МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»

Факультет Управление и информатика в технологических системах
Кафедра Информационная безопасность
Специальность 10.05.03 «Информационная безопасность автоматизированных систем»

Применение SQL в приложениях

Выполнил студент гр. УБ-01
Цой Кирилл Вадимович

Воронеж – 2023

Использование представлений для скрытия столбцов и строк

С помощью представлений можно скрыть отдельные столбцы

таблиц. Это делается для того, чтобы возвращаемый результат имел

более простой вид, а также для предотвращения доступа к конфиденциальным данным. Следующий оператор создает представление CarsCollor, содержащее модель и цвет машины

CREATE VIEW CarsCollor AS

SELECT brand, color

FROM auto;

```
CarSellingShop=# CREATE VIEW CarsCollor AS
CarSellingShop-# SELECT brand, color
CarSellingShop-# FROM auto;
CREATE VIEW
CarSellingShop=#
```

```
CarSellingShop=# SELECT * FROM CarsCollor;
 brand | color
-------+-------
 M8    | black
 RIO   | gray
 RX-8  | red
(3 строки)


CarSellingShop=#
```

Использование представлений для отображения вычисляемых столбцов

Еще одно применение представлений – отображение результатов вычислений, не прибегая к вводу формул пользователем. Например, следующее представление объединит столбцы date и time и форматирует результат

CREATE VIEW ArrivalTime AS

SELECT ’ The car was brought in ’|| date || ’at’ || time || ‘o’clock’

FROM arrival_log;

```
CarSellingShop=# CREATE VIEW ArrivalTime AS
CarSellingShop-# SELECT 'The car was brought in '|| date || ' at '|| time || ' o’clock'
CarSellingShop-# FROM arrival_log;
CREATE VIEW
CarSellingShop=# SELECT * FROM ArrivalTime;
                      ?column?
-------------------------------------------------
 The car was brought in 2022-10-21 at 17:14:52 o’clock
 The car was brought in 2011-12-04 at 16:39:25 o’clock
 The car was brought in 2023-02-07 at 08:36:43 o’clock
(3 строки)


CarSellingShop=#
```

Использование представлений для скрытия сложного синтаксиса

Еще одно применение представлений – скрытие SQL-запросов со сложным синтаксисом. Это может делаться для того, чтобы избавить разработчиков от необходимости вводить сложный запрос. Рассмотрим следующее представление

CREATE VIEW CarMaker AS

SELECT a.brand AS Car, C.name AS Manufacture

FROM auto A

JOIN company C

ON C.id = A.id_company;

```
 car  | manufacture
------+-------------
 M8   | BMW
 RIO  | KIA
 RX-8 | Mazda
(3 строки)


CarSellingShop=#
```

Хранимые процедуры

CREATE OR REPLACE FUNCTION auto_insert(
newid_company IN INT,
newbrand IN char,
newcolor IN char,
newvin IN BIGINT,
newmileage IN BIGINT,
newyear IN INT,
newprice IN BIGINT)
RETURNS int AS $auto_insert$
DECLARE
autocursor CURSOR FOR
SELECT id

```
FROM auto
WHERE brand = newbrand;
rowcount int;
BEGIN
SELECT Count(*) INTO rowcount
FROM auto
WHERE id_company = newid_company
AND color = newcolor
AND mileage  = newmileage
AND year = newyear
AND vin  = newvin
AND price = newprice;
IF rowcount > 0 THEN
RAISE EXCEPTION 'There is client in DB! Count is %!',
rowcount;
END IF;
INSERT INTO auto(id_company, brand, color, vin, mileage,  year, price)
VALUES (newid_company, newbrand, newcolor, newvin, newmileage,  newyear,
newprice);
RAISE INFO 'Client is added!';

RETURN 1;

END;

$auto_insert$ LANGUAGE plpgsql;
```

```
CREATE FUNCTION
CarSellingShop=# SELECT auto_insert(
CarSellingShop(# '2', 'Soul', 'white', '1G8MG35X48Y106575','2502','2023','1500000');
ИНФОРМАЦИЯ:  Client is added!
 auto_insert
-------------
           1
(1 строка)
```

```
CarSellingShop=# SELECT * FROM auto;
 id | id_company | brand | color |        vin        | mileage | year |  price
----+------------+-------+-------+-------------------+---------+------+----------
  1 |          1 | M8    | black | 4A3AB76T68E011282 |   20000 | 2020 | 15672002
  2 |          2 | RIO   | gray  | 1GCHK23244F199207 |   20000 | 2010 |    50000
  3 |          3 | RX-8  | red   | JH4DA9440NS003801 |    5000 | 2022 |   500000
  4 |          2 | Soul  | white | 1G8MG35X48Y106575 |    2502 | 2023 |  1500000
(4 строки)
```

