МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

ЛАБОРАТОРНАЯ РАБОТА № 5

по дисциплине

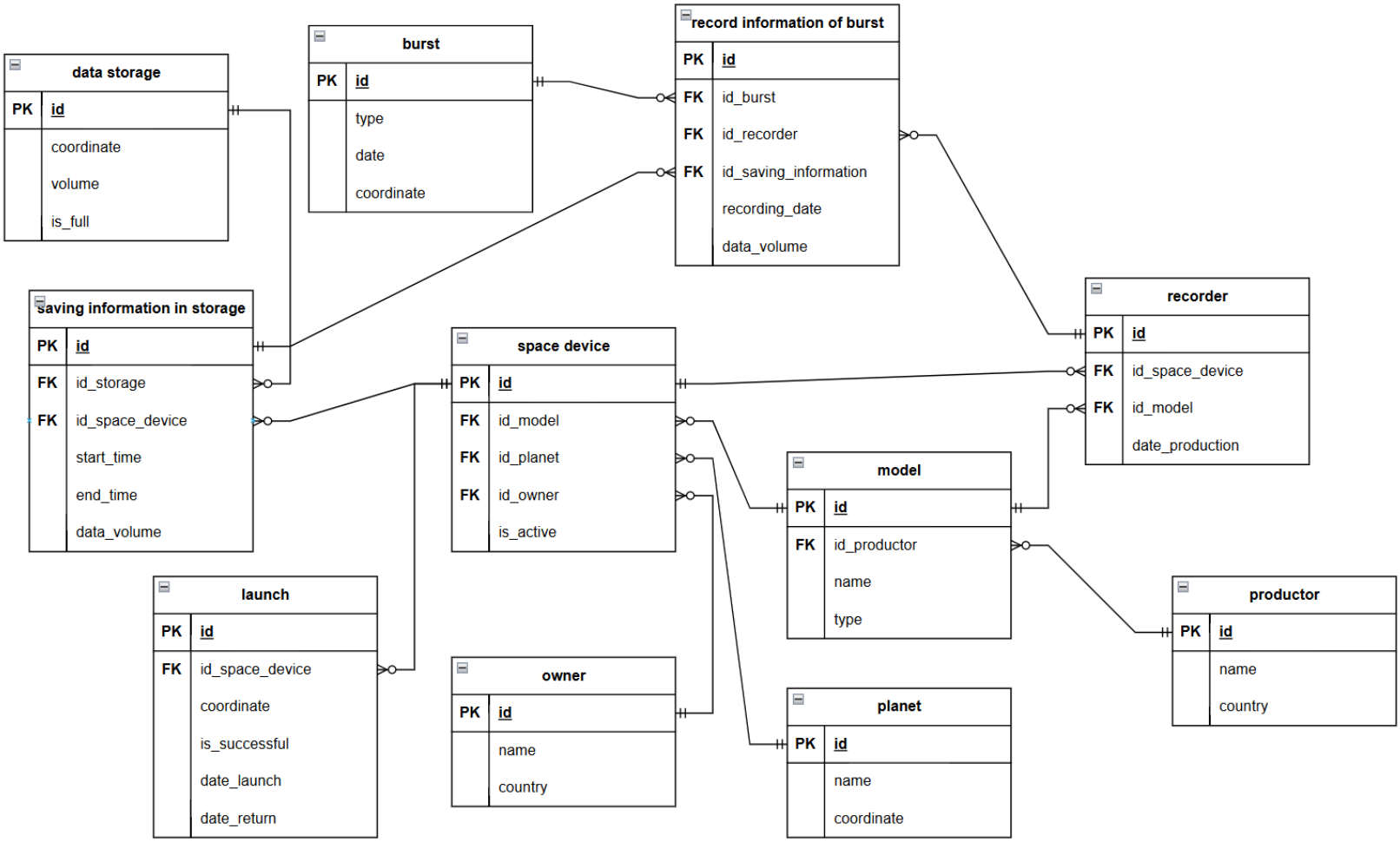
“ Проектирование и реализация баз данных”

