Ministry of Education and Science of Ukraine

Odesa Polytechnic National University

Institute of Computer Systems

Department of Information Systems

Shkekina Yuliia Arturivna,

student of group AI-221

**COURSEWORK**

"Database for automating payroll calculations in university accounting"

Specialty:

122 Computer science

Educational program: Computer science

Head:

Head Maria Hennadiyevna,

Cand. technical Sciences, associate professor

### Odesa- 2023

CONTENT

Tasks for the course work………………………………………………………………3

Introduction……………………………………………………………………………..5

1 Analysis of the subject area and formulation of the problem ……………….………..6

1.1 Description of the subject area ………………………..………….….………6

1.2 Description of system users and their history (User Story)……….….………6

2 Database designaccording to option 3…………………………………………………8

3 Selection of software …………………………………………………………...……12

4 Creating a database………………………………………………………………...…13

4.1 Creation of tables……………………………………………….………...…13

4.2 Creation of views…………………………………………………………....16

4.3 Creation of triggers……………………………………………….….……...17

4.4 Creation of stored procedures (functions) …………………………..……....21

5 Data manipulation ……………………………………………….…………………...30

5.1 Sampling operator ………………………………………….………….……30

6 Creating users and assigning access rights………………………...…………………33

Conclusion….…………………………………………………………………………34

List of used sources…………………………………………………………………….35

Odesa Polytechnic National University

Institute of Distance and Correspondence Education

Department of Information Systems

TASK

FOR COURSE WORK

student Shkekina Yuliia Arturivna group AI-221

1. Topic of work "A database for automating payroll calculations in higher education accounting»

2. Deadline for the student to submit the completed work 17.12.2023

3. Initial data for the project (work): Salary is accrued to employees depending on the established salary. During the period of illness, the employee is credited with 50% of the salary. Income tax is deducted from the total salary. Employee data: personal card number; full name; position; salary; marital status and number of children; data on absence from work due to illness (dates of illness and recovery), etc. Employees are charged bonuses and Salary supplementt

4. The content of the calculation and explanatory note: (list of questions to be developed) setting the problem, designing the database, choosing software, creating a database, manipulating data, creating users and assigning access rights.

5. List of graphic material (with exact indication of mandatory drawings):\_block diagram, screenshots that confirm the correct operation of the program

The task was issued on 05.10.2023

The task was accepted for implementation on 05.10.2023

INTRODUCTION

In modern society, education plays a far from small role. Among us there are many university teachers who have to store information about their employees. The appearance of the Internet on the planet marked the beginning of the development of entire industries related to information processing. Thanks to the development of the Internet space the involvement of millions of users, organizations and structures, this was the beginning of the emergence of databases.

The topic I have chosen is considered relevant since thousands of people are hired by universities every year and information about them is stored in databases.

The purpose of this coursework is the analysis of the subject area "Automation of payments to university employees", development and design of a database for the university, which provides storage, accumulation and processing of information about payments. The pgAdmin 4 program was used as a DBMS to implement the database.

The following tasks were set for this term paper:

* Analysis of the subject area of ​​university accounting;
* Database design for the subject area;
* Acquisition of skills in creating various requests and reports;
* Creation of functions that facilitate the use of the database;

The automated service system will allow the user to save time searching for information about the payments he needs.

1 ANALYSIS OF THE SUBJECT FIELD AND STATEMENT OF THE PROBLEM

* 1. Description of the subject area of ​​higher education accounting
* Subject area: university accounting
* Salary is charged to employees depending on the established salary. During the period of illness, the employee is credited with 50% of the salary.
* Employee data: personal card number; full name; position; salary; marital status and number of children; data on absence from work due to illness (dates of illness and recovery), etc. Employees are charged bonuses and Salary supplementt
  1. Description of system users and their history (User Story)

*Employee*

1. As an employee, I can view information about bonuses and Salary supplementt to know about my earnings.
2. As an employee, I can view information about my position to monitor the correctness of the payroll.
3. As an employee, I can view the number of work days missed due to illness to find out about my salary reduction.
4. As an employee, I can view the number of days I have missed due to illness to find out about my health insurance.
5. As an employee, I can look up child information to see if benefits are available.

*Director*

1. As a director, I can view employee information to see who works for me.
2. As a director, I can look at information about worker positions to see if I need to hire new employees.
3. As a director, I can look at employee status information to see how many employees can request maternity leave.
4. As a director, I can look at workers' illness information to see if a worker can be fired because of poor health.
5. As a director, I can look at bonus and bonus information to see which employees are working hard.

*Accountant*

1. As an accountant, I can look at worker position information to track payroll costs.
2. As an accountant, I can change the amount of the bonus or allowance to follow the fair reward of the workers.
3. As an accountant, I can view information about workers to follow the status of workers.
4. As an accountant, I can look at information about workers' health conditions to find out how much money the university will save.
5. As an accountant, I can look at bonus or bonus information to see if employees are being over-rewarded.

