



**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ**

**ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ**

БЕЗОПАСНОСТЬ СИСТЕМ БАЗ ДАННЫХ

Отчет

по практической работе 4

«Применение SQL в приложениях»

Отчет подготовил

студент 3 курса группы УБ-01

Хомутов Константин

Воронеж 2023

4.1 SQL-представления

SQL-представления (SQL view) – это виртуальная таблица, составленная из других таблиц или представлений и не имеет собственных данных, а объединяет данные из таблиц или представлений, которые в него входят. Представления создаются при помощи операторов CREATE VIEW и SELECT.

Следующий оператор определяет представление под названием CourseNameView, базирующийся на таблице course:

```
CREATE VIEW CourseNameView AS SELECT course_name FROM
course ORDER BY course_name;
```

Для получения отсортированного списка курсов, выведем представление с помощью:

```
SELECT * FROM CourseNameView;
```

```
courses2=# CREATE VIEW CourseNameView AS SELECT
CREATE VIEW
courses2=# SELECT * FROM CourseNameView;
      course_name
-----
администрирование сетей
операционные системы
программирование
устройство компьютера
(4 строки)
```

С помощью представлений можно скрыть отдельные столбцы для придания более простого вида или предотвращения доступа к конфиденциальным данным. Следующий оператор создает представление StudentDocs, содержащее только имя студента и его паспорт:

```
CREATE VIEW StudentDocs AS SELECT name, passport FROM student;
```

```
courses2=# CREATE VIEW StudentDocs AS SELECT name, passport FROM student;
CREATE VIEW
courses2=# SELECT * FROM StudentDocs;
      name                | passport
-----+-----
Смирнова Анна Серафимовна | 2022110483
Прохоров Тимур Эминович   | 2022592815
Лебедев Фёдор Дмитриевич  | 2022997893
Петров Платон Богданович  | 2022183922
Кузьмина Ксения Григорьевна | 2022420416
(5 строк)
```

Использование предложения WHERE позволяет скрыть строки, которые удовлетворяют определенному условию:

```
CRATE VIEW StudentNetAdm AS SELECT name, passport FROM
student WHERE course = 104;
```

```
courses2=# CREATE VIEW StudentNetAdm AS SELECT name, passport
CREATE VIEW
courses2=# SELECT * FROM StudentNetAdm;
      name      | passport
-----+-----
Петров Платон Богданович | 2022183922
Кузьмина Ксения Григорьевна | 2022420416
(2 строки)
```

Использование представлений для скрытия сложного синтаксиса позволяет избавить разработчиков от необходимости вводить сложный запрос всякий раз, когда им требуется определенное представление.

Так, мы можем узнать успеваемость всех студентов по всем предметам:

```
CREATE VIEW CourseStudentGrade AS SELECT C.course_name AS
subject, S.name AS student, grade AS grade FROM course C JOIN grade ON
C.course_id = grade.course_id JOIN student S ON S.student_id = grade.student_id;
```

После создания представления, предыдущий запрос сокращается до следующего:

```
Select * from CourseStudentGrade ORDER BY subject;
```

```
itco2=# Select * from CourseStudentGrade ORDER BY subject;
 subject      | student      | grade
-----+-----+-----
компьютерные сети | Мышкина Дария Кимовна | 100
компьютерные сети | Тетерин Климент Денисович | 65
операционные системы | Тетерин Климент Денисович | 85
операционные системы | Кудрявцев Альберт Миронович | 60
программирование | Николаев Терентий Вячеславович | 80
программирование | Мышкина Дария Кимовна | 93
программирование | Капустина Влада Мироновна | 90
устройство компьютера | Николаев Терентий Вячеславович | 90
устройство компьютера | Кудрявцев Альберт Миронович | 75
устройство компьютера | Капустина Влада Мироновна | 90
(10 строк)
```

4.2 Хранимые процедуры

SQL-запросы можно встраивать в триггеры, хранимые процедуры и прикладные программы. Для этого нужен какой-либо способ, позволяющий записывать результаты выполнения SQL-запросов в программные переменные. Например, подсчитать число строк в переменную rowcount можно сделать следующим запросом:

```
SELECT COUNT(*) INTO rowcount FROM course;
```

И узнать результат записи в эту переменную можно через оператор SELECT:

```
itco2=# SELECT * FROM rowcount;
 count
-----
      4
(1 строка)
```

Хранимая процедура (stored procedure) – это программа, которая выполняет некоторые действия с информацией в базе данных и при этом сама хранится в базе данных. Язык, на котором хранимые процедуры пишутся, - PL/pgSQL (Procedural Language/PostGres Structured Query Language – процедурное расширение языка SQL).