Выполнил:
Мохаджер Алиреза
Джафари Хоссаин

Студент группы K3240

Преподаватель:
Белов Александр Олегович
Говорова Марина Михайловна

Санкт-Петербург, 2025



Первый триггер: Автообновление is_full в data_storage при добавлении информации

Таблица saving_information_in_storage:

	id [PK] integer	id_storage integer	id_space_device integer	start_time timestamp without time zone	end_time timestamp without time zone	data_volume integer
1	1	1	2	2022-01-01 12:01:00	2022-01-01 12:05:00	100
2	2	2	3	2022-02-01 12:01:00	2022-02-01 12:05:00	200
3	3	3	1	2022-03-01 12:01:00	2022-03-01 12:05:00	150
4	4	4	4	2022-04-01 12:01:00	2022-04-01 12:05:00	180

Таблица data_storage:

	id [PK] integer	coordinate character varying (255)	volume integer	is_full boolean
1	1	SOL-3-001	1000	false
2	2	SOL-3-002	2000	true
3	3	SOL-4-001	500	false
4	4	SOL-3-003	1500	false

Триггер:

```
-- Автообновление is_full в data_storage при добавлении информации
CREATE OR REPLACE FUNCTION check_storage_fullness()
RETURNS TRIGGER AS $$
BEGIN
  -- Сумма всего объема данных в этом хранилище
  IF (
    SELECT COALESCE(SUM(data_volume), 0)
    FROM saving_information_in_storage
    WHERE id_storage = NEW.id_storage
  ) >= (
    SELECT volume
    FROM data_storage
    WHERE id = NEW.id_storage
  ) THEN
    UPDATE data_storage
    SET is_full = TRUE
    WHERE id = NEW.id_storage;
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_storage_fullness
AFTER INSERT ON saving_information_in_storage
FOR EACH ROW
EXECUTE FUNCTION check_storage_fullness();
```

INSERT INTO saving_information_in_storage (id_storage, id_space_device, start_time, end_time, data_volume) VALUES (1, 2, '2024-01-25 12:10:00', '2024-01-25 12:15:00', 900);

Data Output | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 35 msec.
```

Таблица data_storage:

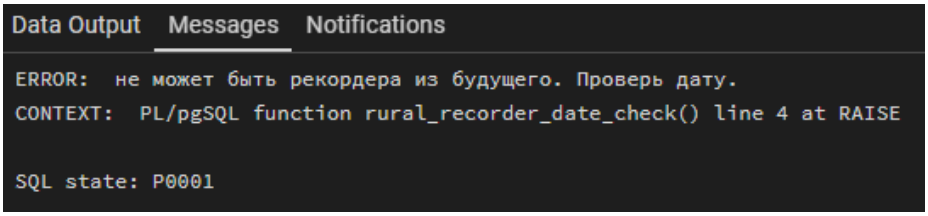
	id [PK] integer	coordinate character varying (255)	volume integer	is_full boolean
1	2	SOL-3-002	2000	true
2	3	SOL-4-001	500	false
3	4	SOL-3-003	1500	false
4	1	SOL-3-001	1000	true

Второй триггер: не дать произвести рекордер в будущем

```
CREATE OR REPLACE FUNCTION rural_recorder_date_check()
RETURNS TRIGGER AS $$
BEGIN
  IF NEW.date_production > CURRENT_DATE THEN
    RAISE EXCEPTION 'не может быть рекордера из будущего. Проверь дату.';
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_rural_recorder_date
BEFORE INSERT ON recorder
FOR EACH ROW
EXECUTE FUNCTION rural_recorder_date_check();
```

INSERT INTO recorder (id_space_device, id_model, date_production) VALUES (2, 2, '2026-01-01');



Третий триггер: автоматически устанавливать флаг is_active = TRUE, когда у космического аппарата (space_device) происходит успешный запуск (launch.is_successful = TRUE).

Таблица space_device:

	id [PK] integer	id_model integer	id_planet integer	id_owner integer	is_active boolean
1	1	1	1	1	true
2	2	2	[null]	1	true
3	3	3	[null]	2	true
4	4	4	3	3	false

Таблица launch:

	id [PK] integer	id_space_device integer	coordinate character varying (255)	is_successful boolean	date_launch date	date_return date
1	1	1	28.5618N-80.5774W	true	2020-01-01	[null]
2	2	2	28.5618N-80.5774W	true	1977-09-05	[null]
3	3	3	5.2397N-52.7688W	true	1990-04-24	[null]
4	4	4	45.9650N-63.3050E	true	1970-11-10	1970-11-17

Триггер:

```
CREATE OR REPLACE FUNCTION activate_device_on_successful_launch()
RETURNS TRIGGER AS $$
BEGIN
  IF NEW.is_successful THEN
    UPDATE space_device
    SET is_active = TRUE
    WHERE id = NEW.id_space_device;
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_activate_device_after_launch
AFTER INSERT ON launch
FOR EACH ROW
EXECUTE FUNCTION activate_device_on_successful_launch();
```

INSERT INTO space_device (id_model, id_planet, id_owner, is_active) VALUES

(1, 1, 1, FALSE);

INSERT INTO launch (id_space_device, coordinate, is_successful, date_launch)

VALUES (5, 'X123-Z9', TRUE, '2025-05-29');

Таблица space_device:

	id [PK] integer	id_model integer	id_planet integer	id_owner integer	is_active boolean
1	1	1	1	1	true
2	2	2	[null]	1	true
3	3	3	[null]	2	true
4	4	4	3	3	false
5	5	1	1	1	true

Первая процедура: Добавить вспышку и сразу записать её, если есть активные устройства

Таблица burst:

	id [PK] integer	type character varying (100)	date timestamp without time zone	coordinate character varying (255)
1	1	gamma	2022-01-01 12:00:00	RA14h20m
2	2	x-ray	2022-02-01 12:00:00	RA18h45m
3	3	radio	2022-03-01 12:00:00	RA22h10m
4	4	optical	2022-04-01 12:00:00	RA5h30m

Таблица record_information_of_burst:

	id [PK] integer	id_burst integer	id_recorder integer	id_saving_information integer	recording_date timestamp without time zone	data_volume integer
1	1	1	1	1	2022-01-01 12:01:30	50
2	2	2	2	2	2022-02-01 12:01:30	80
3	3	3	3	3	2022-03-01 12:01:30	60
4	4	4	4	4	2022-04-01 12:01:30	70

Процедура:

```
CREATE OR REPLACE PROCEDURE add_burst_and_record(
  burst_type VARCHAR,
  burst_date TIMESTAMP,
  burst_coord VARCHAR,
  saving_id INT,
  recorder_id INT,
  data_vol INT
)
LANGUAGE plpgsql
AS $$
DECLARE
  burst_id INT;
BEGIN
  -- Вставим вспышку
  INSERT INTO burst (type, date, coordinate)
  VALUES (burst_type, burst_date, burst_coord)
  RETURNING id INTO burst_id;

  -- Добавим запись о регистрации вспышки
  INSERT INTO record_information_of_burst (
    id_burst, id_recorder, id_saving_information, recording_date, data_volume
  )
  VALUES (burst_id, recorder_id, saving_id, CURRENT_TIMESTAMP, data_vol);
END;
$$;
```

CALL add_burst_and_record('gamma', '2025-05-29 12:00:00', 'GAMMA-01-ZX', 4, 3, 512);

Таблица record_information_of_burst:

	id [PK] integer	id_burst integer	id_recorder integer	id_saving_information integer	recording_date timestamp without time zone	data_volume integer
1	1	1	1	1	2022-01-01 12:01:30	50
2	2	2	2	2	2022-02-01 12:01:30	80
3	3	3	3	3	2022-03-01 12:01:30	60
4	4	4	4	4	2022-04-01 12:01:30	70
5	6	6	3	4	2025-05-29 12:12:55.568473	512

Вторая процедура: Проверяет загруженность конкретного хранилища по ID и выводит его текущий объём использования.

