

Projekt Bazy Danych II

Aplikacja dla wypożyczalni sprzętu budowlanego

Autorzy: Szymon Banyś, Piotr Bosak

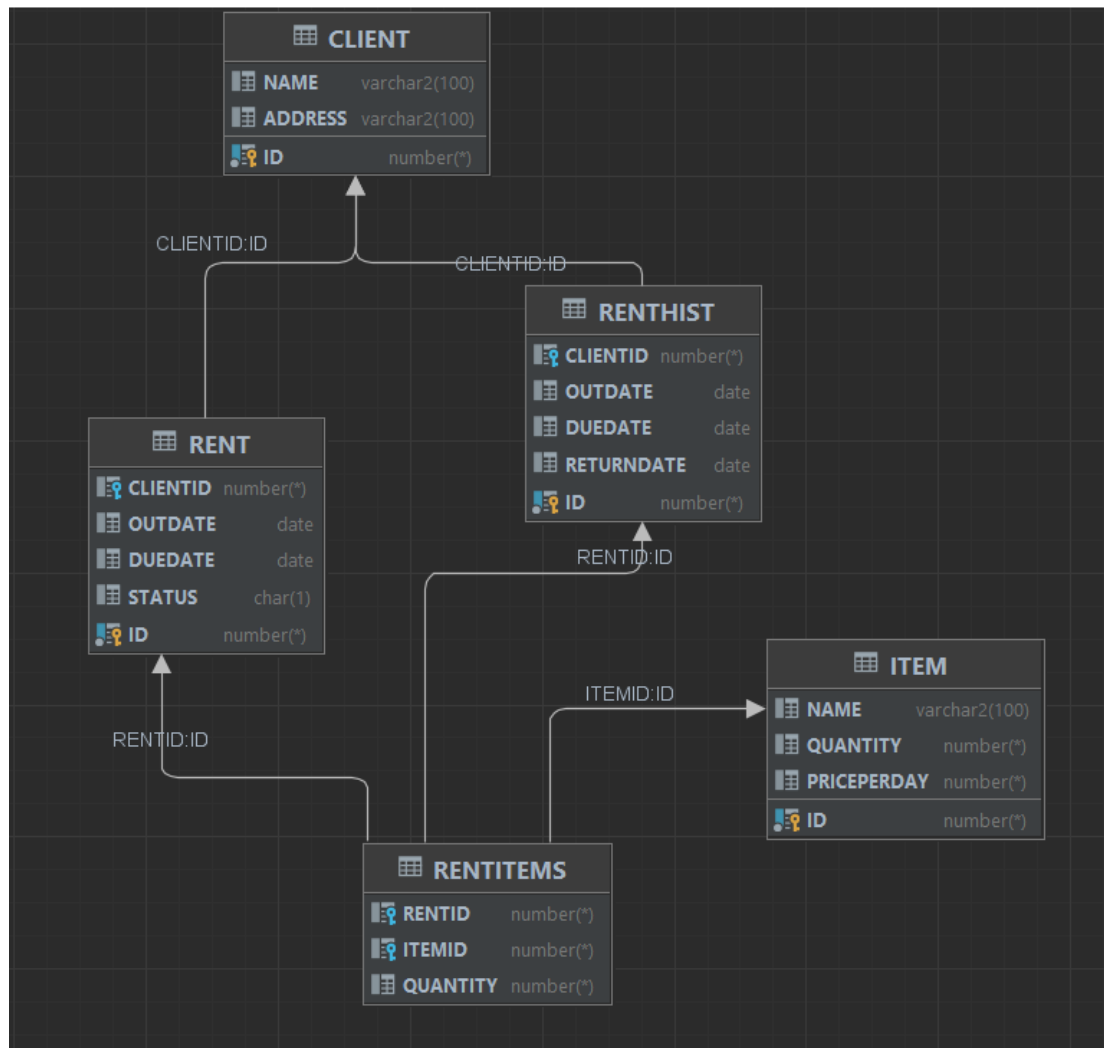
1. Wstęp

Aplikacja korzystająca z bazy danych umożliwiająca rezerwowanie i wypożyczanie sprzętu budowlanego.

2. Technologie

- Python, biblioteki: PySimpleGui, cx_Oracle
- Oracle

3. Schemat Bazy Danych



Schemat 1. Schemat bazy danych.

