**Министерство науки и высшего образования Российской Федерации**
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

# ОТЧЕТ

# ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

## «Процедуры, функции, триггеры в PostgreSQL»

## по дисциплине «Проектирование и реализация баз данных»

**Обучающийся** Шестак Богдан Евгеньевич
**Факультет** прикладной информатики
**Группа** K3240
**Направление подготовки** 09.03.03 Прикладная информатика
**Образовательная программа** Мобильные и сетевые технологии 2025
**Преподаватель** Говорова Марина Михайловна

Санкт-Петербург

2025/2026

1. **Цель работы:**

Овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

**Практическое задание:**

1. Создать 3 процедуры для индивидуальной БД согласно варианту (часть 4 ЛР 2). Допустимо использование IN/OUT параметров. Допустимо создать авторские процедуры. (3 балла)
2. Создать триггеры для индивидуальной БД согласно варианту:

Вариант 2.1. 3 триггера - 3 балла (min). Допустимо использовать триггеры логирования из практического занятия по функциям и триггерам.

Вариант 2.2. 7 оригинальных триггеров - 7 баллов (max).

**Индивидуальное задание:**

**Вариант 7**. БД «Курсы»

Описание предметной области: Сеть учебных подразделений НОУ ДПО занимается организацией внебюджетного образования.

Имеется несколько образовательных программ краткосрочных курсов, предназначенных для определённых специальностей, связанных с программным обеспечением ИТ. Каждая программа имеет определённую длительность (в академических часах), свои перечни изучаемых дисциплин, вид итоговой аттестации, вид документа об окончании программы (сертификат о повышении квалификации, удостоверение о повышении квалификации, диплом о профессиональной подготовке).

Одна дисциплина может относиться к нескольким программам.

На каждую программу может быть набор из нескольких групп обучающихся.

По каждой дисциплине могут проводиться лекции, лабораторные/практические занятия и практика в определённом объёме часов.

По каждой дисциплине и практике проводится аттестация в форме экзаменов/защиты/зачёта.

Необходимо хранить информацию по аттестации обучающихся.