2 DESIGN OF THE ACCOUNTING DATABASE OF THE UNIVERSITY

The selected essential entities characterizing the subject area and their characteristics are presented in Table 2.1.

|  |  |
| --- | --- |
| **Essence** | **Characteristic** |
| Position | Contains information about the positions of employees and their salary |
| Marital status | Contains information about the employee's marital status |
| Children | Contains information about children |
| Diseases | Contains information about employee illnesses and dates of illness and recovery |
| Awards | Contains information about employee bonuses |
| Salary supplement | Contains information about employee salary supplement |
| Workers | Contains information about employees |
| Child worker | Contains the IDs of employees and children |

Table 2.1 – Essential entities characterizing the subject area

Selected properties of objects (attributes), their data types and keys are presented in Table 2.2.

|  |  |  |  |
| --- | --- | --- | --- |
| **Essence** | **Property** | **Data type** | **Key** |
| Position | \*id\_position | numerical | Primary key |
| \*employee\_id | numerical | Foreign key |
| name | symbolic |  |
| salary | numerical |  |
| Marital status | \*state\_id | numerical | Primary key |
| state | symbolic |  |
| Children | \*child\_id | numerical | Primary key |
| name | symbolic |  |
| age | numerical |  |
| sex | symbolic |  |
| Diseases | \*id\_disease | numerical | Primary key |
| name | symbolic |  |
| date\_of illness | date |  |
| recovery\_date | date |  |
| employee\_id | numerical | Foreign key |
| Awards | \*prize\_id | numerical | Primary key |
| reason | symbolic |  |
| Date of receiving | date |  |
| size | symbolic |  |
| employee\_id | numerical | Foreign key |
| Salary supplement | \*id\_salary\_supplement | numerical | Primary key |
| reason | symbolic |  |
| Date of receiving | date |  |
| size | symbolic |  |
| employee\_id | numerical | Foreign key |
| Workers | \*id\_personal card | numerical | Primary key |
| Full name | symbolic |  |
| state\_id | numerical | Foreign key |
| Child worker | \*employee\_id | numerical | foreign key,  Primary key |
| \*child\_id | numerical | foreign key,  Primary key |

Table 2.2 – Selected entity attributes and their characteristics

Defined types of relationships between entities are presented in table 2.3.

|  |  |  |
| --- | --- | --- |
| Essence | Connection | Essence |
| Workers | 1:N | Position |
| Workers | N:1 | Marital status |
| Workers | N: N | Children |
| Workers | 1:N | Diseases |
| Workers | 1:N | Awards |
| Workers | 1:N | Salary supplement |

Table 2.3 – Types of relationships between entities

On the basis of the selected entities, their attributes and relationships between entities, a notation was created in draw.io, which is presented in Fig. 2.1.

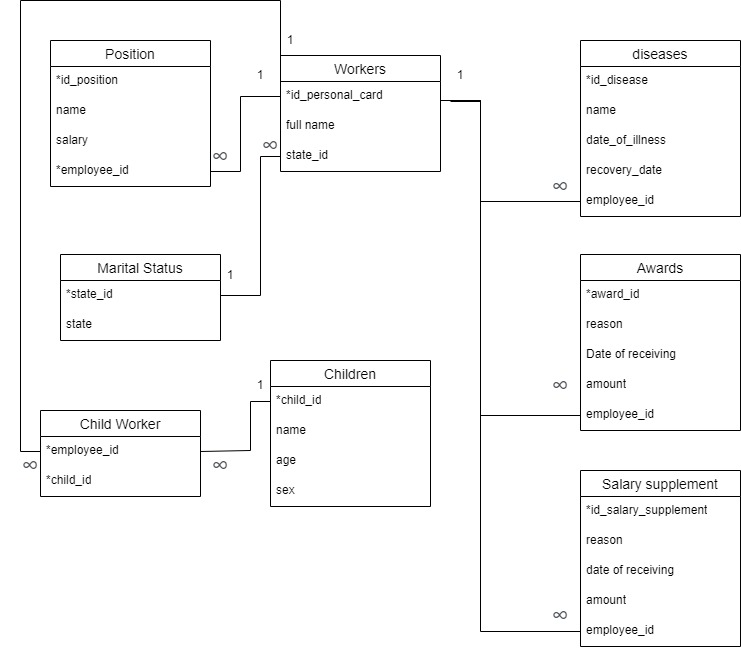


Figure 2.1 – notation in draw.io

3 CHOICE OF SOFTWARE

For this work, I chose pgAdmin 4 because it is a powerful and widely used tool for managing and developing PostgreSQL databases. pgAdmin 4 offers a user-friendly interface that makes it easy to create and manage databases, and to write and execute SQL queries. In addition, pgAdmin 4 has a number of useful features, such as the ability to visualize database structures and query performance, making it an excellent choice for working with PostgreSQL databases.