4. Tworzenie tabel i widoków

```
create table rentHist(  
    id int not null ,  
    clientID int,  
    outDate date,  
    dueDate date,  
    returnDate date,  
    constraint rentHist_pk primary key ( id ) enable  
);  
  
create table rentItems(  
    rentId int,  
    itemId int,  
    quantity int  
);  
  
create table client(  
    id int generated always as identity not null,  
    name varchar(100),  
    address varchar(100),  
    constraint client_pk primary key (id) enable  
);  
  
alter table rentItems  
add constraint rentItems_fk1 foreign key  
(itemId) references item (id) enable;  
  
alter table rentItems  
add constraint rentItems_fk2 foreign key  
(rentId) references rent (id) enable;  
  
alter table rentItems  
add constraint rentItems_fk3 foreign key  
(rentId) references rentHist (id) ;  
  
alter table rentHist  
add constraint rentHist_fk2 foreign key  
(clientId) references client (id) enable;  
  
alter table rentItems  
add price int;  
  
alter table rent  
add constraint rent_chk1 check  
(status in ('R','P','C','D')) enable;
```

```

create or replace view BD_406248.RENTEDITEMS as
select RENTID,ITEMID,name,price,CLIENTID,OUTDATE,DUEDATE,RENTITEMS.QUANTITY,STATUS from RENTITEMS
join rent on RENTITEMS.RENTID = RENT.ID
join item on item.id = RENTITEMS.ITEMID
/

```

5. Funkcjonalności:

1. Wyświetlanie dowolnej tabeli

The screenshot shows a web application interface with a dark background. At the top, it says "Użytkownik: Niezalogowano". Below this is a login form with a text input field and a "Zaloguj" button. Below the login form is a red-bordered box containing a label "Podaj tabelę którą chcesz wyświetlić:", a dropdown menu, and an "Ok" button. The dropdown menu is open, showing a list of table names: "client", "rent", "renthist", "rentitems", and "item". The "rent" option is currently selected and highlighted in yellow.

Rys 1. Widok w aplikacji

```

def select_from_table(self, table_name: str):
    try:
        self._cursor.execute(f"select * from {table_name}")
        data = self._cursor.fetchall()
        new_data = [[] for _ in range(len(data))]

        col_names = [row[0] for row in self._cursor.description]

        if table_name in ['rent', 'renthist']:
            for i in range(len(data)):
                for thing in data[i]:
                    if type(thing) == datetime:
                        new_data[i].append(thing.strftime("%Y-%m-%d"))
                        continue
                    new_data[i].append(thing)
        else:
            new_data = data
            new_data.sort(key=lambda x: x[0])
            new_data.insert(0, col_names)
            return new_data
    except cx_Oracle.DatabaseError:
        return []

