Объектно-реляционные базы данных. Проектирование и создание

1. **Вариант задания (12 вариант)**

Вакансии: волонтерские позиции, название вакансии, организация работодатель, адрес работодателя, диапазон зарплаты, требования к образованию, Обязанности, график работы, требования обязательные, желательные, дата выставления вакансии.

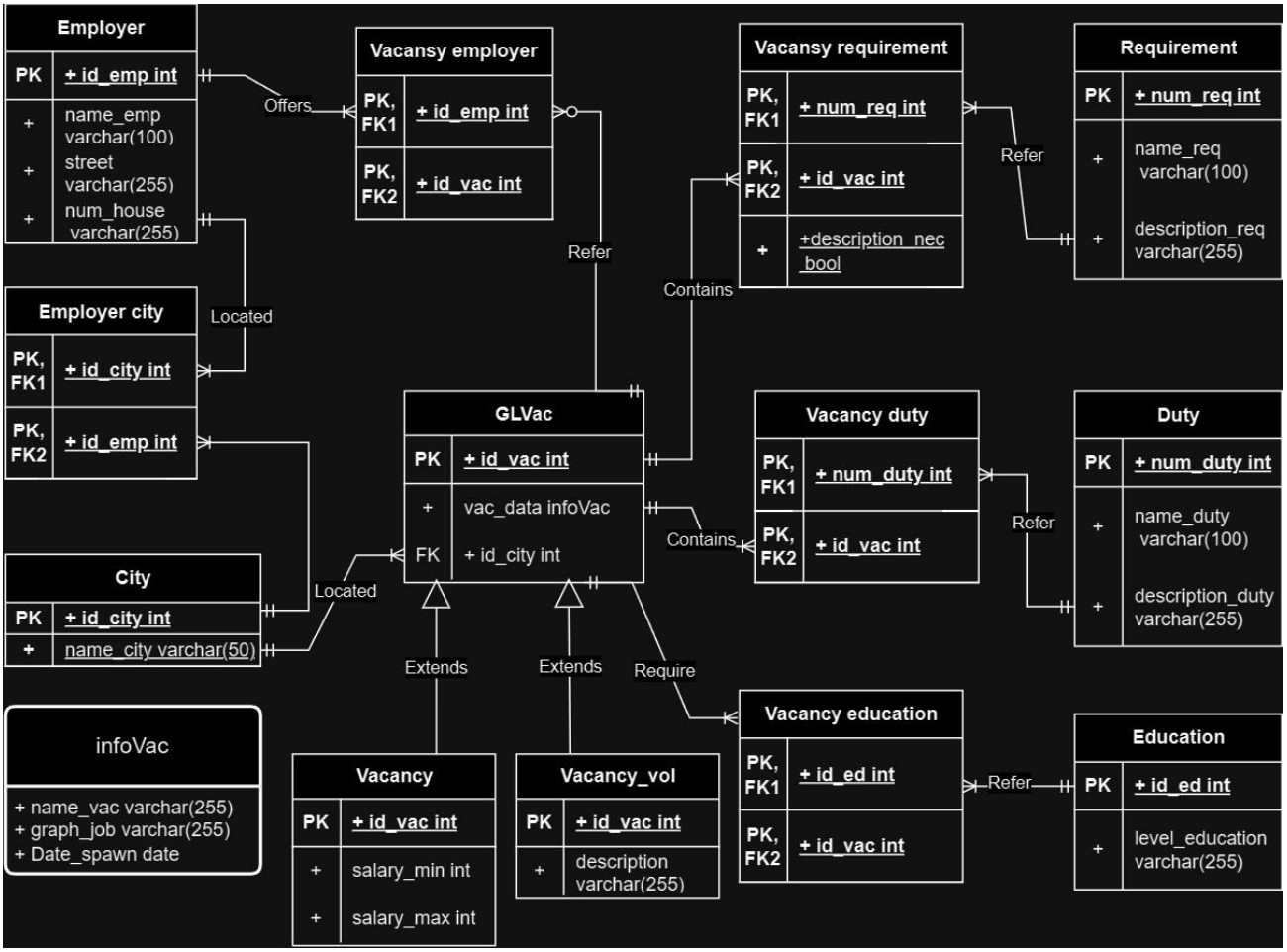
а. вакансии, имеющие в названии SQL, но не заканчивающиеся на него

