Федеральное государственное автономное образовательное учреждение высшего образования  
«Национальный исследовательский университет ИТМО»  
  
Факультет программной инженерии и компьютерной техники  
  
Направление подготовки 09.03.04 «Программная инженерия» – Системное и прикладное программное обеспечение

**Отчёт  
По лабораторной работе №3**

По дисциплине «Базы данных»  
**Вариант: 68643**

Выполнил:  
Студент 1 курса  
Васильев Артём Евгеньевич

Группа: 3119

Преподаватель:

Райла Мартин

2024 г.

# Текст задания:

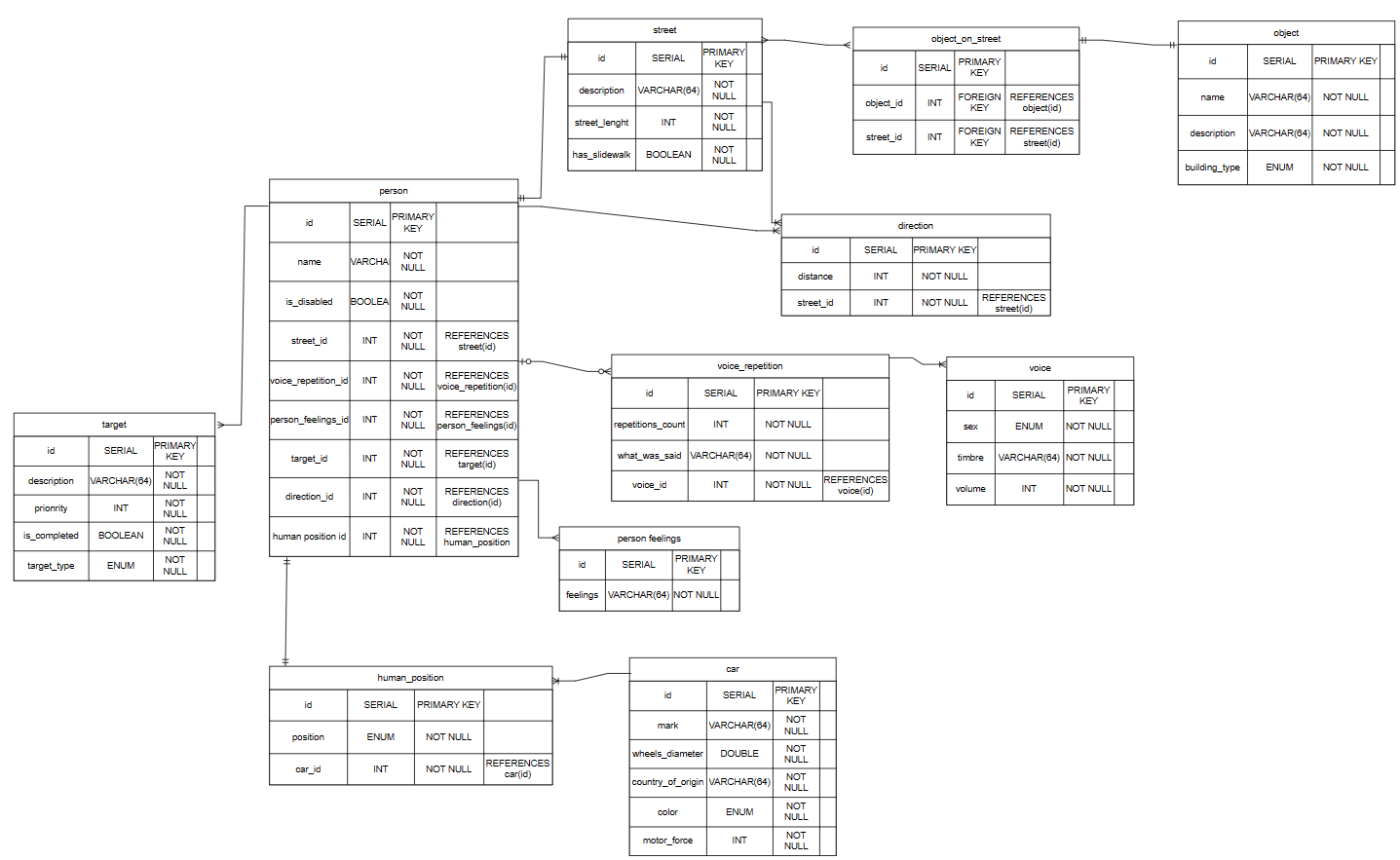
Для отношений, полученных при построении предметной области из лабораторной работы №1, выполните следующие действия:

* Опишите функциональные зависимости для отношений полученной схемы (минимальное множество);
* Приведите отношения в 3NF (как минимум). Постройте схему на основеNF (как минимум).
* Опишите изменения в функциональных зависимостях, произошедшие после преобразования в 3NF (как минимум). Постройте схему на основеNF;
* Преобразуйте отношения в BCNF. Докажите, что полученные отношения представлены в BCNF. Если ваша схема находится уже в BCNF, докажите это;
* Какие денормализации будут полезны для вашей схемы? Приведите подробное описание.

Придумайте триггер и связанную с ним функцию, относящиеся к вашей предметной области, согласуйте их с преподавателем и реализуйте на языке PL/pgSQL.

# Основные этапы работы

***Исходная даталогическая модель (из лабораторной работы 1)***



***Функциональные зависимости***

person:

* id -> (name, is\_disabled)

target:

* id -> (description, priority, is\_completed, target\_type)

human\_position:

* id -> position

car:

* mark -> country\_of\_origin
* id -> (mark, wheels\_diameter, color, motor\_force, country\_of\_origin)

person\_feeligs:

* id -> feelings

voice\_repetition:

* id -> (repetitions\_count, what\_was\_said)

voice:

* id -> (sex, timbre, volume)

direction:

* id -> distance

street:

* id -> (description, street\_lenght, has\_slidewalk)

object:

* id -> (name, description, building\_type)

***Условия полноты***

База данных содержит все необходимые таблицы, поля и данные для работы и обеспечивает полноценную функциональность. База данных удовлетворяет условиям полноты.

***Нормальные формы***

* ***1NF***: Отношение находится в 1NF, если все его атрибуты содержат только атомарные значения. В моём случае все атрибуты атомарны, нет повторяющихся групп, потому моя модель удовлетворяет 1NF
* ***2NF***: Отношение находится во 2NF, если оно находится в 1NF и все его неключевые атрибуты полностью функционально зависят от первичных ключей. В моём случае все неключевые аттрибуты полностью функционально зависят от первичных ключей, потому моя модель удовлетворяет 2NF (док-во полноты. )

*По таблицам:*

Поля name и is\_disabled функционально зависят от ключа id, поскольку, зная id человека, мы можем однозначно определить его имя и наличие инвалидности.

Поля description, priority, is\_completed, target\_type зависят от ключа id, поскольку, зная id цели, мы можем однозначно определить её описание, приоритет, состояние и тип.

Поля mark, wheels\_diameter, color, motor\_force, country\_of\_origin зависят от ключа id, поскольку, зная id машины, мы можем однозначно определить её марку, диаметр колёс, цвет, силу мотора и страну производства.

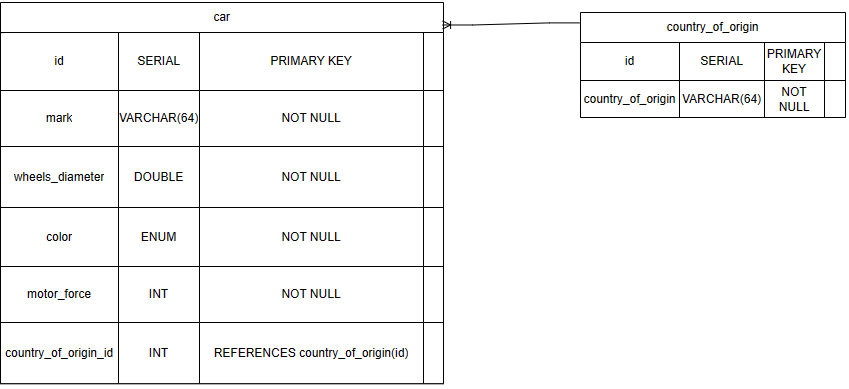
Поля repetitions\_count, what\_was\_said зависят от ключа id, поскольку, зная id повторений голоса, мы сможем однозначно определить их счётчик и что было сказано.

Поля sex, timbre, volume зависят от ключа id, поскольку, зная id голоса, мы можем однозначно определить его пол, тембр и громкость.