```

Rys 2. Implementacja w Pythoni-e

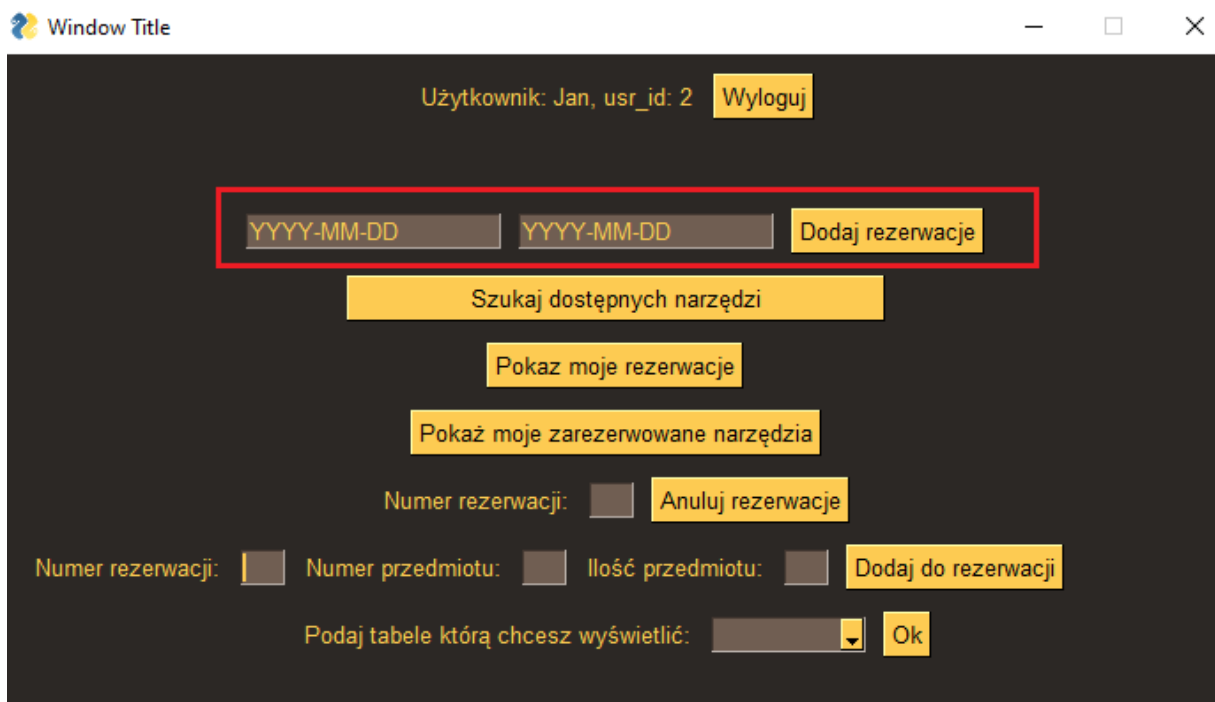
2. Logowanie



Rys 3. Widok logowania w aplikacji

Po uruchomieniu programu możemy zalogować się jako jeden z użytkowników dostępnych w bazie danych (Wszyscy dostępni użytkownicy są pobierani przy starcie aplikacji z bazy danych).

3. Dodawanie rezerwacji



Rys 4. Widok dodawania rezerwacji w aplikacji

W podanych polach należy wpisać odpowiednio od lewej: datę początku rezerwacji, datę końca rezerwacji, a następnie zaakceptować przyciskiem 'Dodaj rezerwację'.

```
def add_rent(self, client_id, date_from, date_to, status="R"):
    self._cursor.execute(f"begin\
        addrent({client_id}, '{date_from}', '{date_to}', '{status}');\
    end;")
```

Rys 5. Implementacja w Pythonie

```
create procedure AddRent(client_id int,date_from date,date_to date, my_status char)
as begin
    insert into RENT(clientid, outdate, duedate, status) values(client_id,date_from,date_to,my_status);
    commit;
end;
```

Rys 6. Implementacja w bazie danych

4. Anulowanie rezerwacji

Użytkownik: Jan, usr_id: 2 Wyloguj

YYYY-MM-DD YYYY-MM-DD Dodaj rezerwacje

Szukaj dostępnych narzędzi

Pokaż moje rezerwacje

Pokaż moje zarezerwowane narzędzia

Numer rezerwacji: Anuluj rezerwacje

Numer rezerwacji: Numer przedmiotu: Ilość przedmiotu: Dodaj do rezerwacji

Podaj tabele którą chcesz wyświetlić: Ok

Rys 7. Widok w aplikacji

```
def cancel_reservation(self, rent_id, status="C"):
    self._cursor.execute(f"begin\
                           modifystatus({rent_id}, '{status}');\
                           end;")
```

Rys 8. Implementacja w Pythoni-e

```
create procedure ModifyStatus(rent_id int, new_status char)
as begin
    update RENT
        set STATUS = new_status
        where rent_id = id;
    commit;
end;
```

Rys 9. Implementacja w bazie danych

5. Wyświetlanie produktów dostępnych w danym okresie

Użytkownik: Jan, usr_id: 2 Wyloguj

Dodaj rezerwacje

Szukaj dostępnych narzędzi

Pokaż moje rezerwacje

Pokaż moje zarezerwowane narzędzia

Numer rezerwacji: Anuluj rezerwacje

Numer rezerwacji: Numer przedmiotu: Ilość przedmiotu: Dodaj do rezerwacji

Podaj tabelę którą chcesz wyświetlić: Ok

Rys 10. Widok w aplikacji

```
def get_available_items(self, date_from, date_to):
    result = self._cursor.execute(f"select * from AVAILABLEITEMS('{date_from}', '{date_to}')" )
    data = result.fetchall()
    data.sort(key=lambda x: x[0])
    col_names = [row[0] for row in self._cursor.description]
    data.insert(0, col_names)
    return data
```

