Лекция 5.

Язык SQL - продолжение

Условная логика

Условная логика в SQL

- Стандартный оператор CASE … WHEN … END.
- Встроенная функция, имитирующая if-then-else. Зависит от СУБД, в SQLite функция IIF().

Условный оператор CASE

Проверяет условие и возвращает один из возможных вариантов.

Первая форма:	Вторая форма:

CASE
  WHEN условие_1
  THEN возвращаемое_значение_1
  …
  WHEN условие_N
  THEN возвращаемое_значение_N
  [ELSE возвращаемое_значение]
END
	CASE проверяемое_значение
	  WHEN сравниваемое_значение_1
	  THEN возвращаемое_значение_1
	  …
	  WHEN сравниваемое_значение_N
	  THEN возвращаемое_значение_N
	  [ELSE возвращаемое_значение]
	END

Можно использовать в операторах SELECT, INSERT, UPDATE, DELETE.

SELECT: условный оператор CASE

Проверяет условие и возвращает один из возможных вариантов.

```
SELECT id, name AS 'Фамилия',
  CASE city
    WHEN 'Саранск' THEN 'Да'
    ELSE 'Нет'
  END 'Земляк'
FROM customer;

SELECT id, name AS 'Фамилия',
  CASE
    WHEN rating >= 300 THEN 'Молодец'
    WHEN rating >= 200 THEN 'Неплохо'
    ELSE 'Старайся больше'
  END 'Оценка по рейтингу'
FROM customer;
```

Условный оператор CASE

Возвращаемым значение может быть результат подзапроса.

```
SELECT v.name,
  CASE
    WHEN v.city='Саранск' THEN
      (SELECT count(*) FROM `order` o
       WHERE v.id=o.vendor_id)
    ELSE 'Не из Саранска'
  END sales
FROM vendor v;
```

Аналитические функции

Структура оператора SELECT

SELECT <выражения>

FROM <таблицы>

JOIN <другая таблица> **ON** <условие>

WHERE <предикаты>

GROUP BY <выражения>

HAVING <предикаты>

ORDER BY <выражения>

LIMIT <количество строк>

Структура оператора SELECT

SELECT Определяет, какие столбцы включить в результирующий набор

FROM Определяет таблицы, из которых выбираются данные, и которые нужно соединить друг с другом

WHERE Отсеивает ненужные данные

GROUP BY Используется для группировки строк по общим значениям столбцов

HAVING Отсеивает ненужные группированные данные

ORDER BY Сортирует строки результирующего набора по одному или нескольким столбцам

LIMIT Ограничивает количество строк в результирующем наборе

Аналитические запросы

Результирующий набор строк, полученный после выполнения всех шагов запроса (соединение, фильтрация, группировка, сортировка), можно дополнительно анализировать и добавить дополнительные столбцы.

Можно разбить результирующий набор данных на **окна** - набор строк, в которых происходит вычисление функции. Окна данных могут содержать от одной строки до всех строк из результирующего набора данных.

Результаты работы оконных функций добавляются к выборке как еще одно поле.

Оконные функции

функция(выражение) **OVER** ([PARTITION BY выражения] [ORDER BY выражения])

Оконные функции:
ROW_NUMBER(), RANK(), FIRST_VALUE(), LAST_VALUE(), ...
Также могут использоваться функции агрегирования SUM(), MIN, MAX, ...

```
SELECT *, ROW_NUMBER() OVER () AS number FROM customer;

SELECT *, ROW_NUMBER() OVER (ORDER BY rating DESC) AS rating_place FROM customer ORDER BY name;

SELECT *, RANK() OVER (ORDER BY rating DESC) AS rank FROM customer ORDER BY name;

SELECT *, MAX(rating) OVER () FROM customer;

SELECT name, city, rating, rating-AVG(rating) OVER () AS delta_rating FROM customer;

SELECT *, COUNT(*) OVER (PARTITION BY city) AS customers_in_city FROM customer;
```

Объединение результатов запросов

UNION

Объединение строк из нескольких результирующих наборов данных.

