Skis Long	Toronto	125	$30 625,00
32	February	Skis Long	San Francisco	663	$162 435,00
33	February	Skis Long	New York	21	$5 145,00

Рисунок 18–Запрос с использованием (*)

Как видим, **СУБД** отобрала только те записи, где в колонке **Product** были товары, содержащие слово ***Skis***. Также отметим, что в данном примере используется метасимвол "звездочка" (*****), поскольку **СУБД Access** не поддерживает "знак процента" (**%**) для оператора **LIKE**.

2. Метасимвол знак подчеркивания (_) или знак (?)

Знак подчеркивания или вопросительный знак применяется для того, чтобы заменить один символ в слове. Давайте в слове ***Bikes*** заменим все гласные буквы на "вопросительный знак" (**?**) и посмотрим на результат:

SELECT * FROM Sumproduct WHERE Product LIKE *'B?k?s'*

ID	Month	Product	City	Quantity	Amount
2	April	Bikes	Montreal	12	$4 500,00
3	April	Bikes	Montreal	56	$21 000,00
4	April	Bikes	San Francisco	854	$320 250,00
5	April	Bikes	New York	25	$9 375,00
22	February	Bikes	Montreal	663	$248 625,00
23	February	Bikes	San Francisco	21	$7 875,00
24	February	Bikes	San Francisco	54	$20 250,00
25	February	Bikes	New York	658	$246 750,00
42	January	Bikes	Montreal	75	$28 125,00

Рисунок 19–Запрос с использованием (?)

Мы использовали метасимвол "вопросительный знак" (**?**), поскольку **СУБД Access** не поддерживает "знак подчеркивания" (_) для оператора **LIKE**.

3. Метасимвол квадратные скобки ([ ])

Метасимвол "квадратные скобки" (**[ ]**) используется для одновременного указания набора символов, по которым нужно выполнить поиск.

SELECT * FROM Sumproduct WHERE City LIKE '[TN]*'

ID	Month	Product	City	Quantity	Amount
5	April	Bikes	New York	25	$9 375,00
7	April	Skates	Toronto	854	$84 546,00
9	April	Skates	New York	663	$65 637,00
11	April	Skis Long	Toronto	25	$6 125,00
13	April	Skis Long	New York	21	$5 145,00
15	April	Skis Short	Toronto	4	$836,00
17	April	Skis Short	New York	136	$28 424,00
19	April	Snow Board	Toronto	522	$160 776,00
21	April	Snow Board	New York	663	$204 204,00

Рисунок 21–Запрос с использованием ([ ])

В примере выше, мы отобрали записи, где в поле **City** названия городов начинаются с буквы **T** или **N**. Также, в данном случае, мы можем использовать еще один метасимвол, который выполняет обратное действие. Добавим в наше

регулярное выражение восклицательный знак (**!**), что будет означать "не равно" (для **СУБД Access**) или знак степени (^) (для других **СУБД**).

SELECT * FROM Sumproduct WHERE City LIKE '[!TN]*'

ID	Month	Product	City	Quantity	Amount
2	April	Bikes	Montreal	12	$4 500,00
3	April	Bikes	Montreal	56	$21 000,00
4	April	Bikes	San Francisco	854	$320 250,00
6	April	Skates	Montreal	56	$5 544,00
8	April	Skates	San Francisco	25	$2 475,00
10	April	Skis Long	Montreal	854	$209 230,00
12	April	Skis Long	San Francisco	663	$162 435,00
14	April	Skis Short	Montreal	21	$4 389,00
16	April	Skis Short	San Francisco	522	$109 098,00

Рисунок 22–Запрос с использованием [!TN]*

То есть, последний созданный нами запрос будет читаться как: выбрать все колонки из таблицы **Sumproduct** и только те записи, где в поле **City** названия городов не начинаются на буквы **T** или **N**. Дополнительно отметим, что набор букв в метасимволе "квадратные скобки" отвечает только за одну позицию в тексте.

Мы можем получить аналогичный результат, если воспользоваться уже известным нам оператором **NOT**, однако с восклицательным знаком (**!**) запись будет короче.

