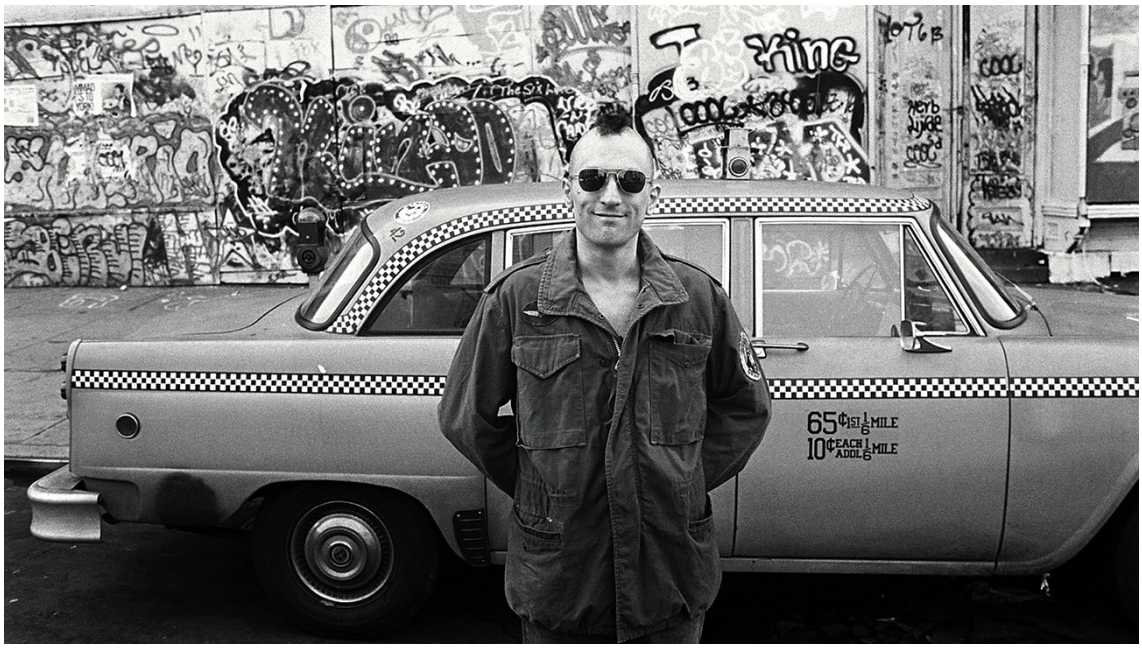
taxopark

Egor Gladilin, Timofey Bakurov, Artem Ilin



December 27, 2023

1 Introduction

We are creating an online taxi company management system, thanks to which the owner of the network, taxi drivers, service personnel and investors can manage the enterprise and bring the company's management processes to a new level. Using modern database management automation technologies such as PostgreSQL and web interfaces will allow you to enjoy working with the AU.

Project Theme: Online Taxi Fleet Management System

A system that enables the network owner, drivers, service personnel, and investors to manage the enterprise and elevate the company's managerial processes to a new level. The use of modern database management technologies, such as PostgreSQL, and web interfaces will provide a satisfying experience in working with the information system.

List of Users and Their Requirements:

Network Director

- Monitor the number of investors.
- Receive reports on profits and expenses.
- Track the amount of dividends paid to investors.
- Provide instructions on staffing and fleet operations.

Parking Director

- Manage supplies for the operational capability of the parking fleet.
- Monitor the technical condition of vehicles.
- Manage the technical maintenance and repair of the fleet.
- Manage personnel, issue penalties and positive sanctions.

Parking Manager

- Order technical maintenance for specific vehicles.
- Create schedules for vehicle operations.
- Track the availability of vehicles for drivers.

Accountant

- Maintain the financial records of the taxi fleet.
- Generate financial reports and budgets.
- Monitor tax reporting.
- Track accounts and payments to contractors.

Technician

- Receive repair requests for vehicles.
- Plan and conduct technical maintenance of vehicles.

Security Guard

- Monitor the security of the parking lot and vehicles through patrols.
- Receive tasks from the chief of security.
- Create incident reports.

Driver

- Receive information about available vehicles.
- Track personal statistics and earnings.

Investor

- Track financial indicators and profitability of the taxi fleet.
- Monitor the percentage of their investments in the overall public capital of the taxi fleet.
- Track the stability of dividend payments.

Functional Requirements for the System by User Roles:

Network Director

- Add and remove investors.
- Set dividend percentages.
- Add and remove employees in the HR table.
- Add and remove cars in the fleet.
- Add and remove contractors.

