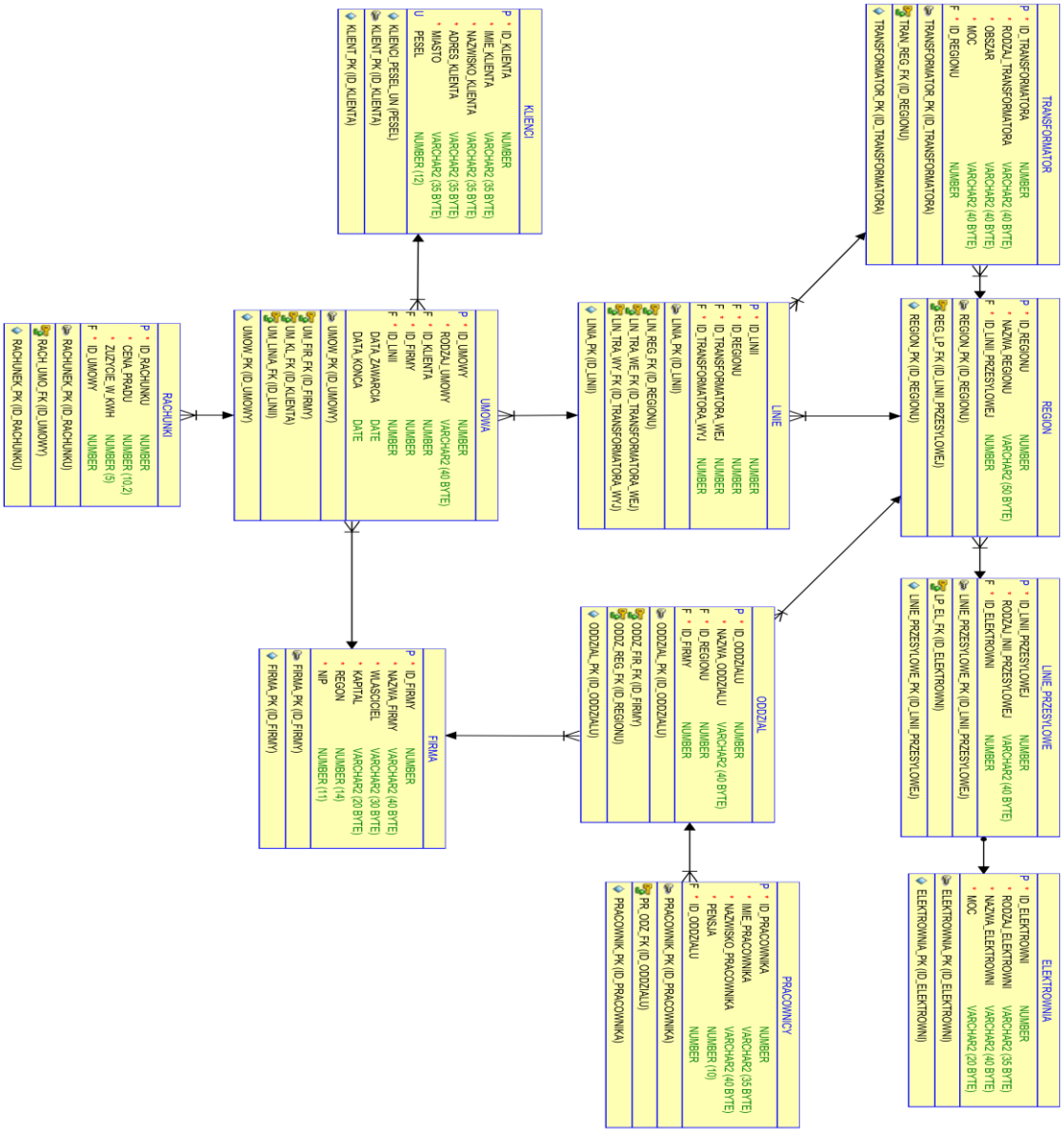
Bazy danych 1.