Rys 11. Implementacja w Pythoni-e

```
create function AvailableItems(date_from date, date_to date) return item_table
as
    result item_table;
begin
    select item_type(id,name,QUANTITYOFAVAILABLEITEMS( ITEM_ID: id, DATE_FROM: date_from, DATE_TO: date_to),pricePerDay) bulk collect
    into result
    from ITEM
    where QUANTITYOFAVAILABLEITEMS( ITEM_ID: id, DATE_FROM: date_from, DATE_TO: date_to) > 0;
    return result;
end;
```

```
create function QuantityOfAvailableItems(item_id int, date_from date, date_to date)
return int
as
    result int;
    my_quantity int;
begin
    select QUANTITY into my_quantity from ITEM where ID=item_id;

    select sum(QUANTITY)
    into result
    from RentedItems
    where ITEMID = item_id and not (date_from > RentedItems.DUEDATE or date_to < RentedItems.OUTDATE );
    if result is null then result := 0;
    end if;
    return (my_quantity - result);
end;
```

Rys 12. Implementacja w bazie danych

6. Dodawanie produktu do rezerwacji

The screenshot shows a web application window titled "Window Title". At the top, it displays "Użytkownik: Jan, usr_id: 2" and a "Wyloguj" button. Below this are two date input fields labeled "YYYY-MM-DD" and a "Dodaj rezerwację" button. A large yellow button labeled "Szukaj dostępnych narzędzi" is centered. Below it are two more yellow buttons: "Pokaż moje rezerwacje" and "Pokaż moje zarezerwowane narzędzia". Further down is a "Numer rezerwacji:" label with an input field and an "Anuluj rezerwację" button. At the bottom, a red rectangle highlights a row containing "Numer rezerwacji:" with an input field, "Numer przedmiotu:" with an input field, "Ilość przedmiotu:" with an input field, and a "Dodaj do rezerwacji" button. Below this row is a "Podaj tabelę którą chcesz wyświetlić:" label with a dropdown menu and an "Ok" button.

Rys 13. Widok w aplikacji

Aby dodać produkt do rezerwacji należy pierw wyszukać dostępne narzędzia za pomocą funkcjonalności z ppkt 4.5, a następnie w odpowiednie pola wpisać numer rezerwacji do której chcemy dodać dane narzędzie, ID narzędzia oraz ilość jaka będzie nam potrzebna. W przypadku sukcesu wyświetli się okienko informujące o pomyślnym dodanie produktu do rezerwacji. W innym przypadku wyskoczy okienko informujące o błędach.

The screenshot shows the same application window as in Rys 13, but with a modal dialog box open in the center. The dialog box has a title bar "Pomyślnie dodano ..." and contains the text "Pomyślnie dodano przedmiot do rezerwacji!" and an "OK" button. In the background, the "Numer rezerwacji:" input field now contains the value "122". The "Dodaj do rezerwacji" button is still visible to the right of the dialog box.

Rys 14. Widok poprawnego dodania produktu do rezerwacji

```
def add_item_to_reservation(self, rent_id, item_id, item_quantity):
    try:
        price = self._cursor.execute(f"select priceperday from item where ID = '{item_id}'")
        add_price = price.fetchone()[0]
        self._cursor.execute(f"begin\
                                additemtorent('{rent_id}', '{item_id}', '{item_quantity}', '{add_price}');\
                                end;")
        return False
    except cx_Oracle.DatabaseError as e:
        return True
    except TypeError:
        return True
```

Rys 15. Implementacja w Pythoni-e

```
create procedure AddItemToRent(rent_id int,item_id int,quantity int, my_price int)
as
    rent_exist int;
    item_exist int;
    date_from date;
    date_to date;
begin
    select count(*) into rent_exist from rent where id = rent_id;
    if rent_exist <=0 then raise_application_error(-20000,'nie ma takiego wypożyczenia');
    end if;

    select count(*) into item_exist from ITEM where item_id = id;
    if item_exist <=0 then raise_application_error(-20001,'nie ma takiego itemu');
    end if;

    select OUTDATE into date_from from RENT where rent_id = id;
    select DUEDATE into date_to from Rent where rent_id = id;

    if QuantityOfAvailableItems( ITEM_ID: item_id, DATE_FROM: date_from, DATE_TO: date_to) < quantity then
        raise_application_error(-20002,'za mało dostępnych przedmiotów');
    end if;

    insert into RENTITEMS(rentid, itemid, quantity, price) VALUES (rent_id,item_id,quantity,my_price);
    commit;
end;
```

