PROJEKT

WPROWADZENIE DO BAZ DANYCH

Dawid Iwański
49321

P R Z E Z N A C Z E N I E

Baza danych została zaprojektowana w celu zarządzania informacjami o pracownikach, firmach, fakturach, rekordach czasu pracy oraz innymi istotnymi danymi w firmie. Umożliwia kompleksowe zarządzanie danymi związanymi z pracownikami, ich uprawnieniami, fakturami, rekordami czasu pracy, logami aktywności oraz rekordami urlopów. System ten jest szczególnie przydatny dla firm, które potrzebują śledzić i zarządzać różnorodnymi informacjami dotyczącymi zasobów ludzkich i finansów.

STRUKTURA

Employees

Przechowuje informacje o pracownikach, takie jak ID pracownika, ID firmy, imię, adres, stawka godzinowa, e-mail oraz hasło.

Company

Zawiera dane dotyczące firm, w tym nazwę firmy, maksymalną liczbę pracowników oraz cenę.

Permissions

Przechowuje informacje o uprawnieniach pracowników, takich jak czy pracownik jest administratorem, czy może edytować czas pracy oraz kategorie.

Invoices

Zawiera informacje o fakturach, w tym ID faktury, ID kategorii, URL faktury oraz datę faktury.

Time_Records

Przechowuje dane o rekordach czasu pracy pracowników, w tym ID rekordu, ID pracownika, ID kategorii, czas rozpoczęcia, czas zakończenia oraz status.

Logs

Przechowuje logi aktywności pracowników, takie jak ID logu, ID pracownika, treść logu oraz data logu.