Firma energetyczna

Pieczaba Mateusz
Rusinek Michał
Sajdak Kamil
Rusiecki Bartosz
GR. 2.14B
2016/2017

Cel projektu:

W naszym projekcie stworzyliśmy bazę w Oracle 11g firm energetycznych a także klientów, oddziałów i pracowników im podlegającym. Wszystko obrazuje poniższy diagram. Klienta zrobiliśmy wykorzystując PHP/HTML/CSS. Pliki SQL jak i strony są spakowane w pliku bd_projekt214b.zip.



Funkcjonalność:

MENU klienta:

- HOME – Strona główna, wyświetla spis elektrowni oraz najnowszych klientów za pomocą podstawowych selectów i widoku.

```
SELECT * FROM elektrownia ORDER BY id_elektrowni
SELECT * FROM najdawniejsi_k
```

- SELECTY – Podstrona pozwalająca wyświetlić zawartość każdej z tabel za pomocą podstawowych selectów. Zmienna $Tabela zawiera po kolei nazwy tabli a $id_s po czym ma być segregowana zawartość.

  Elektrownie:

```
SELECT * FROM ".$Tabela." ORDER BY ".$id_s
```

- WIDOKI – Podstrona pozwalająca wywołać siedem widoków.

  Transformatory dla regionu – podporządkowywuje transformatory pod regiony.

```
create VIEW v_regiony
as
select  R.NAZWA_REGIONU,T.* from region R , TRANSFORMATOR T
where T.ID  REGIONU = R.ID  REGIONU;
```

  Umowa klienta – łączy klienta z wszystkimi jego umowami.

```
create view v_umowa
as
SELECT  K.*, U.DATA_ZAWARCIA,U.RODZAJ_UMOWY , F.REGON, F.NIP,R.ZUZYCIE_W_KWH,R.CENA_PRADU
from Umowa U, Klienci K, Firma F, RACHUNKI R
where U.ID_KLIENTA=K.ID_KLIENTA AND U.ID_FIRMY= F.ID_FIRMY and R.ID_UMOWY = U.ID_UMOWY
```

  Pracownicy z regionu– podporządkowywuje pracowników pod region.

```
create view v_prac_reg
as
select P.*, R.NAZWA_REGIONU from region R, oddzial O, PRACOWNICY P where R.ID_regionu =
O.ID_REGIONU and P.ID_ODDZIALU = O.ID_ODDZIALU;
```

  Linie przesyłowe elektrowni– wyświetla linie przesyłowe z których korzysta dana elektrownia.

```
create view v_ele_linie
as
select E.*, LL.RODZAJ_INII_PRZESYLOWEJ from ELEKTROWNIA E, LINIE_PRZESYLOWE LL
where LL.ID_ELEKTROWNI=E.ID_ELEKTROWNI;
```

Transformatory w danym regionie– podporządkowywuje transformatory pod region w którym się znajdują.

```
create view v_transf_regio
as
select R.NAZWA_REGIONU, T.OBSZAR,T.RODZAJ_TRANSFORMATORA from transformator T, REGION R
where T.Id_regionu = R.id_regionu;
```

Rachunki klienta– łączy klienta przez umowę z jego rachunkami, mnożąc przydzieloną mu cenę za prąd i zużycie wylicza miesięczny koszt.

```
create view v_rachunki
as
select K.imie_klienta, k.nazwisko_klienta ,CAST( Cena_pradu * zuzycie_w_kwh as varchar2(10)) || ' za
miesiac' Cena_za_prad
from rachunki R, umowa U, Klienci K
where U.id_umowy = R.id_umowy and U.id_klienta = K.id_klienta;
```

Najnowsi klienci – wylicza ile miesięcy minęło od podpisania umowy do dnia dzisiejszego i wyświetla ile upłynęło miesięcy.

```
reate view najdawniejsi_k
as
Select  UU.data,K.Imie_klienta,K.Nazwisko_klienta,K.Miasto from (select
ID_UMOWY,Round(MONTHS_BETWEEN(SYSDATE , DATA_ZAWARCIA)) data
 from umowa order by data) UU, umowa U, Klienci K where ROWNUM <= 10 and K.ID_KLIENTA =
U.ID_KLIENTA and  U.ID_UMOWY = UU.ID_UMOWY order by data;
```

- KURSORY – Podstrona pozwalająca wywoływać kursory, które wykonują opisane niżej zadania. Strona zawiera dwa pola do wprowadzania danych. Kursory są zawarte w procedurach , w sekcjach gdzie podaje się kwotę jest ona mnożona przez 100 a następnie w kodzie sql dzielona przez 100 aby móc przekazywać wartości z groszami.

