

Тема 4. Язык SQL- Structured Query Language

Часть 1

1. Общие сведения о языке SQL
2. Простейшие запросы
3. Формирование списка вывода
4. Сортировка данных
5. Использование специальных условий
6. Многотабличные запросы
7. Агрегатные функции
8. Операторы UNION, EXCEPT, INTERSECT

Краткая история языка SQL

- **SQL появился в середине 70-х годов в рамках проекта экспериментальной реляционной СУБД System R.**
- **Исходное название SEQUEL (Structured English Query Language)**

Краткая история языка SQL

- *Возможности языка SQL*
 - Средства формулирования запросов и манипулирования БД
 - Средства авторизации доступа к отношениям и их полям
 - Средства определения различных объектов БД

Краткая история языка SQL

- *Возможности языка SQL*
- Средства определения:
 - схемы БД
 - ограничений целостности и триггеров
 - представлений БД
 - структур физического уровня, поддерживающих эффективное выполнение запросов
 - точек сохранения транзакции и выполнения фиксации и откатов транзакции

Краткая история языка SQL

- **Язык SQL в настоящее время реализован во всех коммерческих реляционных СУБД и почти во всех СУБД, которые изначально основывались не на реляционном подходе**
- **Базовый набор операторов SQL в коммерческих реализациях более или менее соответствует стандарту.**

Стандартизация языка SQL

- Для языка SQL была разработана серия последовательных стандартов:
 - **SQL/86** (Первый стандарт языка SQL)
 - **SQL/89 (SQL1)** – четкая стандартизация синтаксиса и семантики операторов выборки данных, операторов манипулирования данными и фиксация средств ограничения целостности БД

Стандартизация языка SQL

- Для языка SQL была разработана серия последовательных стандартов:
 - **SQL/92 (SQL2)** – расширены способы ограничения целостности, введена поддержка различных языков программирования, предусмотрена обработка транзакций и др.
 - **SQL/99 (SQL3)** – поддержка модели данных, выходящую за границу реляционной и др. (последняя модификация – **SQL:2008**)

Интерактивный и встроенный SQL

- В стандарте предусмотрено существование двух форм языка: **интерактивной** и **встроенной**.
- *Интерактивная* – предполагает работу непосредственно в среде *SQL* и не требует наличия какой-либо другой программной среды. Считается, что запросы формулируются в режиме диалога и выполняются интерпретатором языка.

Интерактивный и встроенный SQL

- В стандарте предусмотрено существование двух форм языка: **интерактивной** и **встроенной**.
- *Встроенная* форма – язык встраивается в другую языковую систему и общается с пользователем через нее. То есть, интерфейс обеспечивает среда другого языка, а *SQL* выполняет запросы, поступающие и программы.

Состав языка SQL

- **DML** (Data Manipulation Language) – язык манипулирования данными – служит для манипуляции данными
- **Основные команды DML**
 - **SELECT**
 - **INSERT**
 - **UPDATE**
 - **DELETE**

Состав языка SQL

- **DDL** (Data Definition Language) – язык определения данных – служит для определения схем данных (таблиц)
- Основные команды языка DDL
 - **CREATE table** (view, function, procedure, index и др)
 - **ALTER table** (view, function, procedure, index и др)
 - **DROP table** (view, function, procedure, index и др)

Состав языка SQL

- **DCL** (Data Control Language) – язык управления данными – служит для управления доступом к данным
- **Основные команды DCL**
 - **GRANT**
 - **REVOKE**

Пример БД

Salespeople	
<i>snum</i>	<i>Номер продавца</i>
sname	Имя
city	Город
comm	КОМИССИОН НЫЕ

Products	
<i>pnum</i>	<i>Номер продукта</i>
pname	наименование
prise	Цена

Customers	
<i>cnum</i>	<i>Номер заказчика</i>
cname	Имя
city	Город
rating	Рейтинг
Snum	Основной продавец

Orders	
<i>onum</i>	<i>Номер заказа</i>
amt	Стоимость
snum	Номер_продавца
cnum	Номер_заказчика
pnum	Номер_продукта
odate	Дата заказа

Простейшие SELECT запросы

Простейшие SELECT запросы

- Оператор SELECT (выбрать) языка SQL является самым важным и часто используемым оператором.
- Посредством данной команды можно получить любые операции реляционной алгебры

Простейшие SELECT запросы

- синтаксис команды **SELECT**:

SELECT [**DISTINCT**|**ALL**] <список атрибутов>

FROM <список таблиц| представлений и др>

[**WHERE** <условие выборки>]

[**GROUP BY** <список атрибутов>]

[**HAVING** <условие>]

[**UNION** <выражение **SELECT**>]

[**ORDER BY** <список атрибутов>]

Простейшие SELECT запросы

Список атрибутов может включать:

1. отдельные или все (*) поля таблицы – в случае (*) - поля выводятся в порядке, в котором они определены в исходной таблице
2. Литералы, Константы, Выражения
3. Функции
4. Скалярные подзапросы
5. и др.