Another reason I chose pgAdmin 4 is that it is open and available, meaning anyone can download and use it without having to pay for a license. This makes it a good option for those working on a tight budget or who want to try out a tool before choosing a paid solution.

Overall, I chose pgAdmin 4 for this job because it's a powerful and easy-to-use tool for managing and developing PostgreSQL databases, and it's freely available to anyone who wants to use it.

4 CREATION OF DATABASE

4.1 Creation and presentation of tables

CREATE TABLE positions

(

position\_id integer PRIMARY KEY,

position\_name varchar,

position\_salary decimal,

worker\_id integer REFERENCES workers(worker\_id)

);

CREATE TABLE marstatuses

(

marstatus\_id integer PRIMARY KEY,

marstatus\_type varchar

);

CREATE TABLE children

(

kid\_id integer PRIMARY KEY,

kid\_name varchar,

kid\_age integer,

kid\_gender varchar

);

CREATE TABLE workers

(

worker\_id integer PRIMARY KEY,

worker\_name varchar,

marstatus\_id integer REFERENCES marstatuses(marstatus\_id)

);

CREATE TABLE bonuses

(

bonus\_id integer PRIMARY KEY,

bonus\_reason varchar,

bonus\_date varchar,

bonus\_size varchar,

worker\_id integer REFERENCES workers(worker\_id)

);

CREATE TABLE surcharges

(

surcharge\_id integer PRIMARY KEY,

surcharge\_reason varchar,

surcharge\_date varchar,

surcharge\_size varchar,

worker\_id integer REFERENCES workers(worker\_id)

);

CREATE TABLE illnesses

(

illness\_id integer PRIMARY KEY,

illness\_name varchar,

illness\_start date,

illness\_end date,

worker\_id integer REFERENCES workers(worker\_id)

);

CREATE TABLE workers\_kids

(

worker\_id integer REFERENCES workers(worker\_id),

kid\_id integer REFERENCES kids(kid\_id),

PRIMARY KEY(worker\_id, kid\_id)

);

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~

INSERT INTO positions VALUES

(1, 'Teacher', 34000, 1),

(2, 'Professor', 75000, 2),

(3, 'Rector', 50000, 3),

(4, 'Docent', 100000, 4),

(5, 'Cleaner', 20000, 5);

INSERT INTO marstatuses VALUES

(1, 'Married'),

(2, 'Unmarried');

INSERT INTO kids VALUES

(1, 'Daniel', 17, 'Man'),

(2, 'Zinaida', 1, 'Woman'),

(3, 'Oleg', 7, 'Man');

INSERT INTO employees VALUES

(1, "Bayrak Lavrin Arturovych", 1),

(2, "Grebin Peremysla Orestivna", 1),

(3, "Tarnovsky Mukhovist Makarovych", 2),

(4, "Pavelko Valentin Yanovych", 2),

(5, "Kryvda Aglaida Volodymyrivna", 1);

INSERT INTO bonuses VALUES

(1, 'Stayed for the weekend', '2022-01', 1000, 5),

(2, 'Stayed for the weekend', '2022-02', 1000, 5),

(3, 'Stayed for the weekend', '2022-04', 10000, 2),

(4, 'Cleaned the office', '2022-08', 16050, 1),

(5, 'Cleaned the office', '2022-06', 16050, 4);

INSERT INTO surcharge VALUES

(1, 'Took more work', '2022-03', 2000, 3),

(2, 'Took more work', '2022-04', 1350, 3);

INSERT INTO illnesses VALUES

(1, 'Covid-19', '2022-10-10', '2022-10-12', 1),

(2, 'Ornithosis', '2022-07-2', '2022-07-14', 2),

(3, 'Cholera', '2022-08-19', '2022-09-4', 3),

(4, 'Flu', '2022-08-02', '2022-08-02', 3),

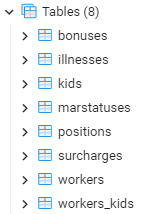
(5, 'Flu', '2022-09-19', '2022-10-10', 5);

INSERT INTO workers\_kids VALUES

(2, 1),

(3, 2),

(5, 2);

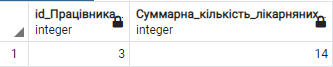


4.2 Creation of representations

* A representation that reflects all patients in August 2022

CREATE OR REPLACE VIEW ill\_for\_2022\_08 AS

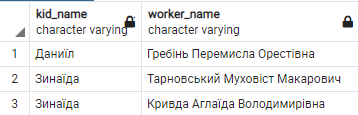
SELECT \* FROM summed\_absent\_days('2022-08') sad WHERE sad.Total\_number\_of\_hospitals > 0;