Поля description, street\_lenght, has\_slidewalk зависят от ключа id, поскольку, зная id улицы, мы можем однозначно определить длину улицы, наличие тротуара и её описание.

Поля name, description, building\_type зависят от ключа id, поскольку, зная id объекта на улице, мы можем однозначно определить объект и его тип.

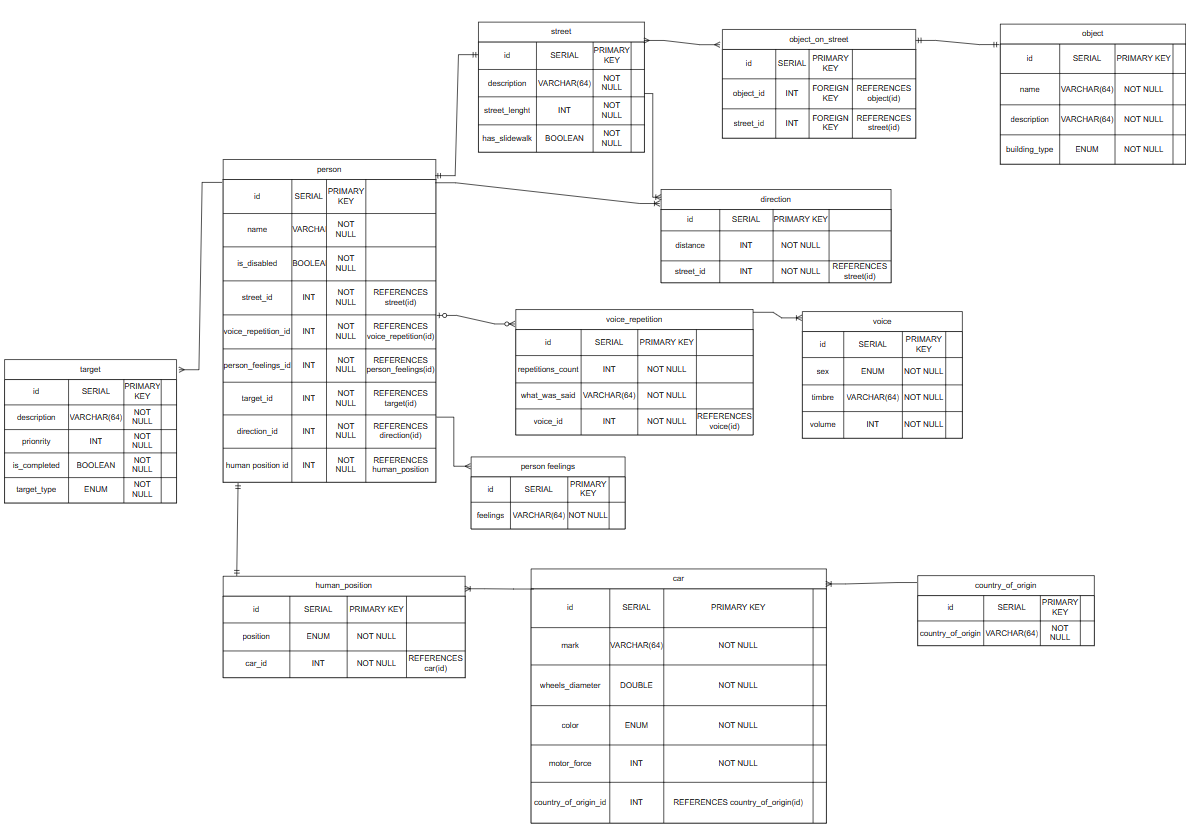
* ***3NF***: Отношение находится в 3NF, если оно находится во 2NF и не содержит транзитивных зависимостей (если у нас есть два атрибута А и В, и В зависит от С, который зависит от А, то мы должны разбить таблицу на две или более таблицы). В моём случае в таблице car атрибут country\_of\_origin зависит от id, а также от атрибута mark, который также зависит от id. Необходимо разбить таблицу на 2 таблицы:



***BCNF:***

Отношение находится в BCNF, если таблица находится в 3NF и ключевые атрибуты составного ключа не должны зависеть от неключевых атрибутов. Моя модель удовлетворяет этим требованиям, поскольку требования BCNF предъявляются только таблицам, имеющим составной PK. Таблицы с простым PK если находятся в 3NF, то находятся и в BCNF.