Rys 16. Implementacja w bazie danych

7. Wyświetlanie naszych rezerwacji

Window Title

Użytkownik: Jan, usr_id: 2 Wyloguj

YYYY-MM-DD YYYY-MM-DD Dodaj rezerwacje

Szukaj dostępnych narzędzi

Pokaz moje rezerwacje

Pokaż moje zarezerwowane narzędzia

Numer rezerwacji: Anuluj rezerwacje

Numer rezerwacji: Numer przedmiotu: Ilość przedmiotu: Dodaj do rezerwacji

Podaj tabele którą chcesz wyświetlić: Ok

Rys 17. Widok w aplikacji


```
def show_my_reservations(self, client_id):
    result = self._cursor.execute(f"select * from myreservations('{client_id}')"
    data = result.fetchall()
    col_names = [row[0] for row in self._cursor.description]

    new_data = [[] for _ in range(len(data))]

    for i in range(len(data)):
        for thing in data[i]:
            if type(thing) == datetime:
                new_data[i].append(thing.strftime("%Y-%m-%d"))
                continue
            new_data[i].append(thing)

    new_data.sort(key=lambda x: x[0])
    new_data.insert(0, col_names)
    return new_data
```

Rys 18. Implementacja w Pythoni-e

```
create function ClientRents(client_id int) return rent_table
as
    result rent_table;
begin
    select rent_type(id,OUTDATE,DUEDATE) bulk collect
    into result
    from RENT
    where CLIENTID = client_id;
    return result;
end;
```

Rys 19 Implementacja w bazie danych

8. Wyświetlanie naszych produktów w rezerwacjach

Window Title

Użytkownik: Jan, usr_id: 2

Wyloguj

YYYY-MM-DD

YYYY-MM-DD

Dodaj rezerwacje

Szukaj dostępnych narzędzi

Pokaz moje rezerwacje

Pokaz moje zarezerwowane narzędzia

Numer rezerwacji:

Anuluj rezerwacje

Numer rezerwacji:

Numer przedmiotu:

Ilość przedmiotu:

Dodaj do rezerwacji

Podaj tabele którą chcesz wyświetlić:

Ok

Rys 20. Widok w aplikacji

```
def show_my_reserved_items(self, client_id):
    result = self._cursor.execute(f"select * from myreservations('{client_id}')"
    data = result.fetchall()
    col_names = [row[0] for row in self._cursor.description]

    new_data = [[] for _ in range(len(data))]

    for i in range(len(data)):
        for thing in data[i]:
            if type(thing) == datetime:
                new_data[i].append(thing.strftime("%Y-%m-%d"))
                continue
            new_data[i].append(thing)

    new_data.sort(key=lambda x: x[0])
    new_data.insert(0, col_names)
    return new_data
```

Rys 21. Implementacja w Pythoni-e

```
create function MyReservations(client_id int) return rented_item_table
as
    result rented_item_table;
begin
    select rented_item_type(itemid,name,quantity,price,rentid,OUTDATE,DUEDATE) bulk collect
    into result
    from RENTEDITEMS
    where CLIENTID = client_id;
    return result;
end;
```

Rys 22. Implementacja w bazie danych

9. Przenoszenie do historii

Window Title

Użytkownik: Adam, usr_id: 8

Wyloguj

YYYY-MM-DD

YYYY-MM-DD

Dodaj rezerwacje

Szukaj dostępnych narzędzi

Pokaż moje rezerwacje

Pokaż moje zarezerwowane narzędzia

Numer rezerwacji:

Anuluj rezerwacje

Numer rezerwacji:

Numer przedmiotu:

Ilość przedmiotu:

Dodaj do rezerwacji

Podaj tabele którą chcesz wyświetlić:

Ok

Numer rezerwacji:

Data oddania:

Przenieś do historii

Rys 23. Widok w aplikacji

```
def move_to_history(self, rent_id, return_date):
    self._cursor.execute(f"begin\
                           movetohist('{rent_id}','{return_date}');\
                           end;")
```

Rys 24. Implementacja w Pythoni-e

```
create procedure MoveToHist(rent_id int, date_return date)
as
    client_id int;
    date_from date;
    date_to date;

begin
    select CLIENTID into client_id from RENT where rent_id = id;
    select OUTDATE into date_from from RENT where rent_id = id;
    select DUEDATE into date_to from RENT where rent_id = id;
    insert into RENTHIST(id, clientid, outdate, due date, returndate) VALUES (rent_id,client_id,date_from,date_to,date_return);
    MODIFYSTATUS( RENT_ID: rent_id, 'NEW_STATUS: 'D');
    commit;
end;
```

Rys 25. Implementacja w bazie danych