* A performance that reflects children and their parents

CREATE OR REPLACE VIEW kids\_parents AS

SELECT k.kid\_name, w.worker\_name FROM kids k, workers w, workers\_kids wk WHERE wk.kid\_id = k.kid\_id AND wk.worker\_id = w.worker\_id;



* A representation that displays all employees with a salary higher than the average and who do not have any bonus or allowance

CREATE OR REPLACE VIEW lazy\_workers AS

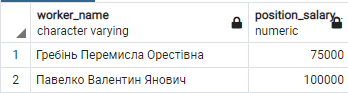
SELECT w.worker\_name, p.position\_salary FROM workers w

JOIN positions p ON p.worker\_id = w.worker\_id,

(SELECT AVG(p.position\_salary) AS average FROM positions p) avg

WHERE p.position\_salary > avg.average

GROUP BY w.worker\_name, p.position\_salary;



4.3 Creation of triggers

* A trigger that sets the deleted value when the workers tuple is deleted worker\_id in other tables as NULL

CREATE OR REPLACE FUNCTION deleting\_worker()

RETURNS trigger

AS $$

BEGIN

UPDATEbonuses SET worker\_id = NULL WHERE worker\_id = old.worker\_id;

UPDATEsurcharges SET worker\_id = NULL WHERE worker\_id = old.worker\_id;

UPDATEillnesses SET worker\_id = NULL WHERE worker\_id = old.worker\_id;

UPDATEworkers\_kids SET worker\_id = NULL WHERE worker\_id = old.worker\_id;

UPDATE positions SET worker\_id = NULL WHERE worker\_id = old.worker\_id;

RETURN old;

END;$$

LANGUAGE 'plpgsql';

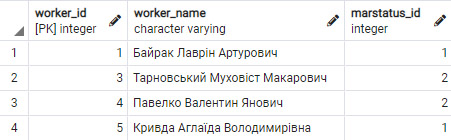
CREATE OR REPLACE TRIGGER deleting\_worker

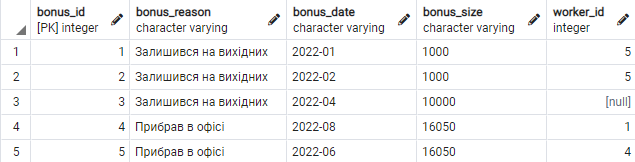
BEFORE DELETE ON workers

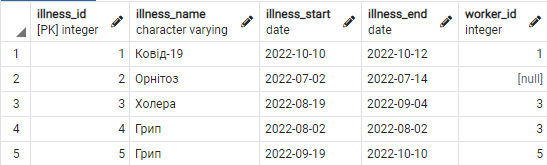
FOR EACH ROW

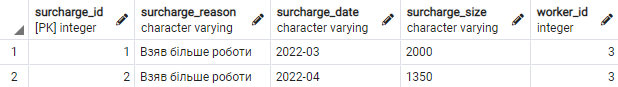
EXECUTE PROCEDURE deleting\_worker();

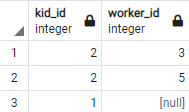
DELETE FROM workers WHERE worker\_id = 2;

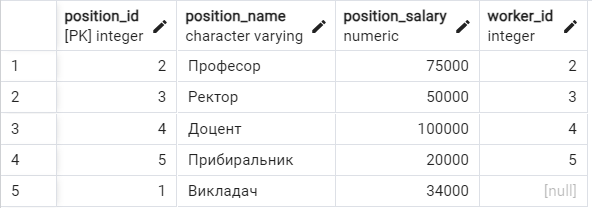












* A trigger that, when adding a workers tuple, sets the added value worker\_id in other tables where it is NULL

CREATE OR REPLACE FUNCTION adding\_worker()

RETURNS trigger

AS $$

BEGIN

UPDATE bonuses SETworker\_id = new.worker\_id WHERE worker\_id IS NULL;

UPDATE surcharges SETworker\_id = new.worker\_id WHERE worker\_id IS NULL;

UPDATE illnesses SETworker\_id = new.worker\_id WHERE worker\_id IS NULL;

UPDATEworkers\_kids SET worker\_id = new.worker\_id WHERE worker\_id IS NULL;

UPDATE positions SETworker\_id = new.worker\_id WHERE worker\_id IS NULL;

RETURN new;

END;$$

LANGUAGE 'plpgsql';