Parking Director

- Add and remove fuel and spare part supplies.
- Add and remove cars in the parking lot.
- Add and remove employees in the parking staff.

Parking Manager

- Approve car repairs requested by technicians.
- Block a car for dispatch.
- Block a driver from taking a shift in case of medical or other contraindications.
- Log driver's shift start and end times.

Accountant

- Manage Profit and Loss (PnL) financial reporting.
- Track and update payment statuses.
- Manage salary payments to parking staff.

Technician

- Generate and update repair requests for cars.
- Manage spare parts inventory.

Security Guard

- Log visitors to the parking lot (entry and exit).
- Add patrol reports.

Driver

- Accept and return cars to the taxi fleet.
- Register the presence of damages.

Investor

- View the financial reports of the company.

Description of Additional Entities:

Car

- ID (int)
- Brand (string)
- Model (string)
- VIN number (string)
- License Plate (string)
- Year of Manufacture (byte)
- Mileage (byte)
- Technical Condition (VALUES(string): Good, Satisfactory, Requires Repair)
- Paint Condition (VALUES(string): Good, Satisfactory, Requires Repair)
- Tariff (VALUES(string): Economy, Standard, Standard Plus, Business, Premium)
- Permission to Use (Boolean)
- Rent Cost

Parking

- ID (int)
- Name (string)
- Address (string)
- Number of Parking Spaces (byte)
- Number of Employees (byte)
- Number of "Zones" for Patrols (byte)

Data Constraints:

Limit on the Number of Cars Assigned to a Driver

- A driver can be assigned only one car for line duty.

Payment Transaction Limitations

- Additional payments cannot be processed if the company's final balance is zero.

Car Availability Restrictions

- Manager must check the car status before assigning it to a driver.

Access Restrictions

- Technician can only edit tables with information about spare parts and repair requests.
- Accountant has the right to view exclusively financial information and edit only accounting documents.
- Driver has the right to view only personal logs.
- Security guard has the right to edit and view only tables with parking security information.
- Parking manager has the right to edit all tables except those with financial information.
- Parking director has the right to edit all tables except those with financial information.
- Network director has the right to edit all tables.
- Investor has the right to view all tables without editing capability.

Driver Status Requirement

- A driver must have a trip permission status: {1 – allowed, 0 – forbidden}.

Repair Request Date Constraint

- The repair request's end date must be greater than or equal to the car repair end date, and the car must be unavailable during the repair.

Employee Type Count Constraint

- The number of employees with types "Director," "Guard," "Technician," "Driver" must be greater than zero.

Investor Dividend Percentage Constraint

- The percentage of dividends to investors cannot be negative.

Fuel and Spare Parts Inventory Constraint

- The quantity of fuel and spare parts cannot be negative.

Car Mileage Constraint

- If the mileage exceeds 1,000,000, the car must be written off or sold.

Car Condition Constraint

- If the technical condition or paint condition of the car is unsatisfactory, repair must be carried out.

Parking Capacity Constraint

- The number of cars in the parking lot cannot exceed the number of parking spaces.

Sanctions on Drivers

- Sanctions on the driver may include:
  - Increase in rent cost by $n\%$
  - Decrease in rent cost by $n\%$
  - Work suspension

Repair Request Status Constraint

- The repair request must have the status: 1 – approved, 2 – not approved.

Limit on the Number of Cars per Technician

- A technician can repair a maximum of 2 cars simultaneously.

Functional Dependencies on Tables:

1. Users_HR (Users)

- **UserID** → UserName, UserRole, UserEmail, UserPassword, UserStatus

2. Investors

- **InvestorID** → UserID, InvestmentPercentage

3. NetworkDirector

- **DirectorID** → UserID, Salary

4. ParkingDirectors

- **ParkingDirectorID** → UserID, ParkingID, Salary

5. ParkingManagers

- **ManagerID** → UserID, ParkingID, Salary

6. Accountants

- **AccountantID** → UserID, ParkingID, Salary

7. Technicians

- **TechnicianID** → UserID, ParkingID, Salary

8. SecurityGuards

- **GuardID** → UserID, ParkingID, Salary

9. Driver

- **DriverID** → UserID, DriverLicense, DriverStatus, ParkingID

10. Car

- **CarID** → Brand, Model, VIN, LicensePlate, Year, Mileage, TechnicalCondition, PaintCondition, Tariff, PermissionToUse, RentCost, ParkingID

11. ParkingLot

- **ParkingID** → Name, Address, Capacity, EmployeeCount, PatrolZones

12. RepairRequest

- **RequestID** → CarID, TechnicianID, ApprovalStatus, StartDate, EndDate, UsedParts