Если ввести уже существующую машину выдаст ошибку

```
CarSellingShop=# SELECT auto_insert('1', 'M8', 'black', '4A3AB76T68E011282', '20000', '2020', '15672002');
ОШИБКА:  There is client in DB! Count is 1!
```

Использование триггеров для проверки допустимости вводных данных

Добавим таблице auto атрибут recomendet_price
ALTER TABLE auto ADD COLUMN recomendet_price BIGINT
И чтобы обеспечивалось правило, что price всегда должен быть равен recomendet_price создадим функцию и триггер auto_price_check()

```
CREATE OR REPLACE FUNCTION auto_price_check()
RETURNS trigger AS $auto_price_check$
BEGIN
   IF NEW.recomendet_price > NEW.price THEN
      NEW.price = NEW.recomendet_price;
 END IF;
 RETURN NEW;
END;
$auto_price_check$ LANGUAGE plpgsql;

CREATE TRIGGER auto_price_check
BEFORE UPDATE ON auto
FOR EACH ROW EXECUTE PROCEDURE auto_price_check();
```

```
CREATE FUNCTION
CarSellingShop=# UPDATE auto SET price = 50000 WHERE id = 2;
UPDATE 1
CarSellingShop=# SELECT * FROM auto
CarSellingShop-# SELECT * FROM auto;
ОШИБКА:  ошибка синтаксиса (примерное положение: "SELECT")
СТРОКА 2: SELECT * FROM auto;
          ^
CarSellingShop=# SELECT * FROM auto;
 id | id_company | brand | color |        vin        | mileage | year |  price   | recomendet_price
----+------------+-------+-------+-------------------+---------+------+----------+------------------
  1 |          1 | M8    | black | 4A3AB76T68E011282 |   20000 | 2020 | 15672002 |         15672002
  3 |          3 | RX-8  | red   | JH4DA9440NS003801 |    5000 | 2022 |   500000 |           500000
  4 |          2 | Soul  | white | 1G8MG35X48Y106575 |    2502 | 2023 |  1500000 |          2000000
  2 |          2 | RIO   | gray  | 1GCHK23244F199207 |   20000 | 2010 |    50000 |            50000
(4 строки)


CarSellingShop=#
```

Использование триггеров для присвоения значений по умолчанию

Напишем функцию которая будет при добавлении новой машины будет устанавливать значением по умолчанию для “recomendet_price” значение из “price”

```
CREATE OR REPLACE FUNCTION set_recomendet_price()

RETURNS trigger AS $set_recomendet_price$

BEGIN

 NEW.recomendet_price = NEW.price;

 RETURN NEW;

END;

$set_recomendet_price$ LANGUAGE plpgsql;
```

CREATE TRIGGER set_recomendet_price_trigger

BEFORE INSERT ON auto

FOR EACH ROW EXECUTE PROCEDURE set_recomendet_price();

```
CarSellingShop=# CREATE OR REPLACE FUNCTION set_recomendet_price()
CarSellingShop-# RETURNS trigger AS $set_recomendet_price$
CarSellingShop$# BEGIN
CarSellingShop$#   NEW.recomendet_price = NEW.price;
CarSellingShop$#   RETURN NEW;
CarSellingShop$# END;
CarSellingShop$# $set_recomendet_price$ LANGUAGE plpgsql;
CREATE FUNCTION
CarSellingShop=#
CarSellingShop=# CREATE TRIGGER set_recomendet_price_trigger
CarSellingShop-# BEFORE INSERT ON auto
CarSellingShop-# FOR EACH ROW EXECUTE PROCEDURE set_recomendet_price();
CREATE TRIGGER
CarSellingShop=#
```

Теперь при добавлении машины процедурой auto_insert значение “recomendet_price” будет браться из “price”

```
CarSellingShop=# SELECT auto_insert(
CarSellingShop(# '3', 'CX-5', 'red', '2FTJW36M6LCA90573','36000','2023','3200000');
ИНФОРМАЦИЯ:  Client is added!
 auto_insert
-------------
           1
(1 строка)


CarSellingShop=# SELECT * FROM auto;
 id | id_company | brand | color |        vin        | mileage | year |  price   | recomendet_price
----+------------+-------+-------+-------------------+---------+------+----------+------------------
  1 |          1 | M8    | black | 4A3AB76T68E011282 |   20000 | 2020 | 15672002 |         15672002
  3 |          3 | RX-8  | red   | JH4DA9440NS003801 |    5000 | 2022 |   500000 |           500000
  4 |          2 | Soul  | white | 1G8MG35X48Y106575 |    2502 | 2023 |  1500000 |          2000000
  2 |          2 | RIO   | gray  | 1GCHK23244F199207 |   20000 | 2010 |    50000 |            50000
  5 |          3 | CX-5  | red   | 2FTJW36M6LCA90573 |   36000 | 2023 |  3200000 |          3200000
(5 строк)


CarSellingShop=#
```