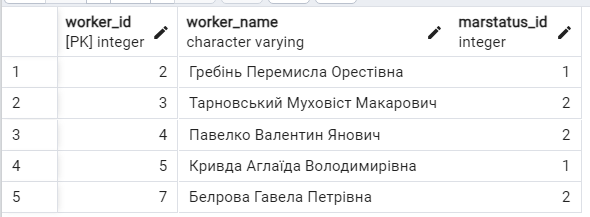
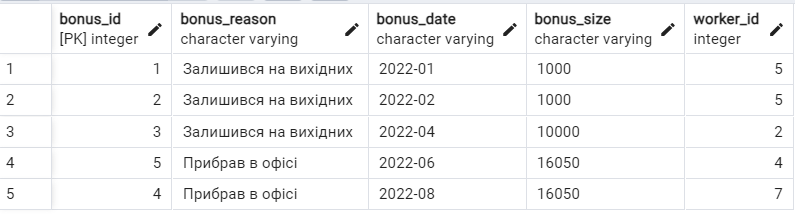
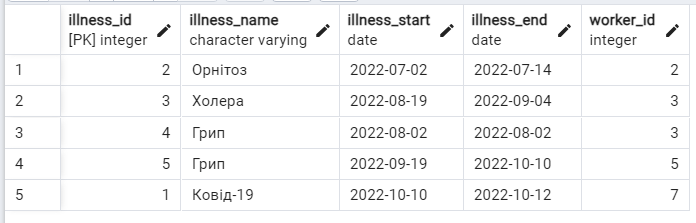
CREATE OR REPLACE TRIGGER adding\_worker

AFTER INSERT ON workers

FOR EACH ROW

EXECUTE PROCEDURE adding\_worker ();

INSERT INTO workers VALUES (7, 'Belrova Havela Petrivna', '2');



* A trigger that when the value is updatedrefund\_amount multiplies the value by 1.5

CREATE OR REPLACE FUNCTION update\_bonus()

RETURNS trigger

AS $$

BEGIN

new.bonus\_size = CAST(new.bonus\_size AS numeric) \* 1.5;

RETURN new;

END;$$

LANGUAGE 'plpgsql';

CREATE OR REPLACE TRIGGER update\_bonus

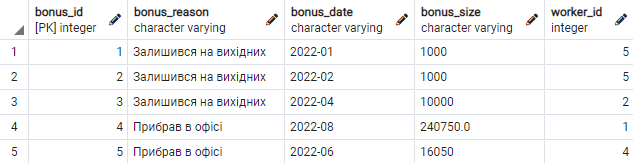
BEFORE UPDATE ON bonuses

FOR EACH ROW

EXECUTE PROCEDURE update\_bonus();

WITH CTE AS (SELECT MIN(worker\_id) AS min\_worker\_id FROM bonuses)

UPDATE bonuses b SETbonus\_size = bonus\_size ::int \* 10 WHERE b.worker\_id = (SELECT min\_worker\_id FROM CTE);



4.4 Creating stored procedures (functions)

* A function that, when called, shows the number of days in a certain month

CREATE OR REPLACE FUNCTION days\_amount (date VARCHAR)

RETURNS INT

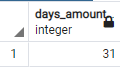
AS $$

BEGIN

RETURN date\_part('days', (date\_trunc('month', to\_date(date, 'YYYY-MM-DD')) + interval '1 month - 1 day')) AS NUMBER;

END;

$$ LANGUAGE 'plpgsql';



* A function that, when called, shows the number of missed days per month due to each illness

CREATE OR REPLACE FUNCTION absent\_days (date VARCHAR)

RETURNS TABLE (

ID\_Diseases INT,

Employee\_ID INT,

Number\_of\_hospitals INT

)

AS $$

BEGIN

RETURN QUERY SELECT i.illness\_id, w.worker\_id,

CASE

WHEN DATE((date\_trunc('month', to\_date(date, 'YYYY-MM-DD')) + INTERVAL '1 month') - INTERVAL '1 day') >= i.illness\_end AND DATE(date\_trunc('month', to\_date(date , 'YYYY-MM-DD'))) <= i.illness\_start THEN i.illness\_end - i.illness\_start + 1

WHEN DATE((date\_trunc('month', to\_date(date, 'YYYY-MM-DD')) + INTERVAL '1 month') - INTERVAL '1 day') >= i.illness\_end AND DATE(date\_trunc('month', to\_date(date , 'YYYY-MM-DD'))) >= i.illness\_start AND DATE(date\_trunc('month', to\_date(date, 'YYYY-MM-DD'))) <= i.illness\_end THEN i.illness\_end - DATE (date\_trunc('month', to\_date(date, 'YYYY-MM-DD'))) + 1

WHEN DATE((date\_trunc('month', to\_date(date, 'YYYY-MM-DD')) + INTERVAL '1 month') - INTERVAL '1 day') <= i.illness\_end AND DATE(date\_trunc('month', to\_date(date , 'YYYY-MM-DD'))) <= i.illness\_start AND DATE((date\_trunc('month', to\_date(date, 'YYYY-MM-DD')) + INTERVAL '1 month') - INTERVAL '1 day') >= i.illness\_start THEN DATE((date\_trunc('month', to\_date(date, 'YYYY-MM-DD')) + INTERVAL '1 month') - INTERVAL '1 day') - i.illness\_start + 1