SQL-Урок 6. Вычисляемые поля

Для чего нужно использовать расчетные поля? Как правило, информация в БД представлена в разрезе отдельных фрагментов, поскольку так легче структурировать данные и оперировать ими. Однако нам часто будет нужно использовать не отдельные части данных, а уже соединенную и обработанную информацию. Например, часто необходимо сочетать имя и фамилию клиентов, сочетать элементы адресов, которые находятся в разных столбцах таблицы, обрабатывать текст и отдельные слова, буквы и символы, суммировать общую стоимость покупки, отображать статистику по информации, находящейся в БД. Данные обычно хранятся отдельными "кусками", что требует их дополнительной обработки на стороне клиентского приложения. Однако есть возможность получать уже обработанную информацию с помощью СУБД. Именно в этом случае помогают расчетные поля. Они автоматически создаются при выполнении запроса и имеют вид и свойства обычных столбцов, которые уже имеются в таблице. Единственное отличие заключается в том, что физически расчетных полей нет, поэтому они не занимают дополнительного места в БД, а временно существуют в "оперативной памяти" СУБД.

Преимуществом выполнения операций на стороне СУБД является скорость обработки данных.

1. Выполнение математических операций

Одним из способов использования расчетных полей является выполнение математических операций над выбранными данными. Давайте на примере рассмотрим как это происходит, использовав снова нашу таблицу **Sumproduct**. Предположим, нам нужно вычислить среднюю цену приобретения каждого товара. Для этого нужно переделить колонку **Amount** (сумма) на **Quantity** (количество):

SELECT DISTINCT Product, Amount/Quantity FROM Sumproduct

Product	Expr1001
Bikes	375
Skates	99
Skis Long	245
Skis Short	209
Snow Board	308

Рисунок 23–Запрос с использованием математических операций

Как видим, СУБД отобрала все наименования товаров и отобразила их среднюю стоимость в отдельном столбце, который был создан во время выполнения запроса. Также можно заметить, что мы использовали дополнительный оператор **DISTINCT**, который нам нужен для отображения уникальных названий товаров (без него мы бы получили дублирование записей).

2. Использование псевдонимов

В предыдущем примере мы рассчитывали среднюю стоимость покупки каждого товара и отобразили значение в расчетном столбце. Однако в дальнейшем, нам неудобно обращаться к этому полю, так как его название является неинформативным для нас (СУБД дала название полю - **Expr1001**). Однако мы можем назвать поле самостоятельно, заранее указав его название в запросе, то есть дать псевдоним. Давайте перепишем предыдущий пример и укажем псевдонима для расчетного поля:

SELECT DISTINCT Product, Amount/Quantity AS AvgPrice FROM Sumproduct

Product	AvgPrice
Bikes	375
Skates	99
Skis Long	245
Skis Short	209
Snow Board	308

Рисунок 24–Запрос с использованием математических операций

Видим, наше расчетное поле получило собственное название **AvgPrice**. Для этого мы использовали оператор **AS**, после которого указали необходимое нам название. Стоит отметить, что в SQL поддерживаются только основные математические операции: сложение (+), вычитание (-), умножение (*), деление (/). Также для изменения очередности выполнения операции можно использовать круглые скобки.

Часто псевдонимы используют не только чтобы называть расчетные поля, но и для переименования действующих. Это может быть необходимым, если действующее поле имеет длинное название или название не достаточно информативным.

3. Соединение полей (конкатенация)

Кроме математических операций мы можем объединять текст и выводить его в отдельном поле. Давайте рассмотрим, каким образом можно осуществить склеивание (конкатенацию) текста. Имеем такой пример:

SELECT Month + ' ' + Product AS NewField, Quantity FROM Sumproduct

NewField	Quantity
April Bikes	12
April Bikes	56
April Bikes	854
April Bikes	25
April Skates	56
April Skates	854
April Skates	25
April Skates	663
April Skis Long	854
April Skis Long	25
April Skis Long	663
April Skis Long	21

Рисунок 25–Запрос с использованием конкатенации

В этом примере мы соединили значение в двух столбцах и вывели результат в новое поле **NewField**.

SQL-Урок 7. Функции обработки данных

Как и в большинстве языков программирования, в SQL существуют функции для обработки данных. Стоит отметить, что в отличие от SQL-операторов, функции не стандартизованы для всех видов СУБД, то есть для выполнения одних и тех же операции над данными, разные СУБД имеют свои собственные имена функций. Это означает, что код запроса написан в одной СУБД может не работать в другой, и это нужно учитывать в дальнейшем. Больше всего это касается функций для обработки текстовых значений, преобразования типов данных и манипуляций над датами.