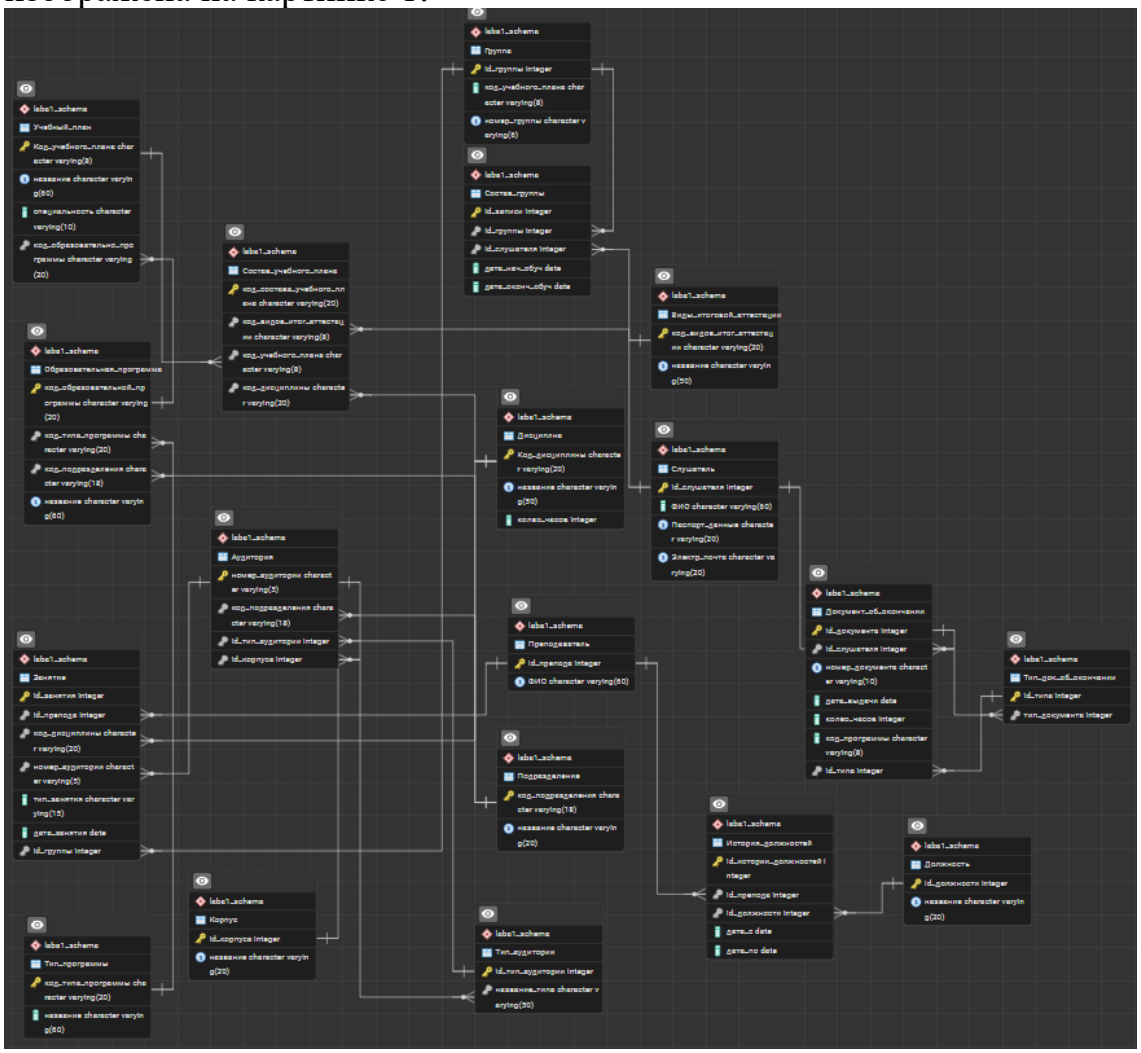
Подразделение обеспечивает следующие ресурсы: учебные классы, лекционные аудитории и преподаватели. Необходимы ресурсы для расписания занятий.

В системе необходимо хранить информацию о количестве и номере выданного документа об окончании программы и дату выдачи.

БД должна содержать следующий минимальный набор сведений: Фамилия слушателя. Имя слушателя. Паспортные данные. Контакты. Код программы. Программа. Тип программы. Образовательное учреждение. Номер группы. Максимальное количество человек в группе (для программы). Образовательный курс. Номер сертификата/удостоверения. Название дисциплины. Количество часов. Дата начала. Дата окончания. Номер удостоверения (при наличии). Фамилия преподавателя. Имя преподавателя. Должность преподавателя. Должность дисциплины.

Дополните состав атрибутов на основе анализа предметной области.

1. Наименование БД: laba1
2. Схема логической модели базы данных, сгенерированная в Generate ERD изображена на картинке 1:



Картинка 1 - Схема ИЛМ, сгенерированная в Generate ERD

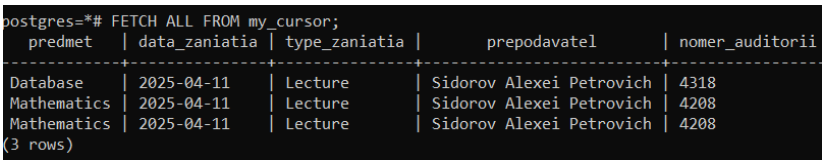
3. **Выполнение:**

**1) Хранимые процедуры**

**1)** Хранимая процедура для получения расписания занятий для групп на определенный день недели.

```
CREATE OR REPLACE PROCEDURE get_schedule_for_group_by_weekday(
    p_id_group INTEGER,
    p_weekday INTEGER,
    p_cursor_name INOUT refcursor
)
LANGUAGE plpgsql
AS $$
BEGIN
```

```
OPEN p_cursor_name FOR
   SELECT
      d.name_disciplini AS predmet,
      z.data_zaniatia,
      z.type_zaniatia,
      p.fio AS prepodavatel,
      z.nomer_auditorii
    FROM public."Zaniatie" z
    JOIN public."Prepodavatel" p ON z.id_prepoda = p.id_prepod
    JOIN public."Disciplina" d ON z.kod_disciplini = d.kod_disciplini
    WHERE z.id_group = p_id_group
     AND EXTRACT(DOW FROM z.data_zaniatia) = p_weekday
    ORDER BY z.data_zaniatia;
END;
$$;
```




Картинка 2 – Результат выполнения.