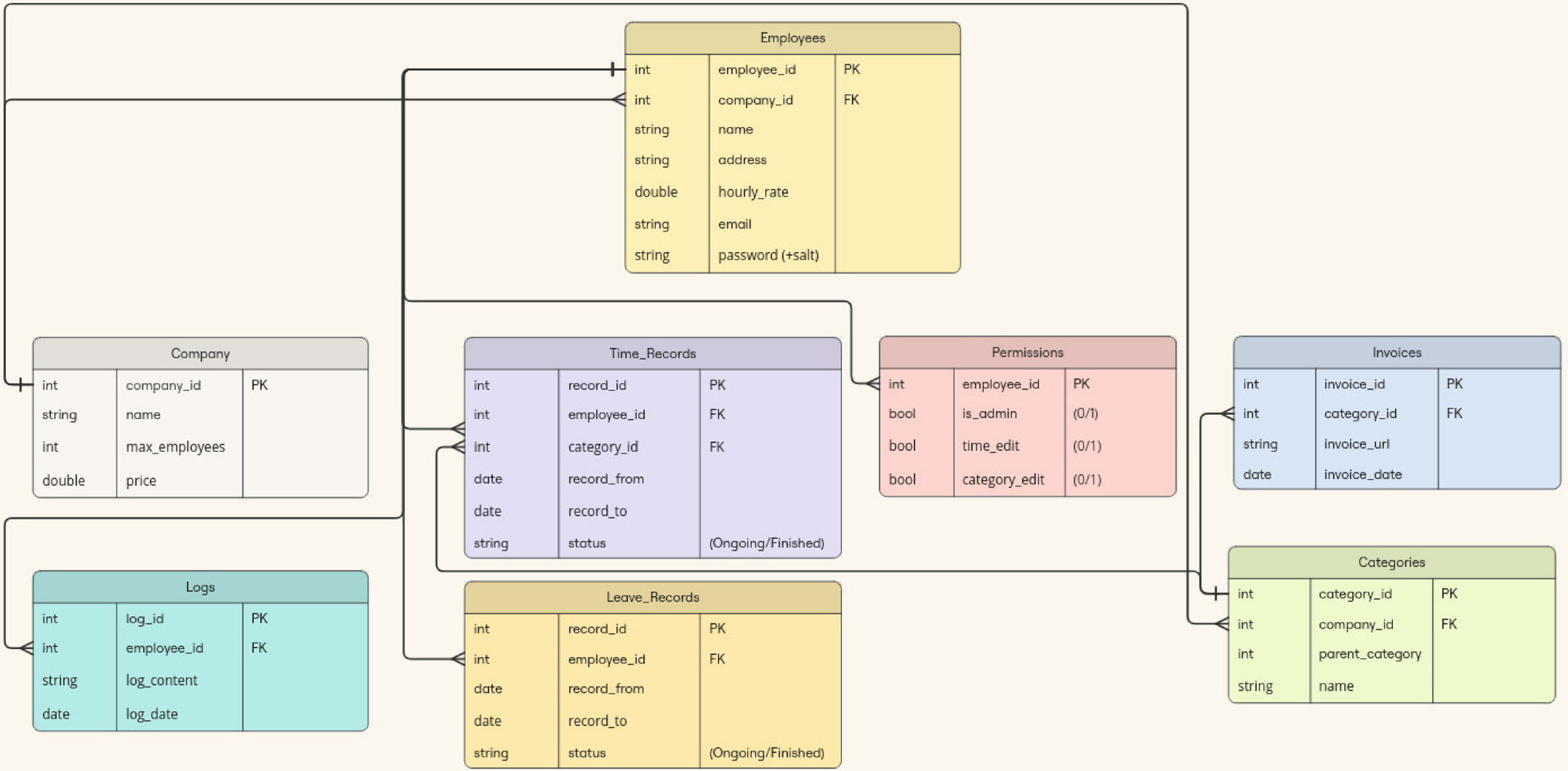
Leave_Records

Przechowuje informacje o rekordach urlopów pracowników, w tym ID rekordu, ID pracownika, data rozpoczęcia, data zakończenia oraz status.

Categories

Zawiera dane o kategoriach, w tym ID kategorii, ID firmy, ID nadrzędnej kategorii oraz nazwę kategorii.

DIAGRAM ERD


Employees
int employee_id PK
int company_id FK
string name
string address
double hourly_rate
string email
string password (+salt)
Company
int company_id PK
string name
int max_employees
double price
Time_Records
int record_id PK
int employee_id FK
int category_id FK
date record_from
date record_to
string status (Ongoing/Finished)
Permissions
int employee_id PK
bool is_admin (0/1)
bool time_edit (0/1)
bool category_edit (0/1)
Invoices
int invoice_id PK
int category_id FK
string invoice_url
date invoice_date
Logs
int log_id PK
int employee_id FK
string log_content
date log_date
Leave_Records
int record_id PK
int employee_id FK
date record_from
date record_to
string status (Ongoing/Finished)
Categories
int category_id PK
int company_id FK
int parent_category
string name

ENCJE

PREZENTACJA ENCJI - EMPLOYEES

```sql
CREATE TABLE Employees (
    employee_id SERIAL PRIMARY KEY,
    company_id INT,
    name VARCHAR,
    address VARCHAR,
    hourly_rate DOUBLE PRECISION,
    email VARCHAR,
    password_salt VARCHAR
);
```

	employee_id [PK] integer	company_id integer	name character varying	address character varying	hourly_rate double precision	email character varying	password_salt character varying
1	1	1	John Doe	123 Main St	20.5	john@example.com	abc123
2	2	1	Jane Smith	456 Elm St	25	jane@example.com	xyz789
3	3	2	Bob Johnson	789 Oak St	18.75	bob@example.com	def456

PREZENTACJA ENCJI - COMPANY

```sql
CREATE TABLE Company (
    company_id SERIAL PRIMARY KEY,
    name VARCHAR,
    max_employees INT,
    price DOUBLE PRECISION
);
```

	company_id [PK] integer	name character varying	max_employees integer	price double precision
1	1	ABC Corp	100	5000
2	2	XYZ Inc	50	3000

PREZENTACJA ENCJI - PERMISSIONS

```sql
CREATE TABLE Permissions (
    employee_id INT,
    is_admin BOOLEAN,
    time_edit BOOLEAN,
    category_edit BOOLEAN,
    FOREIGN KEY (employee_id) REFERENCES Employees(employee_id)
);
```

	employee_id integer	is_admin boolean	time_edit boolean	category_edit boolean
1	1	true	true	true
2	2	false	true	false
3	3	false	true	true