- Количество столбцов в каждом запросе одно и то же.
- Столбцы в каждом запросе имеют совместимые типы.
- Имена столбцов в первом запросе задают имена для всего объединения.
- ORDER BY применяется ко всему набору.
- GROUP BY и HAVING применяются к каждому подзапросу.

Вывести всех покупателей и продавцов с указанием их роли и города, где они живут. Сортировать по имени города и фамилии человека.

```
SELECT name, city, 'Продавец' AS 'Роль' FROM vendor
UNION
SELECT name, city, 'Покупатель' FROM customer
ORDER BY city, name;
```

UNION и UNION ALL

UNION убирает в результирующем наборе повторяющиеся строки.

UNION ALL оставляет все строки.

```
SELECT 1, 2
  UNION
SELECT 1, 2
  UNION
SELECT 3, 4;
```

```
1 | 2
——-
1 | 2
3 | 4
```

```
SELECT 1, 2
  UNION ALL
SELECT 1, 2
  UNION ALL
SELECT 3, 4;
```

```
1 | 2
——-
1 | 2
1 | 2
3 | 4
```

Представления (view)

Представления

Объект базы данных, являющийся результатом выполнения запроса к базе данных, определенного с помощью оператора SELECT, в момент обращения к представлению.

- Упрощение запросов.
- Гибкая настройка прав доступа к данным (права даются не на таблицу, а на представление).
- Разделение логики хранения данных и программного обеспечения.

CREATE VIEW имя [столбцы]
AS выборка

```
CREATE VIEW orders_with_names
AS SELECT `order`.order_date, customer.name AS customer_name,
vendor.name AS vendor_name, `order`.summa
FROM `order`, customer, vendor
WHERE `order`.customer_id = customer.id AND `order`.vendor_id = vendor.id;
```

Обобщенные табличные выражения (CTE)

Обобщенные табличные выражения

Common table expressions, CTE

CTE - именованный подзапрос, указываемый в предложении WITH в верхней части запроса SELECT, INSERT, UPDATE или DELETE. Действует аналогично временному представлению (view).

- Запрос может содержать несколько CTE.
- CTE может обращаться к другому CTE, определенному над ним.

Подзапросы и CTE

Запрос с подзапросом:

```
SELECT a, b, c
FROM (X)
WHERE e = f
```

превращается в:

```
WITH cte AS (X)
SELECT a, b, c
FROM cte
WHERE e = f
```

CTE похожи на функции в языках программирования.

- Запрос становится одноуровневым - проще понимать.
- CTE можно использовать повторно в пределах запроса.

Обычные и рекурсивные CTE

Обычные CTE:

- Действуют как представления (view) с учетом того, что к CTE можно обращаться только в том запросе, где они объявлены.
- Заменяют подзапросы, позволяя упростить структуру запроса.

Рекурсивные CTE (обращаются сами к себе):

- Генерирование последовательности значений.
- Выполнение сложных преобразования данных.
- Обработка иерархических данных.

Рекурсивные CTE

```
WITH [RECURSIVE] cte-table-name AS (
  initial-select
  UNION [ALL]
  recursive-select)
cte-select
```

```
WITH RECURSIVE ten(x) AS (
  SELECT 1
    UNION ALL
  SELECT x+1 FROM ten WHERE x<10)
SELECT * FROM ten;
```

x
1
2
3
5
6
7
8
9
10

Рекурсивные CTE

```
WITH [RECURSIVE] cte-table-name AS (
  initial-select
  UNION [ALL]
  recursive-select)
cte-select
```

1. Выполняется начальная выборка (initial-select), результат помещается в очередь и в результирующую таблицу.