WHEN DATE((date\_trunc('month', to\_date(date, 'YYYY-MM-DD')) + INTERVAL '1 month') - INTERVAL '1 day') <= i.illness\_end AND DATE(date\_trunc('month', to\_date(date , 'YYYY-MM-DD'))) >= i.illness\_start THEN DATE((date\_trunc('month', to\_date(date, 'YYYY-MM-DD'))) + INTERVAL '1 month') - INTERVAL '1 day') - DATE(date\_trunc('month', to\_date(date, 'YYYY-MM-DD'))) + 1

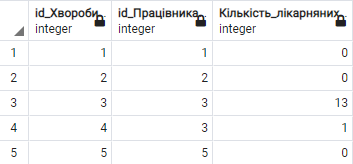
ELSE 0 END ASabsent\_days

FROM workers w, illnesses i

WHERE w.worker\_id = i.worker\_id;

END;

$$ LANGUAGE 'plpgsql';



* A function that, when called, shows the number of missed days per month for each employee

CREATE OR REPLACE FUNCTION summed\_absent\_days (date VARCHAR)

RETURNS TABLE (

Employee\_ID INT,

Total\_number\_of\_hospitals INT

)

AS $$

BEGIN

RETURN QUERY SELECT w.worker\_id,

coalesce(SUM(ad.Number\_of\_hospitals), 0)::INT AS summed\_absent\_days

FROM workers w

LEFT JOIN absent\_days(date) ad

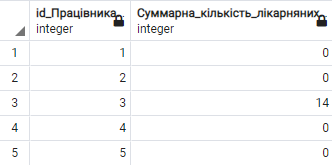
ONw.worker\_id = ad.ID\_Worker

GROUP BY w.worker\_id

ORDER BY w.worker\_id;

END;

$$ LANGUAGE 'plpgsql';



* A function that, when called, shows the accrual for each reason for each employee

CREATE OR REPLACE FUNCTION payment\_statement (date VARCHAR)

RETURNS TABLE (

Employee\_ID INT,

Name\_Employee VARCHAR,

Salary DECIMAL,

Salary\_with\_hospital benefits DECIMAL,

DECIMAL awards,

DECIMAL Salary supplementt

)

AS $$

BEGIN

RETURN QUERY SELECT

w.worker\_id,

w.worker\_name,

p.position\_salary,

ROUND((p.position\_salary - ((p.position\_salary / days\_amount(date) \* 0.5) \* sad.Total\_number\_of\_hospitals)), 0),

coalesce(b.bonus\_size, '0')::numeric,

coalesce(s.surcharge\_size, '0')::numeric

FROM workers w

LEFT JOIN positions p

ONp.worker\_id = w.worker\_id

LEFT JOIN summed\_absent\_days(date) sad

ONw.worker\_id = sad.ID\_Worker

LEFT JOIN bonuses b

ONw.worker\_id = b.worker\_id AND b.bonus\_date = date

LEFT JOIN surcharges s

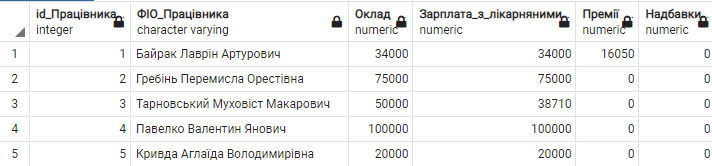
ONw.worker\_id = s.worker\_id AND s.surcharge\_date = date

ORDER BY w.worker\_id

;

END;

$$ LANGUAGE 'plpgsql';



* A function that, when called, shows the accrual for each reason and the total accrual for each employee

CREATE OR REPLACE FUNCTION payment\_statement\_workers (date VARCHAR)

RETURNS TABLE (

Employee\_ID INT,

Name\_Employee VARCHAR,

Salary DECIMAL,

Salary\_with\_hospital benefits DECIMAL,

DECIMAL awards,

DECIMAL Salary supplement,

Total\_Payout DECIMAL

)

AS $$

BEGIN

RETURN QUERY SELECT

ps.ID\_Employee,

ps. Full Name\_Employee,

ps. Salary,

ps. Salary\_with\_hospital benefits,

ps. Awards,

ps. Salary supplementt,

ps.Salary\_with\_hospital + ps.Bonus::numeric + ps.Salary supplementt::numeric AS Total\_Pay

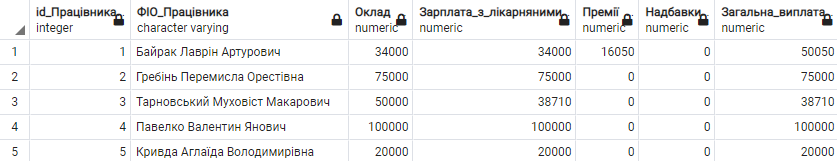
FROM payment\_statement(date) ps

ORDER BY ps.ID\_Employee

;