б. работодатели в Санкт-Петербурге, выставившие несколько вакансий

в. вакансия с наибольшей зарплатой

г. волонтерские позиции с максимальным количеством требований

д. вакансии, в которых нет требования к опыту работы



3. **Созданная модель базы данных:**

```
CREATE TYPE infoVac AS (
    name_vac VARCHAR(255),
    graph_job VARCHAR(255),
    date_spawn DATE
);

CREATE TABLE IF NOT EXISTS Employer(
    id_emp SERIAL NOT NULL PRIMARY KEY,
    name_emp VARCHAR(100) NOT NULL,
    street VARCHAR(255) NOT NULL,
    num_house VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS City(
    id_city SERIAL NOT NULL PRIMARY KEY,
    name_city VARCHAR(50) NOT NULL
```

```
);

CREATE TABLE IF NOT EXISTS Employer_city(
    id_city INT NOT NULL,
    id_emp INT NOT NULL,
    FOREIGN KEY (id_city) REFERENCES City(id_city) ON DELETE CASCADE
ON UPDATE RESTRICT,
    PRIMARY KEY (id_city, id_emp)
);


CREATE TABLE IF NOT EXISTS GLVac(
    id_vac SERIAL NOT NULL,
    vac_data infoVac NOT NULL,
    id_city INT NOT NULL,
    FOREIGN KEY (id_city) REFERENCES City(id_city) ON DELETE CASCADE
ON UPDATE RESTRICT,
    PRIMARY KEY (id_vac)
);

CREATE TABLE IF NOT EXISTS Vacancy(
    salary_min INT,
    salary_max INT,
    FOREIGN KEY (id_city) REFERENCES City(id_city) ON DELETE CASCADE
ON UPDATE RESTRICT,
    PRIMARY KEY (id_vac)
) INHERITS (GLVac);

CREATE TABLE IF NOT EXISTS Vacancy_vol(
    description varchar(255) NOT NULL,
    FOREIGN KEY (id_city) REFERENCES City(id_city) ON DELETE CASCADE
ON UPDATE RESTRICT,
    PRIMARY KEY (id_vac)
) INHERITS (GLVac);


CREATE TABLE IF NOT EXISTS Vacancy_employer(
    id_emp INT NOT NULL,
    id_vac INT NOT NULL,
    FOREIGN KEY (id_emp) REFERENCES Employer(id_emp) ON DELETE
CASCADE ON UPDATE RESTRICT,
    PRIMARY KEY (id_emp, id_vac)
```

```
);


CREATE TABLE IF NOT EXISTS Requirement(
    num_req SERIAL NOT NULL,
    name_req VARCHAR(100) NOT NULL,
    description_req VARCHAR(255) NOT NULL,
    PRIMARY KEY (num_req)
);

CREATE TABLE IF NOT EXISTS Vacancy_requirement(
    num_req INT NOT NULL,
    id_vac INT NOT NULL,
    description_nec BOOLEAN NOT NULL,
    FOREIGN KEY (num_req) REFERENCES Requirement(num_req) ON DELETE
CASCADE ON UPDATE RESTRICT,
    PRIMARY KEY (num_req, id_vac)
);

CREATE TABLE IF NOT EXISTS Duty(
    num_duty SERIAL NOT NULL,
    name_duty VARCHAR(100) NOT NULL,
    description_duty VARCHAR(255) NOT NULL,
    PRIMARY KEY (num_duty)
);

CREATE TABLE IF NOT EXISTS Vacancy_duty(
    num_duty INT NOT NULL,
    id_vac INT NOT NULL,
    FOREIGN KEY (num_duty) REFERENCES Duty(num_duty) ON DELETE
CASCADE ON UPDATE RESTRICT,
    PRIMARY KEY (num_duty, id_vac)
);

CREATE TABLE IF NOT EXISTS Education(
    id_ed SERIAL NOT NULL,
    level_education VARCHAR(255) NOT NULL,
    PRIMARY KEY (id_ed)
);

CREATE TABLE IF NOT EXISTS Vacancy_education(
    id_ed INT NOT NULL,
```

```
    id_vac INT NOT NULL,
    FOREIGN KEY (id_ed) REFERENCES Education(id_ed) ON DELETE CASCADE
ON UPDATE RESTRICT,
    PRIMARY KEY (id_ed, id_vac)
);

--
_____

CREATE OR REPLACE FUNCTION check_VacReqVacEdVacDutyVacEmp()
RETURNS TRIGGER AS $$
BEGIN
   UPDATE Vacancy_requirement SET id_vac = NEW.id_vac
    WHERE id_vac=old.id_vac;
    UPDATE Vacancy_education SET id_vac = NEW.id_vac
    WHERE id_vac=old.id_vac;
    UPDATE Vacancy_duty SET id_vac = NEW.id_vac
    WHERE id_vac=old.id_vac;
    UPDATE Vacancy_employer SET id_vac = NEW.id_vac
    WHERE id_vac=old.id_vac;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

--
_____

CREATE OR REPLACE FUNCTION check_delVacReqVacEdVacDutyVacEmp()
RETURNS TRIGGER AS $$
BEGIN
   DELETE from Vacancy_requirement
    WHERE id_vac= old.id_vac;
    DELETE from Vacancy_education
    WHERE id_vac=old.id_vac;
    DELETE from Vacancy_duty
    WHERE id_vac=old.id_vac;
    DELETE from Vacancy_employer
    WHERE id_vac=old.id_vac;
   RETURN NEW;
END;
```

```
$$ LANGUAGE plpgsql;


CREATE TRIGGER updForeignInsGLVac
AFTER UPDATE OF id_vac ON GLVac
FOR EACH ROW
EXECUTE FUNCTION check_VacReqVacEdVacDutyVacEmp();

CREATE TRIGGER delForeignInsGLVac
BEFORE DELETE ON GLVac
FOR EACH ROW
EXECUTE FUNCTION check_delVacReqVacEdVacDutyVacEmp();



--
_____

CREATE OR REPLACE FUNCTION check_VacAndVacVol()
RETURNS TRIGGER AS $$
BEGIN
   IF NOT EXISTS (SELECT 1 FROM GLVac WHERE id_vac = NEW.id_vac) THEN
    RAISE EXCEPTION 'GLVac with id % does not exist', NEW.id_vac;
    END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER checkGLVacInsVacEmp
BEFORE INSERT OR UPDATE ON Vacancy_employer
FOR EACH ROW
EXECUTE FUNCTION check_VacAndVacVol();

--
_____


CREATE TRIGGER checkGLVacInsVacReq
BEFORE INSERT OR UPDATE ON Vacancy_requirement
```

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

--
_____


CREATE TRIGGER checkGLVacInsVacDuty
BEFORE INSERT OR UPDATE ON Vacancy_duty
FOR EACH ROW
EXECUTE FUNCTION check_VacAndVacVol();

--
_____

CREATE TRIGGER checkGLVacInsVacEd
BEFORE INSERT OR UPDATE ON Vacancy_education
FOR EACH ROW
EXECUTE FUNCTION check_VacAndVacVol();
```