2. Пока в очереди есть элементы, из очереди извлекается одна строка S и добавляется в рекурсивную таблицу.
3. Выполняется рекурсивная выборка (recursive-select), при этом считается, что рекурсивная таблица состоит из одной строки S. Результат выборки добавляется в очередь и в результирующую таблицу.

4. Переходим к шагу 2.

LIMIT в рекурсивном запросе определяет максимальное число строк, которые добавляются в рекурсивную таблицу на шаге 2.

Рекурсивные CTE

1

```
WITH RECURSIVE ten(x) AS (
  SELECT 1
    UNION ALL
  SELECT x+1 FROM ten WHERE x<5)
SELECT * FROM ten;
```

Очередь

x
1

Рекурсивная ten

x

Результат **ten**

x
1

Рекурсивные CTE

2

```
WITH RECURSIVE ten(x) AS (
  SELECT 1
    UNION ALL
  SELECT x+1 FROM ten WHERE x<5)
SELECT * FROM ten;
```

Очередь

x
—

Рекурсивная ten

x
—
1

Результат **ten**

x
—
1

Рекурсивные CTE

3

```
WITH RECURSIVE ten(x) AS (
  SELECT 1
    UNION ALL
  SELECT x+1 FROM ten WHERE x<5)
SELECT * FROM ten;
```

Очередь

x
2

Рекурсивная ten

x
1

Результат **ten**

x
1
2

Рекурсивные CTE

2

```
WITH RECURSIVE ten(x) AS (
  SELECT 1
    UNION ALL
  SELECT x+1 FROM ten WHERE x<5)
SELECT * FROM ten;
```

Очередь

x
—

Рекурсивная ten

x
—
2

Результат **ten**

x
—
1
2

Рекурсивные CTE

3

```
WITH RECURSIVE ten(x) AS (
  SELECT 1
    UNION ALL
  SELECT x+1 FROM ten WHERE x<5)
SELECT * FROM ten;
```

Очередь

x
3

Рекурсивная ten

x
2

Результат **ten**

x
1
2
3

Рекурсивные CTE

2

```
WITH RECURSIVE ten(x) AS (
  SELECT 1
    UNION ALL
  SELECT x+1 FROM ten WHERE x<5)
SELECT * FROM ten;
```

Очередь

x
—

Рекурсивная ten

x
—
3

Результат **ten**

x
—
1
2

Рекурсивные CTE

3

```
WITH RECURSIVE ten(x) AS (
  SELECT 1
    UNION ALL
  SELECT x+1 FROM ten WHERE x<5)
SELECT * FROM ten;
```

Очередь

x
4

Рекурсивная ten

x
3

Результат **ten**

x
1
2
3
4

Рекурсивные CTE

2

```
WITH RECURSIVE ten(x) AS (
  SELECT 1
    UNION ALL
  SELECT x+1 FROM ten WHERE x<5)
SELECT * FROM ten;
```

Очередь

x

Рекурсивная ten

x
4

Результат **ten**

x
1
2
3
4

Рекурсивные CTE

3

```
WITH RECURSIVE ten(x) AS (
  SELECT 1
    UNION ALL
  SELECT x+1 FROM ten WHERE x<5)
SELECT * FROM ten;
```

Очередь

x
5

Рекурсивная ten

x
4

Результат **ten**

x
1
2
3
4
5

Рекурсивные CTE

2

```
WITH RECURSIVE ten(x) AS (
  SELECT 1
    UNION ALL
  SELECT x+1 FROM ten WHERE x<5)
SELECT * FROM ten;
```

Очередь

x
—

Рекурсивная ten

x
—
5

Результат **ten**

x
—
1
2
3
4
5

Рекурсивные CTE

3

```
WITH RECURSIVE ten(x) AS (
  SELECT 1
    UNION ALL
  SELECT x+1 FROM ten WHERE x<5)
SELECT * FROM ten;
```

Очередь

x

Рекурсивная ten

x
5

Результат **ten**

x
1
2
3
4
5

Обработка иерархических структур

Отдел 1
- Сотрудник 1
- Сотрудник 2
- . . .
- **Сектор 1**
  - Сотрудник 1
  - Сотрудник 2
  - . . .
- . . .