END;

$$ LANGUAGE 'plpgsql';



* A function that, when called, shows the total charge for each reason

CREATE OR REPLACE FUNCTION payment\_statement\_summary (date VARCHAR)

RETURNS TABLE (

Total\_salary\_with\_hospitals DECIMAL,

Total\_prizes DECIMAL,

Total\_Salary supplementt DECIMAL

)

AS $$

BEGIN

RETURN QUERY SELECT

SUM(ps.Salary\_with\_hospital benefits),

SUM(ps. Prizes),

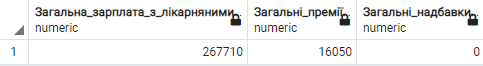
SUM(ps. Salary supplementt)

FROM payment\_statement(date) ps

;

END;

$$ LANGUAGE 'plpgsql';



* A function that, when called, shows the total accrual for each month of the year for each employee

CREATE OR REPLACE FUNCTION payment\_statement\_workers\_summary (date VARCHAR)

RETURNS TABLE (

Employee\_ID INT,

Name\_Employee VARCHAR,

January DECIMAL,

February DECIMAL,

March DECIMAL,

April DECIMAL,

May DECIMAL,

June DECIMAL,

July DECIMAL,

August DECIMAL,

September DECIMAL,

October DECIMAL,

November DECIMAL,

December DECIMAL

)

AS $$

BEGIN

RETURN QUERY SELECT

pswjan.ID\_Employee,

pswjan.FIM\_Employee,

pswjan.total\_payment,

pswfeb.Total\_payout,

pswmar.total\_payment,

pswapr.Total\_payout,

pswmay.total\_payment,

pswjun.Total\_payout,

pswjul.Total\_payment,

pswaug.total\_payment,

pswsep.total\_payment,

pswoct.Total\_payout,

pswnov.Total\_payout,

pswdec.Total\_payment

FROM payment\_statement\_workers(CONCAT(date, '-01')) pswjan

LEFT JOIN payment\_statement\_workers(CONCAT(date, '-02')) pswfeb

ONpswjan.Employee\_ID = pswfeb.Employee\_ID

LEFT JOIN payment\_statement\_workers(CONCAT(date, '-03')) pswmar

ONpswjan.Employee\_ID = pswmar.Employee\_ID

LEFT JOIN payment\_statement\_workers(CONCAT(date, '-04')) pswapr

ONpswjan.Employee\_ID = pswapr.Employee\_ID

LEFT JOIN payment\_statement\_workers(CONCAT(date, '-05')) pswmay

ONpswjan.Employee\_ID = pswmay.Employee\_ID

LEFT JOIN payment\_statement\_workers(CONCAT(date, '-06')) pswjun

ONpswjan.Employee\_ID = pswjun.Employee\_ID

LEFT JOIN payment\_statement\_workers(CONCAT(date, '-07')) pswjul

ONpswjan.Employee\_ID = pswjul.Employee\_ID

LEFT JOIN payment\_statement\_workers(CONCAT(date, '-08')) pswaug

ONpswjan.Employee\_ID = pswaug.Employee\_ID

LEFT JOIN payment\_statement\_workers(CONCAT(date, '-09')) pswsep

ONpswjan.Employee\_ID = pswsep.Employee\_ID

LEFT JOIN payment\_statement\_workers(CONCAT(date, '-10')) pswoct

ONpswjan.Employee\_ID = pswoct.Employee\_ID

LEFT JOIN payment\_statement\_workers(CONCAT(date, '-11')) pswnov

ONpswjan.Employee\_ID = pswnov.Employee\_ID

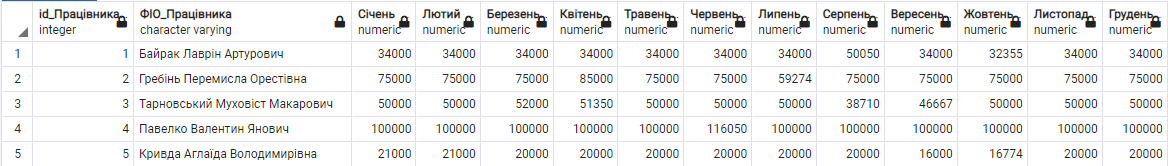
LEFT JOIN payment\_statement\_workers(CONCAT(date, '-12')) pswdec

ONpswjan.Employee\_ID = pswdec.Employee\_ID

;

END;

$$ LANGUAGE 'plpgsql';