Обычно СУБД поддерживается стандартный набор типов функций, а именно:

- Текстовые функции, которые используются для обработки текста (выделение части символов в тексте, определение длины текста, перевод символов в верхний или нижний регистр ...)
- Числовые функции. Используются для выполнения математических операций над числовыми значениями
- Функции даты и времени (осуществляют манипулирования датой и временем, рассчитывают период между датами, проверяют даты на корректность и т.п.)
- Статистические функции (для вычисления максимальных /минимальных значений, средних значений, подсчет количества и суммы ...)
- Системные функции (предоставляют разного рода служебную информацию о СУБД, пользователе и др..).

1. Функции SQL для обработки текста

Реализация SQL в СУБД Access имеет следующие функции для обработки текста:

Таблица 2 Функции для обработки текста:

Знак операции	Значение
LEFT()	Отбирает символы в тексте слева
RIGHT()	Отбирает символы в тексте справа
MID()	Отбирает символы с середины текста
UCase()	Переводит символы в верхний регистр
LCase()	Переводит символы в нижний регистр
LTrim()	Удаляет все пустые символы слева от текста

RTrim()	Удаляет все пустые символы справа от текста
Trim()	Удаляет все пустые символы с обеих сторон текста

Переведем названия товаров в верхний регистр с помощью функции **UCase()**:

SELECT Product, UCase(Product) AS Product_UCase FROM Sumproduct

Product	Product_UCase
Bikes	BIKES
Bikes	BIKES
Bikes	BIKES
Bikes	BIKES
Skates	SKATES
Skates	SKATES
Skates	SKATES
Skates	SKATES
Skis Long	SKIS LONG
Skis Long	SKIS LONG

Рисунок 26–Верхний регистр

Выделим первые три символа в тексте с помощью функции **LEFT()**:

SELECT Product, LEFT(Product, 3) AS Product_LEFT FROM Sumproduct

Product	Product_LEFT
Bikes	Bik
Bikes	Bik
Bikes	Bik
Bikes	Bik
Skates	Ska
Skates	Ska
Skates	Ska
Skates	Ska
Skis Long	Ski
Skis Long	Ski

Рисунок 27– функции LEFT():

2. Функции SQL для обработки чисел

Функции обработки чисел предназначены для выполнения математических операций над числовыми данными. Эти функции предназначены для алгебраических и геометрических вычислений, поэтому они используются значительно реже функций обработки даты и времени. Однако

числовые функции наиболее стандартизированными для всех версий SQL. Давайте взглянем на перечень числовых функций:

Таблица 3 Функции математические:

Знак операции	Значение
SQR()	Возвращает корень квадратный указанного числа
ABS()	Возвращает абсолютное значение числа
EXP()	Возвращает экспоненту указанного числа
SIN()	Возвращает синус указанного угла
COS()	Возвращает косинус указанного угла
TAN()	Возвращает тангенс указанного угла

Мы привели лишь несколько основных функций, однако вы всегда можете обратиться к документации вашей СУБД, чтобы увидеть полный перечень функций, которые поддерживаются с их подробным описанием.

Например, напишем запрос для получения корня квадратного для чисел в столбце **Amount** с помощью функции **SQR()**:

SELECT Amount, SQR(Amount) AS Amount_SQR FROM Sumproduct

Amount	Amount_SQR
$4 500,00	67,08203932499
$21 000,00	144,9137674619
$320 250,00	565,9063526768
$9 375,00	96,82458365519
$5 544,00	74,45804187595
$84 546,00	290,7679487151
$2 475,00	49,74937185533
$65 637,00	256,1971896801
$209 230,00	457,4166590757
$6 125,00	78,26237921249

Рисунок 28– Функции SQR():

3. Функции SQL для обработки даты и времени

Функции манипулирования датой и временем являются одними из важнейших и часто используемых функций SQL. В базах данных значения дат и времени хранятся в специальном формате, поэтому их невозможно использовать напрямую без дополнительной обработки. Каждая СУБД имеет свой набор функций для обработки дат, что, к сожалению, не позволяет переносить их на другие платформы и реализации SQL.