**2)** Хранимая процедура для записи на курс слушателя.

```
CREATE OR REPLACE PROCEDURE enroll_listener_on_course_by_number(
   p_nomer_group TEXT,
   p_id_slushatelia INTEGER,
   p_start_date DATE,
   p_end_date DATE
)
LANGUAGE plpgsql
AS $$
DECLARE
   v_id_group INTEGER;
BEGIN
   SELECT id_group INTO v_id_group
   FROM public."Group"
   WHERE nomer_group = p_nomer_group;

   IF v_id_group IS NULL THEN
      RAISE EXCEPTION 'Group with number % not found', p_nomer_group;
   END IF;
```

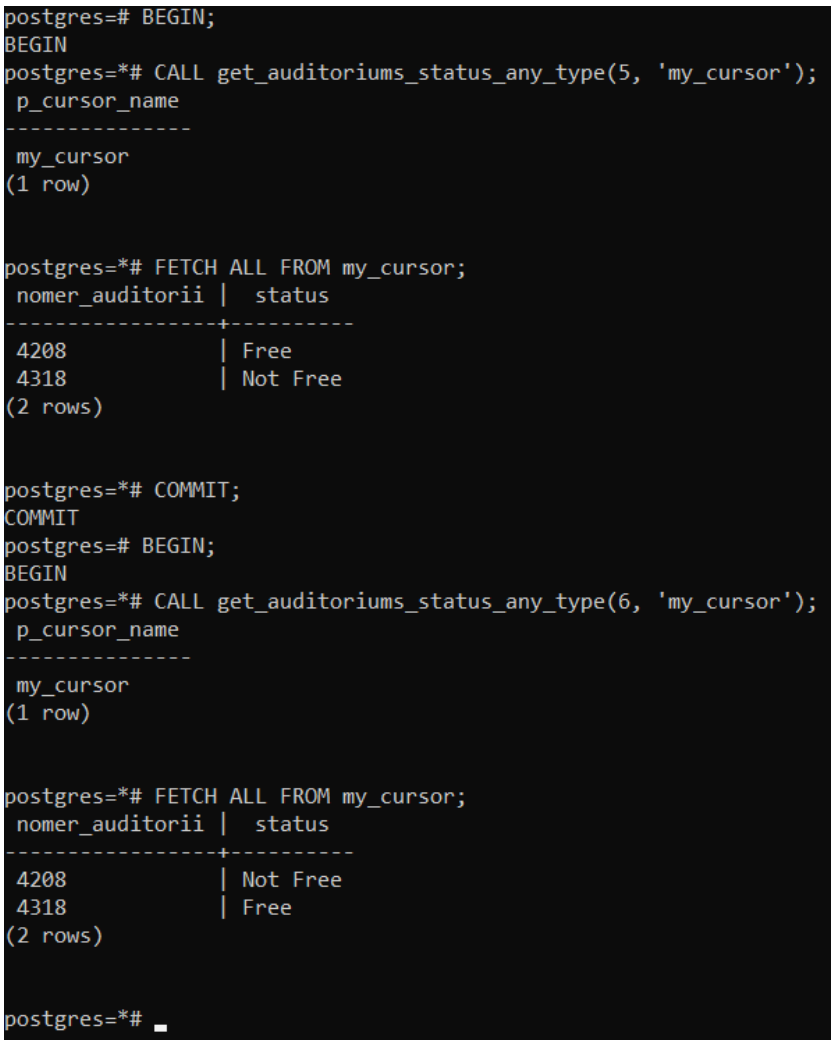
```
    INSERT INTO public."Sostav_group" (id_group, id_slushatelia,
data_start_obuch, data_finish_obuch)
    VALUES (v_id_group, p_id_slushatelia, p_start_date, p_end_date);

    RAISE NOTICE 'Slushatel s ID % bil zapisan v group s nomerom % na cours s
date nachala % i date okonchania %',
                p_id_slushatelia, p_nomer_group, p_start_date, p_end_date;
END;
$$;
```

```
postgres=# CALL enroll_listener_on_course_by_number('02', 2, '2025-06-01', '2025-12-01');
NOTICE:  Slushatel s ID 2 bil zapisan v group s nomerom 02 na cours s date nachala 2025-06-01 i date okonchania 2025-12-
01
```