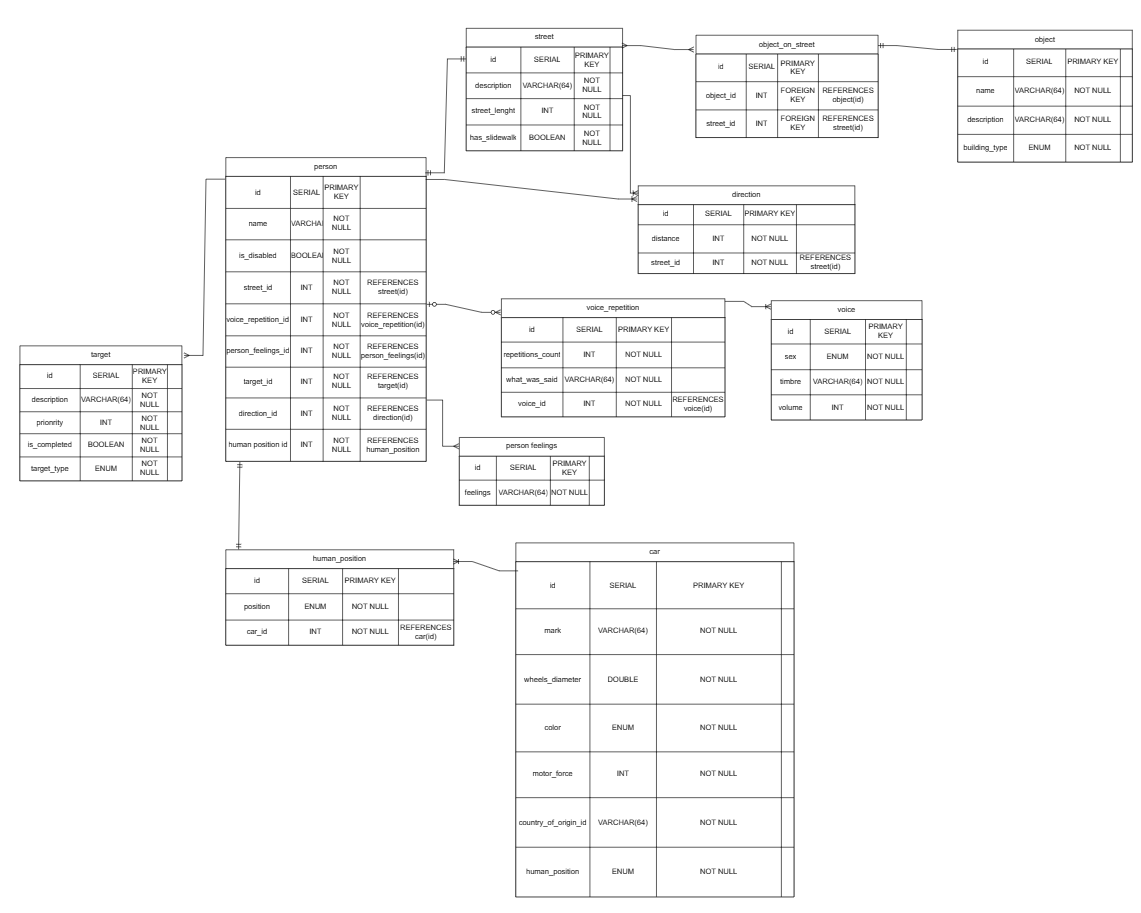
***Нормализованная модель***



***Денормализация***

Можно добавить поля из таблиц human\_position, country\_of\_origin в таблицу car. Будет нарушено соответствие 3NF, потому что появятся транзитивные зависимости. Такое объединение позволит получать машину, страну её происхождения и позицию человека возле этой машины без использования соединений.

***Денормализованная модель***



***Триггер***

*При появлении машины марки “Toyota” у всех людей с именем “Фанат тойоты” главной целью становится дойти до этой машины и встать слева от неё.*

*Если появляется новый человек с именем “Фанат тойоты”, то главной целью становится дойти до машины “Toyota”, если такая есть, и встать слева от неё.*

*При удалении какой-либо улицы удаляются и объекты, расположенные на ней*

*Если появляется новый человек и он инвалид, то его чувства становятся грустными*

***Решение***

CREATE OR REPLACE PROCEDURE replace\_human\_fields() AS $$

BEGIN

IF EXISTS (SELECT \* FROM car WHERE mark='Toyota') THEN

INSERT INTO target(description, priority, is\_completed, target\_type) SELECT 'машина', 1, false, 'физическая'