PREZENTACJA ENCJI - INVOICES

```sql
CREATE TABLE Invoices (
    invoice_id SERIAL PRIMARY KEY,
    category_id INT,
    invoice_url VARCHAR,
    invoice_date DATE
);
```

	invoice_id [PK] integer	category_id integer	invoice_url character varying	invoice_date date
1	1	1	http://example.com/in...	2024-05-01
2	2	2	http://example.com/in...	2024-05-05

PREZENTACJA ENCJI - TIME_RECORDS

```sql
CREATE TABLE Time_Records (
    record_id SERIAL PRIMARY KEY,
    employee_id INT,
    category_id INT,
    record_from DATE,
    record_to DATE,
    status VARCHAR,
    FOREIGN KEY (employee_id) REFERENCES Employees(employee_id)
);
```

	record_id [PK] integer	employee_id integer	category_id integer	record_from date	record_to date	status character varying
1	1	1	1	2024-05-01	2024-05-02	approved
2	2	2	2	2024-05-03	2024-05-04	pending

PREZENTACJA ENCJI - LOGS

```sql
CREATE TABLE Logs (
    log_id SERIAL PRIMARY KEY,
    employee_id INT,
    log_content VARCHAR,
    log_date DATE,
    FOREIGN KEY (employee_id) REFERENCES Employees(employee_id)
);
```

	log_id [PK] integer	employee_id integer	log_content character varying	log_date date
1	1	1	Logged in	2024-05-01
2	2	2	Logged out	2024-05-02

PREZENTACJA ENCJI - LEAVE_RECORDS

```sql
CREATE TABLE Leave_Records (
    record_id SERIAL PRIMARY KEY,
    employee_id INT,
    record_from DATE,
    record_to DATE,
    status VARCHAR,
    FOREIGN KEY (employee_id) REFERENCES Employees(employee_id)
);
```

	record_id [PK] integer	employee_id integer	record_from date	record_to date	status character varying
1	1	1	2024-05-10	2024-05-12	approved
2	2	3	2024-05-15	2024-05-17	pending

PREZENTACJA ENCJI - CATEGORIES

```
60  CREATE TABLE Categories (
61      category_id SERIAL PRIMARY KEY,
62      company_id INT,
63      parent_category INT,
64      name VARCHAR,
65      FOREIGN KEY (company_id) REFERENCES Company(company_id),
66      FOREIGN KEY (parent_category) REFERENCES Categories(category_id)
67  );
```

	category_id [PK] integer	company_id integer	parent_category integer	name character varying
1	1	1	[null]	Category A
2	2	1	1	Subcategory A1
3	3	2	[null]	Category B

WIDOKI

PREZENTACJA WIDOKÓW - EMPLOYEE_DETAILS

```sql
CREATE OR REPLACE VIEW Employee_Details AS
SELECT
    e.employee_id,
    e.name AS employee_name,
    e.address,
    e.hourly_rate,
    e.email,
    e.password_salt,
    c.name AS company_name,
    p.is_admin,
    p.time_edit,
    p.category_edit
FROM
    Employees e
JOIN
    Company c ON e.company_id = c.company_id
LEFT JOIN
    Permissions p ON e.employee_id = p.employee_id;

SELECT * FROM Employee_Details;
```

	employee_id integer	employee_name character varying	address character varying	hourly_rate double precision	email character varying	password_salt character varying	company_name character varying	is_admin boolean	time_edit boolean	category_edit boolean
1	1	John Doe	123 Main St	20.5	john@example.com	abc123	ABC Corp	true	true	true
2	2	Jane Smith	456 Elm St	25	jane@example.com	xyz789	ABC Corp	false	true	false
3	3	Bob Johnson	789 Oak St	18.75	bob@example.com	def456	XYZ Inc	false	true	true