Картинка 3 – Результат выполнения.

**3)** Хранимая процедура получения перечня свободных лекционных аудиторий на любой день недели.

```
CREATE OR REPLACE PROCEDURE
get_auditoriums_status_any_type(
        p_weekday INTEGER,
        INOUT p_cursor_name refcursor
)
LANGUAGE plpgsql
AS $$
BEGIN
        OPEN p_cursor_name FOR
        SELECT
                a.nomer_auditorii,
                CASE
                        WHEN z.nomer_auditorii IS NULL THEN 'Free'
                        ELSE 'Not Free'
                END AS status
        FROM public."Auditoria" a
        LEFT JOIN (
        SELECT DISTINCT nomer_auditorii
        FROM public."Zaniatie"
        WHERE EXTRACT(DOW FROM data_zaniatia) =
    p_weekday
                AND nomer_auditorii IS NOT NULL
        ) z ON a.nomer_auditorii = z.nomer_auditorii
        ORDER BY a.nomer_auditorii;
END;
$$;
```




Картинка 4 – Результат выполнения.

## 2) Триггеры

**1)** Триггер на логирование изменений ФИО слушателя.

```
CREATE TABLE log_slushatel (
        log_id SERIAL PRIMARY KEY,
        action_time TIMESTAMP DEFAULT now(),
        action_type VARCHAR(10),
        id_slushatelia INTEGER,
        fio VARCHAR(100));

CREATE OR REPLACE FUNCTION log_slushatel_update()
RETURNS TRIGGER AS $$
BEGIN
   INSERT INTO log_slushatel(action_type, id_slushatelia, fio)
   VALUES ('UPDATE', NEW.id_slushatelia, NEW.fio);
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
UPDATE public."Slushatel"SET fio = 'Romanenko Gleb
Borisovich'WHERE id_slushatelia = 1;
SELECT * FROM log_slushatel ORDER BY log_id DESC LIMIT 5;
```

```
postgres=# UPDATE public."Slushatel"SET fio = 'Romanenko Gleb Borisovich'WHERE id_slushatelia = 1;
UPDATE 1
postgres=# SELECT * FROM log_slushatel ORDER BY log_id DESC LIMIT 5;
 log_id |        action_time         | action_type | id_slushatelia |           fio
--------+----------------------------+-------------+----------------+--------------------------
      1 | 2025-05-21 22:21:43.395627 | UPDATE      |              1 | Romanenko Gleb Borisovich
(1 row)
```

Картиника 5 – Результат выполнения.

**2)** Триггер проверки даты окончания обучения

```
CREATE OR REPLACE FUNCTION check_study_dates()
RETURNS TRIGGER AS $$BEGIN
        IF NEW.data_finish_obuch < NEW.data_start_obuch THEN
            RAISE EXCEPTION 'Data okonchania obuchenia (%), ne
mozhet bit ranshe date nachala obuchenia (%)',
                NEW.data_finish_obuch, NEW.data_start_obuch;
        END IF;
        RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_study_dates
BEFORE INSERT OR UPDATE ON public."Sostav_group"
FOR EACH ROW
EXECUTE FUNCTION check_study_dates();
```

```
postgres=# INSERT INTO public."Sostav_group" (id_group, id_slushatelia, data_start_obuch, data_finish_obuch)VAL
UES (1, 1, '2025-06-01', '2025-05-01');
ERROR:  Data okonchania obuchenia (2025-05-01), ne mozhet bit ranshe date nachala obuchenia (2025-06-01)
CONTEXT:  PL/pgSQL function check_study_dates() line 1 at RAISE
postgres=#
```

Картиника 6 – Результат выполнения.