WHERE

NOT EXISTS (

SELECT description FROM target WHERE description = 'машина');

INSERT INTO human\_position(position\_type, car\_id) SELECT 'справа', 1

WHERE

NOT EXISTS(

SELECT position\_type FROM human\_position WHERE position\_type = 'справа');

UPDATE person SET

target\_id = (SELECT id FROM target WHERE description = 'машина' LIMIT 1),

human\_position\_id = (SELECT id FROM human\_position WHERE position\_type = 'справа' LIMIT 1)

WHERE name = 'Фанат тойоты';

END IF;

END;

$$ LANGUAGE plpgsql;

--первый доп

CREATE OR REPLACE FUNCTION come\_to\_car\_trigger() RETURNS TRIGGER AS $$

BEGIN

IF (NEW.mark = 'Toyota') THEN

CALL replace\_human\_fields();

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER come\_to\_car

AFTER INSERT OR UPDATE ON car

FOR EACH ROW

EXECUTE FUNCTION come\_to\_car\_trigger();

--второй доп

CREATE OR REPLACE FUNCTION set\_come\_to\_car\_trigger() RETURNS TRIGGER AS $$

BEGIN

IF (NEW.name = 'Фанат тойоты') THEN

CALL replace\_human\_fields();

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER set\_come\_to\_car

AFTER INSERT ON person

FOR EACH ROW

EXECUTE FUNCTION set\_come\_to\_car\_trigger();

--select \* from person;

delete from person where name='Фанат тойоты';

--delete from car where mark='Toyota';

--delete from target where description = 'машина';

--delete from human\_position where position\_type='справа';

insert into person(name, is\_disabled, street\_id, voice\_repetition\_id, person\_feelings\_id, target\_id, direction\_id, human\_position\_id) VALUES ('Фанат тойоты', true, 2, 1, 1, 1, 1, 1);

--select \* from person;

--insert into car(mark, wheels\_diameter, color, motor\_force, country\_of\_origin\_id) VALUES('Toyota', 12.5, 'белый', 123, 1);

select \* from car;

--третий доп

DROP MATERIALIZED VIEW object\_ids;

CREATE MATERIALIZED VIEW object\_ids AS (SELECT \* FROM object\_on\_street);

CREATE OR REPLACE FUNCTION delete\_objects\_trigger() RETURNS TRIGGER AS $$

BEGIN

REFRESH MATERIALIZED VIEW object\_ids;

DELETE FROM object\_on\_street WHERE street\_id = OLD.id;

DELETE FROM objects WHERE objects.id IN (SELECT object\_id FROM object\_ids WHERE street\_id = OLD.id);

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER delete\_objects

BEFORE DELETE ON street

FOR EACH ROW

EXECUTE FUNCTION delete\_objects\_trigger();

--INSERT INTO objects(name, description, building\_type) VALUES ('ступенька', 'не очень и большая', 'элемент инфраструктуры');

--INSERT INTO street(description, street\_lenght, has\_slidewalk) VALUES ('такая себе', 777, true);

--INSERT INTO object\_on\_street(object\_id, street\_id) VALUES (8, 8);

--select \* from objects;

--delete from street where id=8;

--select \* from objects;

--четвертый доп

CREATE OR REPLACE FUNCTION set\_person\_feelings\_trigger() RETURNS TRIGGER AS $$

BEGIN

if (NEW.is\_disabled) THEN

INSERT INTO person\_feelings(feelings) SELECT 'грусть'

WHERE

NOT EXISTS(

SELECT \* FROM person\_feelings WHERE feelings = 'грусть');

NEW.person\_feelings\_id = (SELECT person\_feelings.id FROM person\_feelings WHERE feelings='грусть');

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER set\_person\_feelings

BEFORE INSERT ON person

FOR EACH ROW

EXECUTE FUNCTION set\_person\_feelings\_trigger();

--insert into person(name, is\_disabled, street\_id, voice\_repetition\_id, person\_feelings\_id, target\_id, direction\_id, human\_position\_id) VALUES ('Инвалид', true, 2, 1, 1, 1, 1, 1);

--select \* from person;

# Заключение

В ходе выполнения данной лабораторной работы я научился работать с SQL запросами, делать подзапросы, соединять таблицы и использовать различные команды.