PREZENTACJA WIDOKÓW - EMPLOYEE_TIME_RECORDS

```sql
CREATE OR REPLACE VIEW Employee_Time_Records AS
SELECT
    e.employee_id,
    e.name AS employee_name,
    e.email,
    c.name AS company_name,
    t.record_id,
    t.record_from,
    t.record_to,
    t.status,
    cat.name AS category_name
FROM
    Employees e
JOIN
    Company c ON e.company_id = c.company_id
JOIN
    Time_Records t ON e.employee_id = t.employee_id
JOIN
    Categories cat ON t.category_id = cat.category_id;

SELECT * FROM Employee_Time_Records;
```

	employee_id integer	employee_name character varying	email character varying	company_name character varying	record_id integer	record_from date	record_to date	status character varying	category_name character varying
1	1	John Doe	john@example.com	ABC Corp	1	2024-05-01	2024-05-02	approved	Category A
2	2	Jane Smith	jane@example.com	ABC Corp	2	2024-05-03	2024-05-04	pending	Subcategory A1

PREZENTACJA WIDOKÓW - EMPLOYEE_TIME_RECORDS

```sql
CREATE OR REPLACE VIEW Employee_Invoice_Summary AS
SELECT
    e.employee_id,
    e.name AS employee_name,
    e.email,
    c.name AS company_name,
    inv.invoice_id,
    inv.invoice_url,
    inv.invoice_date,
    cat.name AS category_name
FROM
    Employees e
JOIN
    Company c ON e.company_id = c.company_id
JOIN
    Categories cat ON e.company_id = cat.company_id
JOIN
    Invoices inv ON inv.category_id = cat.category_id
ORDER BY
    inv.invoice_date DESC;

SELECT * FROM Employee_Invoice_Summary;
```

	employee_id integer	employee_name character varying	email character varying	company_name character varying	invoice_id integer	invoice_url character varying	invoice_date date	category_name character varying
1	2	Jane Smith	jane@example.com	ABC Corp	2	http://example.com/in...	2024-05-05	Subcategory A1
2	1	John Doe	john@example.com	ABC Corp	2	http://example.com/in...	2024-05-05	Subcategory A1
3	2	Jane Smith	jane@example.com	ABC Corp	1	http://example.com/in...	2024-05-01	Category A
4	1	John Doe	john@example.com	ABC Corp	1	http://example.com/in...	2024-05-01	Category A

FUNKCJE

PREZENTACJA FUNKCJI - GETALLSUBCATEGORIES

```
199  CREATE OR REPLACE FUNCTION GetAllSubcategories(category_id INT)
200  RETURNS TABLE (sub_category_id INT, sub_category_name VARCHAR) AS $$
201  WITH RECURSIVE subcategories AS (
202      SELECT category_id, name
203      FROM Categories
204      WHERE category_id = $1
205      UNION ALL
206      SELECT c.category_id, c.name
207      FROM Categories c
208      INNER JOIN subcategories s ON c.parent_category = s.category_id
209  )
210  SELECT category_id, name FROM subcategories;
211  $$ LANGUAGE sql;
212
213  -- Pobierz wszystkie subkategorie
214  SELECT * FROM GetAllSubcategories(1);
```

	sub_category_id integer	sub_category_name character varying
1	1	Category A
2	2	Subcategory A1

PREZENTACJA FUNKCJI - CALCULATETOTALHOURS

```
216 CREATE OR REPLACE FUNCTION CalculateTotalHours(emp_id INT, start_date DATE, end_date DATE)
217 RETURNS DOUBLE PRECISION AS $$
218 DECLARE
219     total_hours DOUBLE PRECISION := 0;
220 BEGIN
221     SELECT COALESCE(SUM(EXTRACT(EPOCH FROM (record_to::timestamp - record_from::timestamp))/3600), 0)
222     INTO total_hours
223     FROM Time_Records
224     WHERE Time_Records.employee_id = emp_id
225     AND record_from >= start_date
226     AND record_to <= end_date;
227
228     RETURN total_hours;
229 END;
230 $$ LANGUAGE plpgsql;
231
232 -- Podlicz godziny przepracowane przez pracownika
233 SELECT CalculateTotalHours(1, '2024-05-01', '2024-05-10');
```