Список некоторых функций для обработки даты и времени в **СУБД Access**:

Таблица 4 Функции SQL для обработки даты и времени

Знак операции	Значение
DatePart()	Возвращает часть даты: год, квартал, месяц, неделя, день, час, минуты, секунды
Year(), Month()	Возвращает год и месяц соответственно
Hour(), Minute(), Second()	Возвращает час, минуты и секунды указанной даты
WeekdayName()	Возвращает название дня недели

Посмотрим на примере как работает функция **DatePart()**:

SELECT Date1, DatePart("m", Date1) AS Month1 FROM Sumproduct

Date1	Month1
11.02.2013	2

Функция **DatePart ()** имеет дополнительный параметр, который нам позволяет отобразить необходимую часть даты. В примере мы использовали параметр **"m"** , который отображает номер месяца (таким же образом мы можем отразить год - **"yyyy"** , квартал - **"q "**, день - **" d "**, неделю - **" w "**, час - **" h "**, минуты - **"n"** , секунды - **"s"** и т.д.).

4. Статистические функции SQL

Статистические функции помогают нам получить готовые данные без их выборки. SQL-запросы с этими функциями часто используются для анализа и создания различных отчетов. Примером таких выборок может быть: определение количества строк в таблице, получение суммы значений по

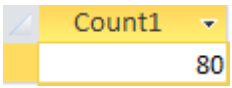
определенному полю, поиск наибольшего /наименьшего или среднего значения в указанном столбце таблицы. Также отметим, что статистические функции поддерживаются всеми СУБД без особых изменений в написании.

Таблица 5 Список статистических функций в **СУБД Access**:

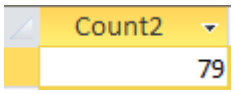
Знак операции	Значение
COUNT()	Возвращает число строк в таблице или столбце
SUM()	Возвращает сумму значений в столбце
MIN()	Возвращает наименьшее значение в столбце
MAX()	Возвращает наибольшее значение в столбце
AVG()	Возвращает среднее значение в столбце

*Примеры использования функции **COUNT()**:*

SELECT COUNT(*) AS Count1 FROM Sumproduct - возвращает количество всех строк в таблице



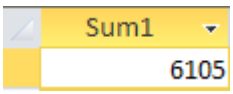
SELECT COUNT(Product) AS Count2 FROM Sumproduct - возвращает количество всех непустых строк в поле **Product**



Мы намеренно удалили одно значение в столбце **Product** , чтобы показать разницу в работе двух запросов.

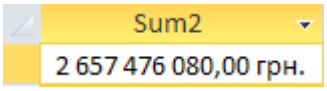
*Примеры использования функции **SUM()**:*

SELECT SUM(Quantity) AS Sum1 FROM Sumproduct WHERE Month = 'April'



Данным запросом мы отразили общее количество проданного товара в апреле.

SELECT SUM(Quantity*Amount) AS Sum2 FROM Sumproduct

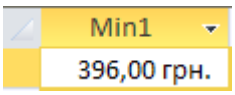


Sum2
2 657 476 080,00 грн.

Как видим, в статистических функциях мы можем осуществлять вычисления над несколькими столбцами с использованием стандартных математических операторов.

*Пример использования функции **MIN()**:*

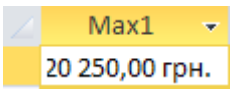
SELECT MIN(Amount) AS Min1 FROM Sumproduct



Min1
396,00 грн.

*Пример использования функции **MAX()**:*

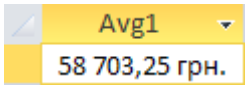
SELECT MAX(Amount) AS Max1 FROM Sumproduct



Max1
20 250,00 грн.

*Пример использования функции **AVG()**:*

SELECT AVG(Amount) AS Avg1 FROM Sumproduct



Avg1
58 703,25 грн.

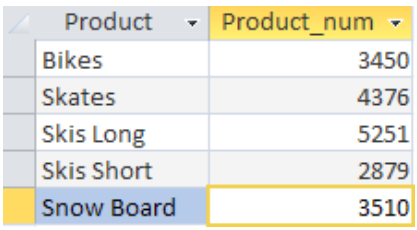
SQL-Урок 8. Группировка данных (GROUP BY)

Группировка данных позволяет разделить все данные на логические наборы, благодаря чему становится возможным выполнение статистических вычислений отдельно в каждой группе.

1. Создание групп (GROUP BY)

Группы создаются с помощью предложения **GROUP BY** оператора **SELECT**. Рассмотрим на примере.