Простейшие SELECT запросы

- **Пример 1.** Вывести информацию по всем продавцам из Москвы

- *Операция?* **Выборка**

```
SELECT *  
FROM Salespeople  
WHERE city = 'Москва'
```

Результат:

snum	sname	city	comm
1001	Иванов	Москва	.12
1004	Сидоров	Москва	.11

Простейшие SELECT запросы

- **Пример 2.** Вывести всю информацию по заказчикам в следующем порядке: Имя, рейтинг, город, номер заказчика, номер продавца

- Операция? **Проекция**

```
SELECT cname, rating, city, cnum, snum  
FROM Customers
```

Простейшие SELECT запросы

- **Пример 3. Вывести города, в которых проживают продавцы**

- **Операция? Проекция**

Результат:

```
SELECT city  
FROM Salespeople
```

city
Москва
Новгород
Москва
Самара
Иркутск
Иркутск

Простейшие SELECT запросы

- Устранение избыточности данных – **DISTINCT**
- Обратный параметр **ALL** – обеспечивает вывод всей информации. Принимается по умолчанию

Простейшие SELECT запросы

- **Пример 4. Вывести города, в которых проживают продавцы**

Результат:

```
SELECT DISTINCT city  
FROM Salespeople
```

city
Москва
Новгород
Самара
Иркутск

Простейшие SELECT запросы

- **Пример 5. Вывести номера заказчиков и продавцов, у которых общие заказы**

```
SELECT DISTINCT snum, cnum  
FROM Orders
```

Результат?

Результат:

DISTINCT

cnum	snum
2008	1007
2001	1001
2007	1004
2003	1002
2002	1003
2004	1002
2006	1001
2004	1002

без DISTINCT

cnum	snum
2008	1007
2001	1001
2007	1004
2003	1002
2008	1007
2002	1003
2004	1002
2006	1001
2004	1002
2006	1001

Самостоятельно!

- 1. Напишите запрос, который может выдать вам все заказы со значениями суммы выше \$1,000.**
- 2. Напишите запрос, который может выдать вам поля sname и city для всех продавцов из Саратова с комиссиянными выше .10.**
- 3. Напишите запрос к таблице Заказчиков, чей вывод включит всех заказчиков с оценкой ≤ 100 , если они не находятся в Москвы.**
- 4. Что может быть выведено в результате следующего запроса? `SELECT * FROM Orders WHERE (amt < 1000 OR NOT (odate = 10/03/1990 AND cnum > 2003));`**
- 5. Что может быть выведено в результате следующего запроса? `SELECT * FROM Orders WHERE NOT ((odate = 10/03/1990 OR snum > 1006) AND amt >= 1500);`**
- 6. Как можно проще переписать такой запрос? `SELECT snum, sname, city, comm FROM Salespeople WHERE (comm > + .12 OR comm < .14);`**

Формирование вывода запросов

Формирование вывода запросов

Использование выражений

- унарный оператор « - » (знак минус) меняет знак выражения на противоположный;
- бинарные операторы « + », « - », « * », « / » предоставляют возможность выполнения арифметических действий;
- операция конкатенации строк + (||) позволяет «склеивать» значения двух и более строк.

Формирование вывода запросов

- **Пример 6. Вывести информацию по
комиссионным продавцов в
процентном отношении**

```
SELECT sname, city, comm*100  
FROM Salespeople
```

Результат:

sname	city	EXPR1
Иванов	Москва	12
Петров	Новгород	13
Сидоров	Москва	11
Краснов	Самара	15
Симонов	Иркутск	10



Столбец вывода

Формирование вывода запросов

- **Столбцы вывода** – это столбцы данных, созданные в запросе.
 - создаются всегда, когда в запросе используются выражения, константы, функции и др (любые данные, за исключением полей таблицы)
 - не имеют имени, поэтому в результирующем множестве именуются автоматически средствами СУБД
 - в разных СУБД имена по умолчанию могут отличаться

Формирование вывода запросов


- **Пример.** Вывести информацию по продавцам в следующем виде:

Продавец - имя, проживает в город

```
SELECT 'Продавец - ' + sname,  
       ' проживает в ' + city  
FROM Salespeople
```

Результат:

EXPR1	EXPR2
Продавец - Иванов	проживает в Москва
Продавец - Петров	проживает в Новгород
Продавец -Сидоров	проживает в Москва
Продавец - Краснов	проживает в Самара
Продавец - Симонов	проживает в Иркутск



**Столбцы
вывода**

Вывод выражений

- **Пример 7.** Вывести информацию по
комиссионным продавцов в
процентном отношении со знаком %

```
SELECT sname, city, ' % ', comm*100  
FROM Salespeople
```

Результат:

sname	city	EXPR1	EXPR2
Иванов	Москва	%	12
Петров	Новгород	%	13
Сидоров	Москва	%	11
Краснов	Самара	%	15
Симонов	Иркутск	%	10

Формирование вывода запросов

- Для именования столбцов вывода, а также для переименования существующих полей таблицы используется слово **AS** – это столбцы данных, созданные в запросе.