13. Supplies

- **SupplyID** → ParkingID, FuelQuantity, SparePartsQuantity

14. SecurityLog

- **LogID** → GuardID, VisitorLog, PatrolZone, PatrolStartTime, PatrolEndTime, PatrolReports

15. Sanctions

- **SanctionID** → DriverID, Type, PercentageChange, WorkSuspension

16. CompanyBalance

- **OperationID** → OperationTime, InitialBalance, OperationAmount, OperationType, FinalBalance

17. RentalLog

- **RentalID** → DriverID, CarID, RentalStartTime, RentalEndTime, TotalCost

Challenges without Normalization:

Data Redundancy:

- Duplication of user information (Users_HR) as role and status data are stored in each employee table (Director, Manager, Accountant, Technician, SecurityGuard, Driver). This can lead to difficulties in updating user information in multiple places.

Dependency on Role Changes:

- If the role structure changes (e.g., a new role is added), changes need to be made to multiple tables (Director, Manager, Accountant, Technician, SecurityGuard, Driver).

Implicit Connection Between Objects:

- Salary data is stored in each employee table. This may cause issues if salary information needs to be modified or historical salary tracking is required.

Direct Storage of Repair Request Data:

- Repair request data is directly stored, and UsedParts field stores part IDs. This may lead to difficulties in processing and searching for part data in different requests.

Sanctions Storage in a Single Table:

- Sanction data is tied to the driver, and it could be beneficial to separate this information into a dedicated table for better normalization.

Rental Logic Complexity:

- RentalLog includes information related to rent cost, which is associated with car data. Without normalization, this may lead to data duplication and loss of integrity if car data changes.

Figure 1: The final ER diagram

To watch diagram in more details go here.

2 SQL DDL script for database creation based on normalized schema

The next code will be directly imported from a file

```sql
--1. Table Users:
CREATE TABLE Users (
    UserID SERIAL PRIMARY KEY,
    UserName VARCHAR(255),
    UserRole VARCHAR(50),
    UserEmail VARCHAR(255),
    UserPassword VARCHAR(255),
    UserStatus BOOLEAN
);

--2. Table Investors:
CREATE TABLE Investors (
    InvestorID SERIAL PRIMARY KEY,
```

```
    UserID INTEGER REFERENCES Users(UserID),
    InvestmentPercentage FLOAT
);

--3. Table Employees:
CREATE TABLE Employees (
    EmployeeID SERIAL PRIMARY KEY,
    UserID INTEGER REFERENCES Users(UserID),
    RoleID INTEGER REFERENCES Roles(RoleID),
    ParkingID INTEGER REFERENCES ParkingLot(ParkingID),
    Salary DECIMAL(15, 2)
);

--4. Table Roles:
CREATE TABLE Roles (
    RoleID SERIAL PRIMARY KEY,
    RoleName VARCHAR(50)
);

--5. Table Driver:
CREATE TABLE Driver (
    DriverID SERIAL PRIMARY KEY,
    UserID INTEGER REFERENCES Users(UserID),
    DriverLicense VARCHAR(20),
    DriverStatus BOOLEAN,
    ParkingID INTEGER REFERENCES ParkingLot(ParkingID)
);

--6. Table Car:
CREATE TABLE Car (
    CarID SERIAL PRIMARY KEY,
    Brand VARCHAR(50),
    Model VARCHAR(50),
    VIN VARCHAR(17),
    LicensePlate VARCHAR(15),
    Year SMALLINT,
    Mileage INT,
    TechnicalCondition VARCHAR(20),
    PaintCondition VARCHAR(20),
    Tariff VARCHAR(10),
    PermissionToUse BOOLEAN,
    RentCost DECIMAL(10, 2),
    ParkingID INTEGER REFERENCES ParkingLot(ParkingID)
);

--7. Table ParkingLot:
CREATE TABLE ParkingLot (
    ParkingID SERIAL PRIMARY KEY,
    Name VARCHAR(255),
    Address VARCHAR(255),
    Capacity SMALLINT,
    EmployeeCount SMALLINT,
    PatrolZones SMALLINT
);

--8. Table RepairRequest:
CREATE TABLE RepairRequest (
    RequestID SERIAL PRIMARY KEY,
    CarID INTEGER REFERENCES Car(CarID),
    TechnicianID INTEGER REFERENCES Employees(EmployeeID),
    ApprovalStatus BOOLEAN,
    StartDate DATE,
    EndDate DATE,
    UsedParts SERIAL REFERENCES Supplies(SupplyID)
);

--9. Table Supplies:
CREATE TABLE Supplies (
    SupplyID SERIAL PRIMARY KEY,
    ParkingID INTEGER REFERENCES ParkingLot(ParkingID),
    FuelQuantity DECIMAL(10, 2),
```

```
    SparePartsQuantity SMALLINT,
    SupplyDate DATE
);

--10. Table SecurityLog:
CREATE TABLE SecurityLog (
    LogID SERIAL PRIMARY KEY,
    GuardID INTEGER REFERENCES Employees(EmployeeID),
    VisitorLog TEXT,
    PatrolZone VARCHAR(50),
    PatrolStartTime TIME,
    PatrolEndTime TIME,
    PatrolReports TEXT
);

--11. Table Sanctions:
CREATE TABLE Sanctions (
    SanctionID SERIAL PRIMARY KEY,
    DriverID INTEGER REFERENCES Driver(DriverID),
    Type VARCHAR(50),
    PercentageChange DECIMAL(5, 2),
    WorkSuspension BOOLEAN
);

--12. Table CompanyBalance:
CREATE TABLE CompanyBalance (
    OperationID SERIAL PRIMARY KEY,
    OperationTime TIMESTAMP,
    InitialBalance DECIMAL(15, 2),
    OperationAmount DECIMAL(15, 2),
    OperationType VARCHAR(10),
    FinalBalance DECIMAL(15, 2)
);

--13. Table RentalLog:
CREATE TABLE RentalLog (
    RentalID SERIAL PRIMARY KEY,
    DriverID INTEGER REFERENCES Driver(DriverID),
    CarID INTEGER REFERENCES Car(CarID),
    RentalStartTime TIMESTAMP,
    RentalEndTime TIMESTAMP,
    TotalCost DECIMAL(15, 2)
);
```