Пример следующей процедуры, которая выводит процент студентов, которые имеют определенную оценку:

```
create function grade_percent11 (param IN int)
    returns int
    language plpgsql
as
$$
declare
    gradecount numeric;
    mass numeric;
    resulter numeric;
begin
```

```

SELECT COUNT(*) INTO gradecount FROM grade WHERE grade =
param;

SELECT COUNT(*) INTO mass FROM grade;

resulter = gradecount/mass;

resulter = resulter*100;

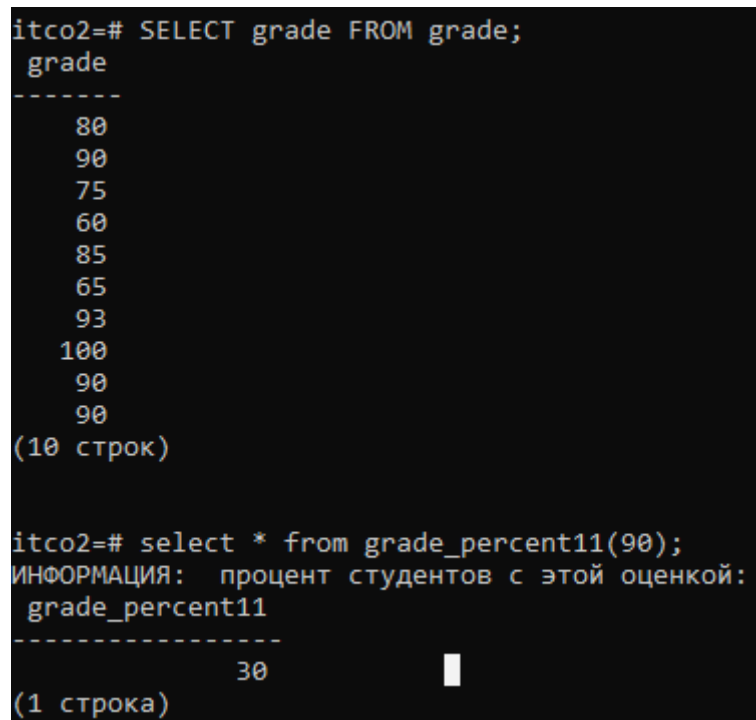
RAISE INFO 'процент студентов с этой оценкой:';

RETURN resulter;

end;

$$;

```



```

itco2=# SELECT grade FROM grade;
 grade
-----
    80
    90
    75
    60
    85
    65
    93
   100
    90
    90
(10 строк)

itco2=# select * from grade_percent11(90);
ИНФОРМАЦИЯ: процент студентов с этой оценкой:
 grade_percent11
-----
              30
(1 строка)

```

На скриншоте сначала показывается, что всего имеется 10 оценок, 3 из которых 90, значит процентное число этих оценок 30%.

4.3 Триггеры

Триггер (trigger) – это специальная программа, назначаемая таблице или представлению. Триггер вызывается СУБД, когда пользователь запрашивает вставку, обновление или удаление строки из таблицы или представления, которому принадлежит данный триггер. PostgreSQL

поддерживает три вида триггеров: предваряющие (BEFORE), замещающие (INSTEAD OF) и завершающие (AFTER).

Следующая триггерная функция выполняет проверку введенной электронной почты на содержание символа «@». Если этот символ есть, то функция вводит данные, если символа нет, то выводится сообщение о некорректности введенных данных. Для демонстрации примера этой функции была создана таблица email, которая содержит поля email_id и email_name.

```
CREATE FUNCTION email_check() RETURNS trigger AS
$email_check$
begin
    IF NEW.email_name ~ '@' THEN
        RAISE INFO 'ДАННЫЕ ЗАПИСАНЫ';
        RETURN NEW;
    ELSE
        RAISE INFO 'НЕКОРРЕКТНЫЙ EMAIL';
    END IF;
end;
$email_check$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER email_check AFTER INSERT OR UPDATE ON
email
```

```
FOR EACH ROW EXECUTE FUNCTION email_check();
```

```
itco2=#
itco2=# insert into email values
itco2=# (1001, 'nikolai2002oi.com');
ИНФОРМАЦИЯ: НЕКОРРЕКТНЫЙ EMAIL
ОШИБКА: конец триггерной процедуры достигнут без RETURN
КОНТЕКСТ: функция PL/pgSQL email_check()
itco2=#
itco2=#
itco2=#
itco2=# insert into email values
itco2=# (1002, 'kudmir@oi.com');
ИНФОРМАЦИЯ: ДАННЫЕ ЗАПИСАНЫ
INSERT 0 1
itco2=#
itco2=#
itco2=#
itco2=# SELECT * FROM email;
 email_id | email_name
-----+-----
      1002 | kudmir@oi.com
(1 строка)
```

В примере работы функции были попытки добавить две электронные почты `nikolai2002oi.com` и kudmir@oi.com. Функция не дает добавить первую почту, но позволяет добавить вторую.

4.4 Словарь метаданных

PostgreSQL поддерживает исчерпывающий словарь метаданных (`information_schema`), содержащий описание структур таблиц, последовательностей, представлений, индексов, ограничений, хранимых процедур и т. д. Он также содержит исходные тексты хранимых процедур и триггерных функций. Словарь метаданных можно использовать путем формирования запросов к СУБД.

Получение имени текущей базы данных:

```
SELECT * FROM information_schema.information_schema_catalog_name;
```

Получение списка ограничений:

```
SELECT * FROM information_schema.table_constraints;
```

Получение списка внешних ключей:

```
SELECT * FROM information_schema.referential_constraints;
```

Получение списка процедур:

```
SELECT * FROM information_schema.routines;
```

Получения списка последовательностей:

```
SELECT * FROM information_schema.sequences;
```

Получения списка таблиц:

```
SELECT * FROM information_schema.tables;
```

Получения списков триггеров:

```
SELECT * FROM information_schema.referential_triggers;
```

Получения списка представлений:

```
SELECT * FROM information_schema.views;
```