SELECT Product,
SUM(Quantity) AS Product_num FROM Sumproduct GROUP BY Product



Product	Product_num
Bikes	3450
Skates	4376
Skis Long	5251
Skis Short	2879
Snow Board	3510

Рисунок 28– Функции SQR():

Данным запросом мы извлекли информацию о количестве реализованной продукции в каждом месяце. Оператор **SELECT** приказывает вывести два столбца **Product** - название продукта и **Product_num** - расчетное поле, которое мы создали для отображения количества реализованной продукции (формула поля **SUM (Quantity)**). Предложение **GROUP BY** указывает СУБД сгруппировать данные по столбцу **Product**. Стоит также отметить, что **GROUP BY** должен идти после предложения **WHERE** и перед **ORDER BY**.

2. Фильтрующие группы (HAVING)

Так же, как мы фильтровали строки в таблице, мы можем осуществлять фильтрацию по сгруппированным данным. Для этого в **SQL** существует оператор **HAVING**. Возьмем предыдущий пример и добавим фильтрацию по группам.

```
SELECT Product,
SUM(Quantity) AS Product_num FROM Sumproduct GROUP
BY Product HAVING SUM(Quantity)>4000
```

Product	Product_num
Skates	4376
Skis Long	5251

Видим, что после того, как была посчитана количество реализованного товара в разрезе каждого продукта, СУБД "отсекла" те продукты, которых было реализовано меньше 4000 шт.

Как видим, оператор **HAVING** очень похож на оператора **WHERE** , однако между собой они имеют существенное отличие: **WHERE** фильтрует данные до того, как они будут сгруппированы, а **HAVING** - осуществляет фильтрацию после группировки. Таким образом, строки, которые были изъяты предложением **WHERE** НЕ будут включены в группу. Итак, операторы **WHERE** и **HAVING** могут использоваться в одном предложении. Рассмотрим пример:

```
SELECT Product,
SUM(Quantity) AS Product_num FROM Sumproduct WHERE Product<>'Skis
Long' GROUP BY Product HAVING SUM(Quantity)>4000
```

Product	Product_num
Skates	4376

Мы к предыдущему примеру добавили оператор **WHERE**, где указали товар **Skis Long**, что в свою очередь повлияло на группирование оператором **HAVING**. Как результат видим, что товар **Skis Long** не попал в перечень групп с количеством реализованной продукции больше 4000 шт.

3. Группировка и сортировка

Как и при обычной выборке данных, мы можем сортировать группы после группировки оператором **HAVING**. Для этого мы можем использовать уже знакомый нам оператор **ORDER BY**. В данной ситуации его применения аналогичное предыдущим примерам. К примеру:

SELECT Product,
SUM(Quantity) AS Product_num FROM Sumproduct GROUP
BY Product HAVING SUM(Quantity)>3000 ORDER BY SUM(Quantity)

или просто укажем номер поля по порядку, по которому хотим сортировать:

SELECT Product,SUM(Quantity) AS Product_num FROM Sumproduct GROUP BY Product HAVING SUM(Quantity)>3000 ORDER BY 2

Product	Product_num
Bikes	3450
Snow Board	3510
Skates	4376
Skis Long	5251

Рисунок 29– Сортировка

Видим, что для сортировки сводных результатов нам нужно просто прописать предложения с **ORDER BY** после оператора **HAVING**. Однако есть один нюанс. **СУБД Access** не поддерживает сортировку групп по псевдонимами колонок, то есть в нашем примере, чтобы сортировать значения, мы не сможем в конце запроса прописать **ORDER BY Product_num** .

Список литературы

1. Аллен, Г. Тейлор SQL для чайников / Аллен Г. Тейлор. - М.: Диалектика, ильямс, 2015. - 416 с.
2. Бен, Форта SQL за 10 минут / Форта Бен. - М.: Диалектика / Вильямс, 2015. - 673 с
3. Бьюли, А. Изучаем SQL / А. Бьюли. - М.: Символ-плюс, 2014. - 108 с..
4. Грабер, Мартин SQL для простых смертных / Мартин Грабер. - М.: ЛОРИ, 2014. - 378 с.
5. Гудсон, Джон Практическое руководство по доступу к данным (+ DVD-ROM) / Джон Гудсон , Роб Стюард. - М.: БХВ-Петербург, 2013. - 304 с.
6. Дейт, К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL / К.Дж. Дейт. - М.: Символ-плюс, **2017**. - 480 с.
7. Дунаев, В. В. Базы данных. Язык SQL для студента / В.В. Дунаев. - М.: БХВ-Петербург, 2016. - 288 с.
8. Карвин, Билл Программирование баз данных SQL. Типичные ошибки и их устранение / Билл Карвин. - М.: Рид Групп, 2013. - 336 с.