Отдел 2
- Сотрудник 1
- . . .
- **Сектор 1**
  - Сотрудник 1
  - Сотрудник 2
  - . . .
- . . .

Необходимо выделить всех сотрудников определенного отдела (вне зависимости от уровня вложенности)

Практическое применение SQL

Обработка и анализ наборов данных нужны многим (аналитики, разработчики, администраторы, тестировщики, менеджеры, …)

andrey@mono:~/projects/mrsu-examples

Файл Правка Вид Поиск Терминал Справка

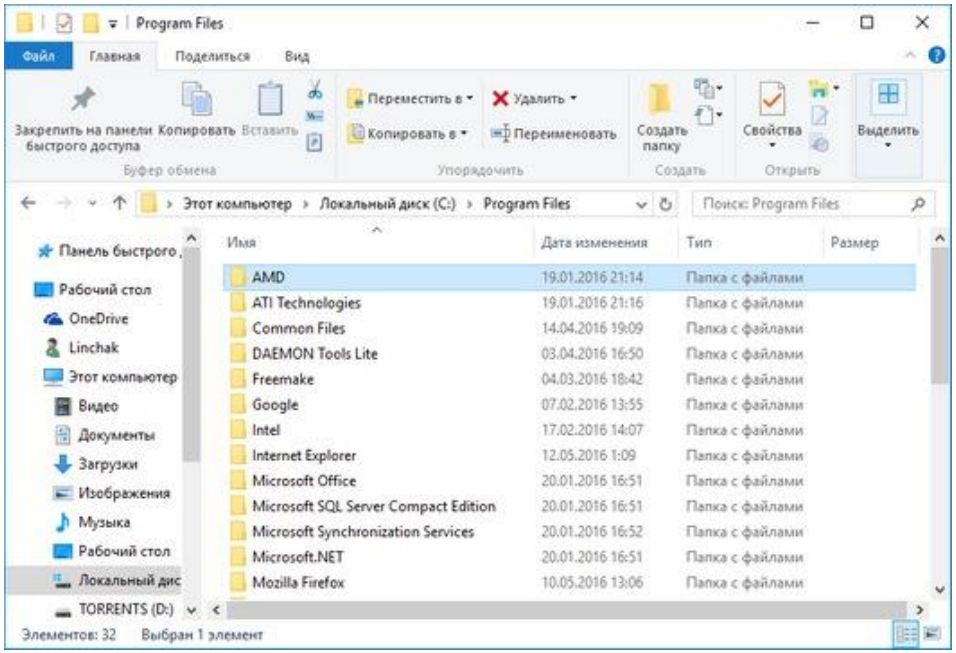
```
1   address,postal_code,country,federal_district,region_type,region,area_type,area,city_type,city,
    settlement_type,settlement,kladr_id,fias_id,fias_level,capital_marker,okato,oktmo,tax_office,
    timezone,geo_lat,geo_lon,population,foundation_year
  1 "Респ Адыгея, г Адыгейск",385200,Россия,Южный,Респ,Адыгея,,,г,Адыгейск,,,0100000200000,ccdfd496-8108-
    4655-aadd-bd228747306d,4,0,79403000000,79703000001,0107,UTC+3,44.878414,39.190289,12689,1969
  2 г Майкоп,385000,Россия,Южный,Респ,Адыгея,,,г,Майкоп,,,0100000100000,8cfbe842-e803-49ca-9347-
    1ef90481dd98,4,2,79401000000,79701000001,0105,UTC+3,44.6098268,40.1006606,144055,1857
  3 г Горно-Алтайск,649000,Россия,Сибирский,Респ,Алтай,,,г,Горно-Алтайск,,,0400000100000,0839d751-b940-
    4d3d-afb6-5df03fdd7791,4,2,84401000000,84701000,0400,UTC+7,51.9581028,85.9603235,62861,1830
  4 "Алтайский край, г Алейск",658125,Россия,Сибирский,край,Алтайский,,,г,Алейск,,,2200000200000,
    ae716080-f27b-40b6-a555-cf8b518e849e,4,0,01403000000,01703000,2201,UTC+7,52.4922513,82.7793606,28528,
    1913
  5 г Барнаул,656000,Россия,Сибирский,край,Алтайский,,,г,Барнаул,,,2200000100000,d13945a8-7017-46ab-b1e6-
    ede1e89317ad,4,2,01401000000,01701000,2200,UTC+7,53.3479968,83.7798064,635585,1730
  6 "Алтайский край, г Белокуриха",659900,Россия,Сибирский,край,Алтайский,,,г,Белокуриха,,,2200000300000,
    e4edca96-9b86-4cac-8c7f-cc93d9ba4cd1,4,0,01404000000,01704000001,2204,UTC+7,51.996152,84.9839604,
    15072,1846
  7 "Алтайский край, г Бийск",659300,Россия,Сибирский,край,Алтайский,,,г,Бийск,,,2200000400000,52f876f6-