**3)** Триггер на подсчет количества слушателей в группе.

```
CREATE OR REPLACE FUNCTION update_group_status()
RETURNS TRIGGER AS $$
DECLARE
   grp_id INTEGER;
   cnt INTEGER;
BEGIN
   IF (TG_OP = 'DELETE') THEN
      grp_id := OLD.id_group;
   ELSE
      grp_id := NEW.id_group;
   END IF;
```

```
    SELECT COUNT(*) INTO cnt FROM public."Sostav_group" WHERE
id_group = grp_id;

    IF cnt = 0 THEN
        UPDATE public."Group" SET status = 'Closed' WHERE id_group =
grp_id;
    ELSE
        UPDATE public."Group" SET status = 'Active' WHERE id_group =
grp_id;
    END IF;

    IF (TG_OP = 'DELETE') THEN
        RETURN OLD;
    ELSE
        RETURN NEW;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Триггер сработает после вставки, обновления и удаления записи в Sostav_group.

```
CREATE TRIGGER trg_update_group_status_insert_update
AFTER INSERT OR UPDATE ON public."Sostav_group"
FOR EACH ROW
EXECUTE FUNCTION update_group_status();

CREATE TRIGGER trg_update_group_status_delete
AFTER DELETE ON public."Sostav_group"
FOR EACH ROW
EXECUTE FUNCTION update_group_status();
```

```
postgres=# SELECT id_group, status FROM public."Group" WHERE id_group = 1;
 id_group | status
----------+--------
        1 | Active
(1 row)
```

Картиника 7 – Состав группы активен.

А теперь попробуем удалить запись из Sostav_group

```
postgres=# DELETE FROM public."Sostav_group" WHERE id_group = 1;
DELETE 1
Asynchronous notification "debug" with payload "Group ID: 1" received from server process with PID 17908.
Asynchronous notification "debug" with payload "Count of members: 0" received from server process with PID 1790
8.
postgres=# SELECT id_group, status FROM public."Group" WHERE id_group = 1;
 id_group | status
----------+--------
        1 | Closed
(1 row)
```

Картиника 8 – Состав группы закрыт.

**4)** Триггер ограничения на количество занятий в один день для преподавателя

```
CREATE OR REPLACE FUNCTION check_teacher_daily_limit()
RETURNS TRIGGER AS $$
DECLARE
   lesson_count INTEGER;
   max_lessons CONSTANT INTEGER := 3;
BEGIN
  SELECT COUNT(*) INTO lesson_count
  FROM public."Zaniatie"
  WHERE id_prepoda = NEW.id_prepoda
     AND data_zaniatia = NEW.data_zaniatia
     AND (TG_OP = 'INSERT' OR id_zaniatia <> NEW.id_zaniatia);

  IF lesson_count >= max_lessons THEN
      RAISE EXCEPTION 'Previshen limit iz % zaniatii v dzen dlia
prepodavatelia ID %', max_lessons, NEW.id_prepoda;
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER trg_check_teacher_daily_limit
BEFORE INSERT OR UPDATE ON public."Zaniatie"
FOR EACH ROW
EXECUTE FUNCTION check_teacher_daily_limit();
```

```
postgres=# SELECT id_zaniatia, id_prepoda, data_zaniatia FROM public."Zaniatie" WHERE id_prepoda = 1  AND data_
zaniatia = '2025-04-11';
 id_zaniatia | id_prepoda | data_zaniatia
-------------+------------+---------------
           1 |          1 | 2025-04-11
           4 |          1 | 2025-04-11
           3 |          1 | 2025-04-11
(3 rows)
```

Картиника 9 – Количество занятий у преподавателя 1.