{Имя поля|выражение} **AS** {новое имя}

Вывод выражений

```
SELECT sname, city, ' % ',  
       comm*100 AS Procent  
FROM Salespeople
```

Результат:

sname	city	Expr1	Procent
Иванов	Москва	%	12
Петров	Новгород	%	13
Сидоров	Москва	%	11
Краснов	Самара	%	15
Симонов	Иркутск	%	10

Формирование вывода запросов

Упорядочивание вывода полей ORDER BY

- **Синтаксис:**

ORDER BY поле [{ASC | DESC}]

- **ASC (по умолчанию) - сортировка по возрастанию**
- **DESC - сортировка по убыванию**
- **Важно: Поля сортировки должны обязательно присутствовать в списке вывода **SELECT****

ORDER BY

```
SELECT sname, city, '%', comm*100 AS Procent  
FROM Salespeople  
ORDER BY sname
```

Результат:

sname	city	Expr1	Procent
Иванов	Москва	%	12
Краснов	Самара	%	15
Петров	Новгород	%	13
Сидоров	Москва	%	11
Симонов	Иркутск	%	10

ORDER BY

```
SELECT sname, city, ' % ',  
        comm*100 AS Procent  
  
FROM Salespeople  
  
ORDER BY snum
```

Результат?

ОШИБКА!

SNUM нет в списке SELECT

ORDER BY

```
SELECT DISTINCT odate, snum, cnum,  
FROM Orders  
WHERE amt > 2000  
ORDER BY odate DESC, cnum, snum
```

Результат?

odate	snum	cnum
10/08/2016	1002	2003
27/05/2015	1001	2006
22/05/2015	1001	2006

Формирование вывода запросов

Упорядочивание вывода полей ORDER BY

- Синтаксис2:
ORDER BY номер столбца [{ASC | DESC}]
- *Номер столбца* – это порядковый номер столбца в списке SELECT!
- Такой вариант ORDER BY чаще используется для упорядочивания столбцов вывода

ORDER BY

```
SELECT sname, city, ' % ',  
       comm*100 AS Procent  
FROM Salespeople  
ORDER BY 4 DESC, 1 ASC, city
```

Результат?

sname	city	Expr1	Procent
Краснов	Самара	%	15
Петров	Новгород	%	13
Иванов	Москва	%	12
Сидоров	Москва	%	11
Симонов	Иркутск	%	10

Использование специальных операторов в условиях

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- Оператор **IN** позволяет определять принадлежность значения поля некоторому множеству значений
- Синтаксис:
поле **IN** (значение1, значение2, ...)
- **IN** не чувствителен к порядку следования значений!

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- **Пример 8.**

**Вывести информацию
продавцам из Самары,
Новгорода и Иркутска**

Пример 8. Сравните

```
SELECT *  
FROM Salespeople  
WHERE city = 'Самара' or city =  
'Новгород' or city='Иркутск'
```

```
SELECT *  
FROM Salespeople  
WHERE city IN ('Самара', 'Новгород',  
              'Иркутск')
```

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- **Пример 9.** Вывести всех заказчиков, основные продавцы которых имеют номера: 1001, 1007, 1004

```
SELECT *
```

```
FROM Customers
```

```
WHERE snum IN (1001, 1007, 1004)
```

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- Оператор **BETWEEN** позволяет определять принадлежность некоторому диапазону значений
- Синтаксис:
поле **BETWEEN** начал_значение **AND**
конеч_значение

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- Оператор **BETWEEN**

ВАЖНО:

- 1. начальное значение должно быть меньше конечного**
- 2. Значения, совпадающие с границами диапазона, включаются в результат запроса!**

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- **Пример 10.** Вывести всех продавцов, комиссионные которых находятся в диапазоне 0.10 и 0.14. Границы диапазона не включать!

```
SELECT *  
FROM Salespeople  
WHERE comm BETWEEN 0.10 AND 0.14  
AND NOT comm IN (0.10, 0.14)
```

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- **Пример 11. Вывести всех заказчиков, первые буквы имен которых - в диапазоне от «А» до «Л»**

```
SELECT *
```

```
FROM Customers
```

```
WHERE cname BETWEEN 'А' AND 'Л'
```