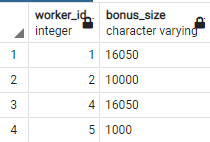


5 DATA MANIPULATION

5.1 Sampling operator

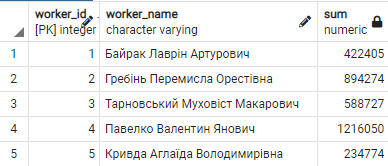
* Output of grouped worker\_id and bonus\_size

SELECT w.worker\_id, b.bonus\_size FROM workers w, bonuses b WHERE w.worker\_id = b.worker\_id GROUP BY w.worker\_id, b.bonus\_size ORDER BY w.worker\_id;



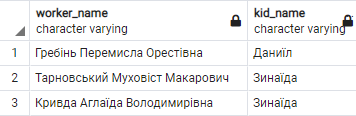
* Output of grouped workers and their total accruals for 2022

SELECT w.worker\_id, w.worker\_name, SUM(ps.January + ps.February + ps.March + ps.April + ps.May + ps.June + ps.July + ps.August + ps.September + ps.October + ps.November + ps.December) FROM workers w, payment\_statement\_workers\_summary('2022') ps WHERE w.worker\_id = ps.Worker\_ID GROUP BY w.worker\_id, w.worker\_name ORDER BY w.worker\_id;



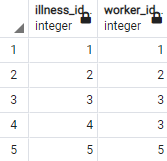
* Output of the relationship worker\_name and kid\_name

SELECT w.worker\_name, k.kid\_name FROM kids k, workers w, workers\_kids wk WHERE wk.kid\_id = k.kid\_id AND wk.worker\_id = w.worker\_id;



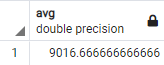
* Output of the relationship between illness\_id and worker\_id

SELECT i.illness\_id, w.worker\_id FROM workers w, illnesses i WHERE w.worker\_id = i.worker\_id;



* Derivation of the average premium

SELECT AVG(DISTINCT b.bonus\_size::float) FROM bonuses b;



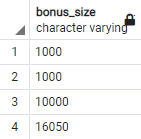
* Withdrawal of the minimum premium

SELECT MIN(b.bonus\_size) FROM bonuses b;



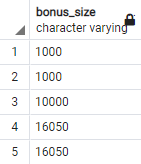
* Output all refund\_amount values ​​where worker\_id is greater than minimum from tuple where worker\_id is 1

SELECT b.bonus\_size FROM bonuses b WHERE b.worker\_id > ANY(SELECT w.worker\_id FROM workers w WHERE worker\_id = 1);



* Output of all refund\_amount values ​​in a tuple where worker\_id matches refunds and workers using subqueries

SELECT b.bonus\_size FROM bonuses b WHERE b.worker\_id IN (SELECT b.worker\_id FROM workers w WHERE b.worker\_id = w.worker\_id);



6 CREATION OF USERS AND ASSIGNMENT OF ACCESS RIGHTS

* Creating a worker who can view information where only his id

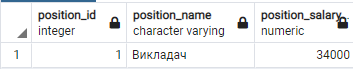
CREATE USER worker1;

CREATE VIEW CTE AS SELECT p.\* FROM workers w, positions p WHERE w.worker\_id = 1 AND w.position\_id = p.position\_id;

GRANT SELECT ON CTE TO worker1;

SET ROLE 'worker1';

SELECT \* FROM CTE;



* Creating a worker who can do everything

CREATE USER admin1;

GRANT ALL ON ALL TABLES IN SCHEMA public TO admin1;

SELECT \* FROM summed\_absent\_days('2022-08');



CONCLUSIONS

In the course of this coursework, the subject area "University Accounting" was researched and a database was developed that displays lists of charges in universities. This database can be used as a basis for creating an information system of real universities or for educational purposes. The skills of designing a database, deriving the necessary information structure and grouping data according to a set of essential features from the point of view of minimizing information storage costs were acquired. The use of the basic features of the postgreSQL programming language, such as triggers, views, and stored procedures, was established in practice.

LIST OF USED SOURCES

1. Head M.G. Methodical instructions for completing the course work from the course "Organization of databases and knowledge" for students of all forms of education in the specialty 122 "Computer science" / Acc.: M.G. Chapter. - Odesa, 2022. - 27 p. – Access mode: http://library.opu.ua.
2. Malakhov E.V., Blazhko O.A., Hlava M.G. Database design and their implementation using standard SQL and PostgreSQL: Training. study guide higher education institutions - O.: VMV, 2012. - 248 p.
3. Head M.G. Organization of databases and knowledge: Synopsis of lectures. [Electronic resource] - Access mode: http://library.opu.ua.
4. Head M.G. Methodical instructions for performing laboratory work in the discipline "Organization of databases and knowledge". - Access mode: http://library.opu.ua.
5. Head M.G. Methodical instructions for independent work in the discipline "Organization of databases and knowledge". – Access mode: http://library.opu.ua.