3 SQL DML queries that implement functional requirements

1. Queries for CEO

```
-- insert new investor

INSERT INTO users(username, userrole,useremail,userpassword,userstatus)
VALUES ('some name', 'Investor', 'email@gmail.com','pass',true);

INSERT INTO investors (userid)
(SELECT userid  FROM users u
WHERE userrole = 'Investor'
EXCEPT
SELECT userid FROM investors i);

--set dividens share

UPDATE investors
SET investmentpercentage = /* share */
WHERE investorid = (select investorid from investorse where name = /* some name */);

-- remove and add stuff

delete from users
where userid = (select investorid from investorse
```

```
where name = /* some name */);

delete from employees
where userid = (select employeeid from employees
where name = /* some name */ and roleid = (select roles from rolename = /* role */ );

INSERT INTO users(username, userrole,useremail,userpassword,userstatus)
VALUES ('some name', 'Stuff', 'email@gmail.com','pass',true);

INSERT INTO employees (userid)
(SELECT userid  FROM users u
WHERE userrole = 'Stuff'
EXCEPT
SELECT userid FROM employees e);

-- remove and add new car

delete from cars
where carid = (select carid from car
where licenseplare = /* licnse plate */);

insert into car (brand, model, vin, licenseplate, year, mileage,technicalcondition,
    paintcondition,tariff,permissiontouse,rentcost,parkingid)
values ( /* coresponding data */);

-- remove and add counter-agent

delete from driver
where userid = (select driverid from driver
where driverlicense = /*licence code*/);

INSERT INTO users(username, userrole,useremail,userpassword,userstatus)
VALUES ('some name', 'Driver', 'email@gmail.com','pass',true);

INSERT INTO driver (userid)
(SELECT userid  FROM users u
WHERE userrole = 'Driver'
EXCEPT
SELECT userid FROM driver d);
```

2. Chief parking officer

```
    --add car to parking

    UPDATE car
    SET parkingid = (select parkingid from parkinglot where name = /* some name */)
    WHERE carid =
    (select carid from car
    where licenseplare = /* licnse plate */);

    --add employee to parking

    UPDATE employees
    SET parkingid = (select parkingid from parkinglot where name = /* some name */)
    WHERE carid =
    (select carid from car
    where licenseplare = /* licnse plate */);

    -- supply fuel/spare parts

    insert into Supplies (parkingid, fuealquantity,sparepartsquantity, supplydate)
    value (/* corresponding values */);
```

3. Parking manager

```
    --approve repair request

    update repairrequest
    set approvalstatus = true
    where startdate < current_date and enddate > current_date;