**Исходная
таблица**

snum	sname	city	rating	snum
2001	Сергеев	Москва	100	1001
2002	Москвин	Саратов	200	1003
2003	Ломов	Новгород	200	1002
2004	Градов	Белгород	300	1002
2006	Кольцов	Москва	100	1001
2008	Новиков	Новгород	300	1007
2007	Дятлов	Саратов	100	1004

Результат?

snum	sname	city	rating	snum
2004	Градов	Белгород	300	1002
2006	Кольцов	Москва	100	1001
2007	Дятлов	Саратов	100	1004

Специальные операторы IN, BETWEEN, LIKE, IS NULL

```
SELECT *  
FROM Customers  
WHERE cname BETWEEN 'A' AND 'M'  
and cname <> 'M'
```

Результат

snum	cname	city	rating	snum
2004	Градов	Белгород	300	1002
2006	Кольцов	Москва	100	1001
2007	Дятлов	Саратов	100	1004
2003	Ломов	Новгород	200	1002

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- Оператор **LIKE** - используется для поиска по шаблону по полям типа строковых CHAR, VARCHAR.
- Синтаксис:
поле **LIKE** шаблон [**ESCAPE** символ]

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- Оператор **LIKE**

ТИПЫ ШАБЛОНОВ:

1. «**_**» - замещает любой одиночный СИМВОЛ
2. «**%**» – замещает последовательность СИМВОЛОВ
3. «**[]**» - Любой символ из перечисленных в скобках
4. «**[^]**» - Любой символ, кроме перечисленных в квадратных скобках

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- **Пример 12.** Вывести всех продавцов, имена которых начинаются с буквы «С»

```
SELECT *  
FROM Salespeople  
WHERE sname LIKE 'С%'
```

Результат

snum	sname	city	comm
1004	Сидоров	Москва	.11
1003	Симонов	Иркутск	.10

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- **Пример 13.** Вывести всех продавцов, в именах которых есть буквы «А» и «С»

```
SELECT *  
FROM Salespeople  
WHERE sname LIKE '%A%C%'
```

Результат

snum	sname	city	comm
------	-------	------	------

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- **Пример 14.** Вывести всех продавцов, в именах которых третья буква «А» «Д» или «Л»

```
SELECT *  
FROM Salespeople  
WHERE sname LIKE '___[АДЛ]%'
```

Результат

snum	sname	city	comm
------	-------	------	------

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- **Пример 14.** Вывести всех продавцов, в именах которых 2-я буква не «И»

```
SELECT *  
FROM Salespeople  
WHERE sname LIKE '_[^И]%'
```

Специальные операторы IN, BETWEEN, LIKE, IS NULL

Оператор LIKE - параметр ESCAPE

- **ESCAPE-СИМВОЛ**
 - позволяет интерпретировать символ-шаблона «%» (или «_») как обычный символ
 - должен быть одиночным и применяется к непосредственно расположенному за ним символу

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- **Пример 15.** Вывести всех продавцов, в именах которых есть символ «_»

```
SELECT *  
FROM Salespeople  
WHERE sname LIKE '%/_%' ESCAPE '/'
```

Результат

snum	sname	city	comm
------	-------	------	------

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- Оператор **LIKE** - параметр **ESCAPE**
- **ESCAPE-символ** может использоваться самостоятельно.
- В этом случае символ вводится дважды.

Специальные операторы IN, BETWEEN, LIKE, IS NULL

Оператор LIKE - параметр **ESCAPE**

- **Пример 16.** Вывести всех продавцов, в именах которых есть символы «_ /»

```
SELECT *
```

```
FROM Salespeople
```

```
WHERE sname LIKE '%/_//%' ESCAPE '/'
```

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- Оператор **IS NULL** используется для проверки NULL значений.

Синтаксис: поле **IS NULL**

Важно:

Результат операций содержащих NULL
(поле **= NULL** , поле **IN (NULL)** и др.)
будет не известен в любом случае

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- **Пример 16. Вывести всех продавцов, у которых город неизвестен**

```
SELECT *  
FROM Salespeople  
WHERE city IS NULL
```

Специальные операторы IN, BETWEEN, LIKE, IS NULL

- **Пример 17. Вывести всех продавцов, у которых нет NULL-значений в поле город**

```
SELECT *  
FROM Salespeople  
WHERE NOT city IS NULL
```

Самостоятельно

- 1. Напишите два запроса, которые могли бы вывести все заказы на 3 или 4 октября 2010.**
- 2. Напишите запрос, который выберет всех заказчиков, обслуживаемых продавцами Петров или Сидоров.**
- 3. Напишите запрос, который может вывести всех заказчиков, чьи имена начинаются с буквы, попадающей в диапазон от А до К.**
- 4. Напишите запрос, который выберет всех пользователей, чьи имена не содержат букву Е и заканчиваются на букву В.**
- 5. Напишите запрос, который выберет все заказы, имеющие нулевые значения или NULL в поле amt (сумма).**