```
CREATE OR REPLACE PROCEDURE check_storage_usage(p_storage_id INT)
LANGUAGE plpgsql
AS $$
DECLARE
  total_used INT;
  max_volume INT;
  free_space INT;
  is_full_now BOOLEAN;
BEGIN

  IF NOT EXISTS (SELECT 1 FROM data_storage WHERE id = p_storage_id) THEN
    RAISE EXCEPTION 'Хранилище с id = % не найдено.', p_storage_id;
  END IF;


  SELECT volume, is_full INTO max_volume, is_full_now
  FROM data_storage
  WHERE id = p_storage_id;

  SELECT COALESCE(SUM(data_volume), 0) INTO total_used
  FROM saving_information_in_storage
  WHERE id_storage = p_storage_id;

  free_space := max_volume - total_used;

  RAISE NOTICE 'Хранилище:';
  RAISE NOTICE '- Общий объём: % единиц', max_volume;
  RAISE NOTICE '- Использовано: % единиц', total_used;
  RAISE NOTICE '- Свободно: % единиц', free_space;
  RAISE NOTICE '- Заполнено полностью? %', is_full_now;
END;
$$;
```

Таблица data_storage:

	id [PK] integer	coordinate character varying (255)	volume integer	is_full boolean
1	2	SOL-3-002	2000	true
2	3	SOL-4-001	500	false
3	4	SOL-3-003	1500	false
4	1	SOL-3-001	1000	true

CALL check_storage_usage(2);

Data Output | Messages | Notifications

```
NOTICE:  Хранилище:
NOTICE:  - Общий объём: 2000 единиц
NOTICE:  - Использовано: 200 единиц
NOTICE:  - Свободно: 1800 единиц
NOTICE:  - Заполнено полностью? t
CALL

Query returned successfully in 58 msec.
```

третья процедура: переназначает владельца устройств, зная имя и страну старого владельца и имя и страну нового владельца.

```
CREATE OR REPLACE PROCEDURE reassign_owner(
  old_owner_name TEXT,
  old_owner_country TEXT,
  new_owner_name TEXT,
  new_owner_country TEXT
)
LANGUAGE plpgsql
AS $$
DECLARE
  old_owner_id INT;
  new_owner_id INT;
BEGIN

  SELECT id INTO old_owner_id
  FROM owner
  WHERE name = old_owner_name AND country = old_owner_country;

  IF NOT FOUND THEN
    RAISE EXCEPTION 'Старый владелец с именем "%" и страной "%" не найден.',
      old_owner_name, old_owner_country;
  END IF;


  SELECT id INTO new_owner_id
  FROM owner
  WHERE name = new_owner_name AND country = new_owner_country;

  IF NOT FOUND THEN
    RAISE EXCEPTION 'Новый владелец с именем "%" и страной "%" не найден.',
      new_owner_name, new_owner_country;
  END IF;


  UPDATE space_device
  SET id_owner = new_owner_id
  WHERE id_owner = old_owner_id;

  RAISE NOTICE 'Все устройства были переданы от % (%s) к % (%s).',
    old_owner_name, old_owner_country, new_owner_name, new_owner_country;
END;
$$;
```

select space_device.id,id_owner, owner.name, owner.country from space_device
INNER JOIN owner
on owner.id = space_device.id_owner;

	id integer	id_owner integer	name character varying (255)	country character varying (255)
1	1	1	US Gov	USA
2	2	1	US Gov	USA
3	5	1	US Gov	USA
4	4	2	EU Space	Europe
5	3	2	EU Space	Europe

```
CALL reassign_owner(
  'EU Space', 'Europe',
  'Russian Fed', 'Russia'
);
```

Data Output | Messages | Notifications

```
NOTICE:  Все устройства были переданы от EU Space (Europes) к Russian Fed (Russias).
CALL

Query returned successfully in 38 msec.
```

```
select space_device.id,id_owner, owner.name, owner.country from space_device
INNER JOIN owner
on owner.id = space_device.id_owner;
```

	id integer	id_owner integer	name character varying (255)	country character varying (255)
1	1	1	US Gov	USA
2	2	1	US Gov	USA
3	5	1	US Gov	USA
4	4	3	Russian Fed	Russia
5	3	3	Russian Fed	Russia