	calculatetotalhours double precision
1	24

PREZENTACJA FUNKCJI - GENERATEINVOICEREPORT

```sql
CREATE OR REPLACE FUNCTION GenerateInvoiceReport(comp_id INT, start_date DATE, end_date DATE)
RETURNS TABLE (invoice_id INT, invoice_url VARCHAR, invoice_date DATE, category_name VARCHAR) AS $$
BEGIN
    RETURN QUERY
    SELECT
        inv.invoice_id,
        inv.invoice_url,
        inv.invoice_date,
        cat.name AS category_name
    FROM
        Invoices inv
    JOIN
        Categories cat ON inv.category_id = cat.category_id
    WHERE
        cat.company_id = comp_id
        AND inv.invoice_date BETWEEN start_date AND end_date
    ORDER BY
        inv.invoice_date;
END;
$$ LANGUAGE plpgsql;

-- Lista faktur pomiedzy tymi datami
SELECT * FROM GenerateInvoiceReport(1, '2024-05-01', '2024-05-31');
```

	invoice_id integer	invoice_url character varying	invoice_date date	category_name character varying
1	1	http://example.com/in...	2024-05-01	Category A
2	2	http://example.com/in...	2024-05-05	Subcategory A1

WYZWALACZE

PREZENTACJA WYZWALACZY

```
262  CREATE OR REPLACE FUNCTION UpdateLeaveRecordStatus()
263  RETURNS TRIGGER AS $$
264  BEGIN
265      IF NEW.record_from < CURRENT_DATE AND NEW.status = 'pending' THEN
266          NEW.status := 'expired';
267      END IF;
268      RETURN NEW;
269  END;
270  $$ LANGUAGE plpgsql;
271
272  CREATE TRIGGER LeaveRecordStatusTrigger
273  BEFORE INSERT OR UPDATE ON Leave_Records
274  FOR EACH ROW EXECUTE FUNCTION UpdateLeaveRecordStatus();
```

	record_id [PK] integer	employee_id integer	record_from date	record_to date	status character varying
1	1	1	2024-05-10	2024-05-12	approved
2	2	3	2024-05-15	2024-05-17	pending
3	4	3	2024-04-22	2024-05-01	expired

PREZENTACJA WYZWALACZY

```
-- Ten wyzwalacz sprawdza, czy liczba pracowników w firmie nie przekracza maksymalnej liczby pracowników określonej dla tej firmy.
CREATE OR REPLACE FUNCTION CheckMaxEmployees()
RETURNS TRIGGER AS $$
DECLARE
    current_employee_count INT;
BEGIN
    SELECT COUNT(*) INTO current_employee_count
    FROM Employees
    WHERE company_id = NEW.company_id;

    IF TG_OP = 'INSERT' AND current_employee_count >= (SELECT max_employees FROM Company WHERE company_id = NEW.company_id) THEN
        RAISE EXCEPTION 'Cannot add more employees. Maximum limit reached for this company.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER MaxEmployeesTrigger
BEFORE INSERT ON Employees
FOR EACH ROW EXECUTE FUNCTION CheckMaxEmployees();
```

```
ERROR:  Cannot add more employees. Maximum limit reached for this company.
CONTEXT:  funkcja PL/pgSQL checkmaxemployees(), wiersz 10 w RAISE
SQL state: P0001
```

PREZENTACJA WYZWALACZY

```
304  -- Pomaga przy usuwaniu pracowników z wszystkich rekordów
305  CREATE OR REPLACE FUNCTION CascadeDeleteEmployee()
306  RETURNS TRIGGER AS $$
307  BEGIN
308      -- Usuwanie rekordów z tabeli Time_Records
309      DELETE FROM Time_Records WHERE employee_id = OLD.employee_id;
310
311      -- Usuwanie rekordów z tabeli Leave_Records
312      DELETE FROM Leave_Records WHERE employee_id = OLD.employee_id;
313
314      -- Usuwanie rekordów z tabeli Logs
315      DELETE FROM Logs WHERE employee_id = OLD.employee_id;
316
317      -- Usuwanie rekordów z tabeli Permissions
318      DELETE FROM Permissions WHERE employee_id = OLD.employee_id;
319
320      RETURN OLD;
321  END;
322  $$ LANGUAGE plpgsql;
323
324  CREATE TRIGGER cascade_delete_employee
325  BEFORE DELETE ON Employees
326  FOR EACH ROW
327  EXECUTE FUNCTION CascadeDeleteEmployee();
```

Wyzwalacz przy usuwaniu użytkownika usuwa wszystkie Recordy, Logi i Premissje z nim związane.