```
postgres=# INSERT INTO public."Zaniatie" (id_zaniatia, id_prepoda, kod_disciplini, nomer_auditorii, type_zaniat
ia, data_zaniatia, id_group)VALUES (5, 1, 'D001', '4208', 'Lecture', '2025-04-11', 1);
ERROR:  Previshen limit iz 3 zaniatii v dzen dlia prepodavatelia ID 1
CONTEXT:  PL/pgSQL function check_teacher_daily_limit() line 1 at RAISE
postgres=#
```

Картиника 10 – Результат выполнения.

**5)** Триггер автоматического заполнения email слушателя (если пустой)

```
CREATE OR REPLACE FUNCTION fill_default_email()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.email IS NULL OR trim(NEW.email) = '' THEN
        NEW.email := 'unknown@example.com';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_fill_default_email
BEFORE INSERT ON public."Slushatel"
FOR EACH ROW
EXECUTE FUNCTION fill_default_email();

INSERT INTO public."Slushatel" (id_slushatelia, fio, passport_danie,
email)VALUES (4, 'Kisigach Anastasia Konstantinovna', '4321 098765', NULL);
```

| | id_slushatelia [PK] integer | fio character varying (60) | passport_danie character varying (20) | email character varying (20) |
|---|---|---|---|---|
| 1 | 2 | Petrov Petr Petrovich | 2345 678901 | petrov@mail.ru |
| 2 | 3 | Lobanov Semen Semenovich | 3456 789012 | lobanov@mail.ru |
| 3 | 1 | Romanenko Gleb Borisovich | 1234 567890 | ivanov@mail.ru |
| 4 | 4 | Kisigach Anastasia Konstantinovna | 4321 098765 | unknown@example.com |

Картиника 11 – Результат выполнения.

**6)** Триггер, проверющий количество часов

```
CREATE OR REPLACE FUNCTION check_kolvo_chasov()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.kolvo_chasov < 20 THEN
        RAISE EXCEPTION 'Kolichestvo chasov dolzhno bit ne menshe
20. Vashe znachenie: %', NEW.kolvo_chasov;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trg_check_kolvo_chasov
BEFORE INSERT OR UPDATE ON public."Disciplina"
FOR EACH ROW
EXECUTE FUNCTION check_kolvo_chasov();
```

```
postgres=# INSERT INTO public."Disciplina" (kod_disciplini, name_disciplini, kolvo_chasov)VALUES ('D004', 'PE',
 15);
ERROR:  Kolichestvo chasov dolzhno bit ne menshe 20. Vashe znachenie: 15
CONTEXT:  PL/pgSQL function check_kolvo_chasov() line 1 at RAISE
postgres=#
```

Картиника 12 – Результат выполнения.

**7)** Триггер автоматического обновления количества часов в дисциплине при изменении связанной учебной программы

При содержании в названии дисциплины "Profile", автоматически прибовлялось количество часов на 10.

```
CREATE OR REPLACE FUNCTION update_hours_based_on_name()
RETURNS TRIGGER AS $$
BEGIN
   IF NEW.name_disciplini ILIKE '% Profile%' THEN
      NEW.kolvo_chasov := NEW.kolvo_chasov + 10;
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_update_hours_based_on_name
BEFORE INSERT OR UPDATE ON public."Disciplina"
FOR EACH ROW
EXECUTE FUNCTION update_hours_based_on_name();

INSERT INTO public."Disciplina" (kod_disciplini, name_disciplini,
kolvo_chasov)VALUES ('D005', 'Profile Mathematics', 60);
```

| | kod_disciplini [PK] character varying (20) | name_disciplini character varying (50) | kolvo_chasov integer |
|---|---|---|---|
| 1 | D001 | Mathematics | 60 |
| 2 | D002 | Physics | 50 |
| 3 | D003 | Database | 70 |
| 4 | D005 | Profile Mathematics | 70 |

Картиника 13 – Результат выполнения.

**Вывод**

В ходе выполнения данной лабораторной работы были изучены и практически применены хранимые процедуры, функции и триггеры. Самостоятельная разработка и тестирование этих компонентов на базе реализованной базы данных позволили глубже понять их работу и приобрести навыки создания качественных систем управления базами данных.