    cb1d-4f23-a22f-b692609fc1e0,4,0,01405000000,01705000001,2204,UTC+7,52.5393864,85.2138453,203826,1709
  8 "Алтайский край, г Горняк",658420,Россия,Сибирский,край,Алтайский,р-н,Локтевский,г,Горняк,,,
    2202700100000,094b3627-2699-4782-8492-4d82aac71958,4,1,01225501000,01625101,2209,UTC+7,50.9979622,81.
    4643358,13040,1942
  9 "Алтайский край, г Заринск",659100,Россия,Сибирский,край,Алтайский,,,г,Заринск,,,2200001100000,
    142e04ef-dec1-44fa-b553-fac215764374,4,0,01406000000,01706000001,2208,UTC+7,53.7063476,84.9315081,
    47035,1748
 10 "Алтайский край, г Змеиногорск",658480,Россия,Сибирский,край,Алтайский,р-н,Змеиногорский,г,
    Змеиногорск,,,2201500100000,e7001b8f-d104-4873-96d4-66339f5e530a,4,1,01214501000,01614101,2209,UTC+7,
    51.1581094,82.1872547,10569,1736
 11 "Алтайский край, г Камень-на-Оби",658700,Россия,Сибирский,край,Алтайский,р-н,Каменский,г,Камень-на-
    Оби,,,2201800100000,810ca9c7-f10e-4def-9c48-f0aa83168ca7,4,1,01216501000,01616101001,2207,UTC+7,53@@@
NORMAL  us  ~/projects/mrsu-examples/db/02_SQL/dataset/city.csv          utf-8[unix]  0% ☰ 1/1118 ln :1
```

Практическое применение SQL

Инструменты для работы с наборами данных

Технология	Продукт	Достоинства	Недостатки
Электронные таблицы	Microsoft Excel	GUI, простота и наглядность. Хорошо подходит для простых разовых задач.	Тяжело работать со связанными данными. Данные не отделены от представления. Слабая повторяемость.
Скриптовый язык программирования	Python + Pandas (Jupyter Notebook)	Мощные возможности. Наглядная визуализация.	Необходимо изучать ЯП и специфические мето ды библиотек.
Язык SQL	SQLite	Декларативный язык специально для обработки данных. Простота установки.	Нет стандартных средств визуализации

GUI, CLI, скрипты и программы




```
C:\> copy d:\a.txt d:\b.txt
```

```
C:\python3.exe

>>> import shutil
>>> shutil.copyfile("d:\\a.txt", "d:\\b.txt")
```

Командный интерфейс для управления

- Файловой системой
- Объектами ОС
- Базами данных и структурированными данными
- Текстом