```

```
    --permit or block car to use

    UPDATE car
    SET permissiontouse = /* true or false */
    WHERE carid = (select carid from car where licenseplare = /* licnse plate */) ;

    -- permit driver ro work

    UPDATE driver
    SET driverstatus = /* true or false */
    WHERE carid = (select carid from car where licenseplare = /* licnse plate */);

    -- logging working start time and end time of a car

    insert into rentallog(carid, driverid, rentalstarttime)
    values((select carid from car where licenseplare = /* licnse plate */), (select driverid
        from driver where name = 'some name'),current_timestamp;

    UPDATE rentallog
    SET rentalendtime = current_timestamp, totalcost = /* some cost */
    where carid = (select carid from car where licenseplare = /* licnse plate */) limit1;
```

4. Accountant
   Managing Profit and Loss (PnL) Financial Reports:

```
-- Profit and Loss report data
SELECT OperationID, OperationTime, InitialBalance, OperationAmount, OperationType, FinalBalance
FROM CompanyBalance
WHERE OperationType = 'PnL';
```

   Managing Payroll for Parking Lot Employees:

```
-- retrieve and update payroll information for employees
SELECT EmployeeID, UserID, RoleID, Salary
FROM Employees
WHERE RoleID IN (SELECT RoleID FROM Roles WHERE RoleName = 'Employee');

-- update the salary for a specific employee
UPDATE Employees
SET Salary = <NewSalary>
WHERE EmployeeID = (select employeeid from employees where name = 'some name');
```

5. Technician

```
--assign repair request

insert into repairrequest (carid, technicianid,startdate,enddate,usedparts)
values ((select carid from car where licenseplare = /* licnse plate */),(select employeeid from
    employees where name = /* name */ and roleid = (select roleid from roles where name = '
    Technician'),current_date, /* some date */,/* supply id */);
```

6. Security guard
   Logging Visitors (Entry and Exit):

```
-- log the entry of a visitor to the parking lot
INSERT INTO SecurityLog (GuardID, VisitorLog, PatrolZone, PatrolStartTime)
VALUES (<GuardID>, 'Visitor entered the parking lot', '<PatrolZone>', CURRENT_TIME);

-- log the exit of a visitor from the parking lot
INSERT INTO SecurityLog (GuardID, VisitorLog, PatrolZone, PatrolEndTime)
VALUES (<GuardID>, 'Visitor exited the parking lot', '<PatrolZone>', CURRENT_TIME);
```

   Adding Patrol Reports:

```
--  add patrol reports
UPDATE SecurityLog
SET PatrolReports = '<PatrolReport>'
WHERE GuardID = (select employee from employees where name = /* some name */ and roleid = (
    select roleid from ) AND PatrolStartTime = CURRENT_DATE;
```

7. Taxi driver
   Receiving and Returning a Car:

```sql
-- taxi driver to receive a car
UPDATE Car
SET PermissionToUse = true
WHERE CarID = (select carid from car where licenseplare = /* licnse plate */), AND ParkingID = (
    select parkingid from parkinglot where name = /* some name */);

-- Code to allow a taxi driver to return a car
UPDATE Car
SET PermissionToUse = false
WHERE CarID = (select carid from car where licenseplare = /* licnse plate */);
```

8. Investor: view the financial report of the company

```sql
-- Query to retrieve financial report for an investor
SELECT
    c.OperationTime AS operation_time,
    c.InitialBalance AS initial_balance,
    c.OperationAmount AS operation_amount,
    c.OperationType AS operation_type,
    c.FinalBalance AS final_balance
FROM
    CompanyBalance c
    JOIN Investors i ON c.OperationID = i.UserID
WHERE
    i.InvestorID = (select invrestorid from investors where name = 'some name');
```

4 Requests grouped into transactions

```sql

begin;

update rentallog set renatlendtme = current_timestamp, total cost = /* some cost */ where carid =  (
    select carid from licenseplate = /* licence code */);

transaction time = current_timestamp;

insert into companybalance (operationtime, initialbalance, operationtype)
values transactiontime, (select finalbalance from company balance ORDER BY operationtime DESC limit
    1),'Income';

update companybalance set totalbalance = initialbalance + (select totalcost from rentallog where
    carid = (select carid from car where licenseplate = /* license code*/ limit 1), operationamount
    = (select totalcost from rentallog where carid = (select carid from car where licenseplate = /*
    license code*/ limit 1) where operationtime = transactiontime;

if (select finalbalance from companybalance order by operationtime desc limit 1) - (select
    finalbalance from companybalance order by operationtime desc limit 1) = 0 or (select
    operationamount from companybalance order by operationtime desc limit 1)=0then
rollback;
endif;

commit;
```