Cena prądu dla klienta – kursor pobiera id i kwotę w złotówkach o którą ma być rachunek zwiększony/zmniejszony. Kursor na podstawie id znajduje id_umowy z klientem. W sekcji begin wykonywany jest update o podaną kwotę.

```
CREATE OR REPLACE procedure zniz(idd  number, hajs rachunki.cena_pradu%type )
IS zmienna number; i umowa%rowtype;
CURSOR ln1 IS
SELECT U.id_umowy FROM klienci K, umowa U, rachunki R  where K.ID_klienta = idd and U.ID_KLIENTA =
K.ID_KLIENTA and R.ID_UMOWY = U.ID_UMOWY ;
BEGIN
zmienna := hajs / 100;
FOR i IN ln1 LOOP
UPDATE rachunki set CENA_PRADU = CENA_PRADU + zmienna where ID_UMOWY = i.ID_UMOWY ;
END LOOP;
END;
```

Podwyżka pensji pracownikom z danego oddziału – kursor pobiera id i procentową wartość o którą ma być zwiększona pensja. Kursor na podstawie id znajduje id_oddziału z a potem podwyższa pensję wszystkim pracownikom z oddziału. W begin wykonywany się update, który zwiększa kwotę.

```
CREATE OR REPLACE procedure zniz(idd  number, hajs rachunki.cena_pradu%type )
IS zmienna number; i umowa%rowtype;
CURSOR ln1 IS
SELECT U.id_umowy FROM klienci K, umowa U, rachunki R  where K.ID_klienta = idd and U.ID_KLIENTA =
K.ID_KLIENTA and R.ID_UMOWY = U.ID_UMOWY ;
BEGIN
zmienna := hajs / 100;
FOR i IN ln1 LOOP
UPDATE rachunki set CENA_PRADU = CENA_PRADU + zmienna
where ID_UMOWY = i.ID_UMOWY ;
END LOOP;
END;
```

Zmiana ceny prądu dla miasta – kursor pobiera nazwę miasta oraz kwotę w złotówkach o jaką ma być zwiększona cena prądu dla wszystkich mieszkających w danym mieście. W begin wykonywany jest update zwiększający cenę.

```
CREATE OR REPLACE procedure miasto_transfF(miastoo VARCHAR2,hajs rachunki.cena_pradu%TYPE )
IS
i klienci%rowtype;
zmienna NUMBER;
CURSOR ln1 IS
SELECT U.id_umowy FROM  Klienci K, Umowa U where K.id_klienta = U.id_klienta and  K.miasto = miastoo;
BEGIN
zmienna := hajs / 100;
FOR i IN ln1 LOOP
UPDATE Rachunki set CENA_PRADU = CENA_PRADU + zmienna
 where id_umowy = i.id_umowy ;
END LOOP;
END;
```

Przedłużenie umowy – kursor pobiera wartość w latach o jaką ma być przedłużona umowa oraz id klienta. Zapytanie jest wywoływane w pętli w celu zwiększenia daty o daną wartość. W sekcji begin wywoływany jest kod przedłużający wartość daty w umowach dla danego klienta.

```
create or replace procedure umowa_rok_dluzej(idd  klienci.id_klienta%type)
is
i umowa%rowtype;
CURSOR dataa is
select U.id_umowy from klienci K, umowa U where K.id_klienta = U.id_klienta and K.id_klienta = idd ;
begin
for i in dataa LOOP
update Umowa set data_konca = (data_konca + INTERVAL '1' YEAR) where id_umowy = i.id_umowy;
end loop;
end;
```

- INSERTY – Podstrona umożliwia wywołanie procedur dodających do bazy określone dane. Posiada inputy do wprowadzania danych. ID pobierane są z sekwencji.

  Dodaj klienta – po wypełnieniu textboxów przekazywane są one do procedury, która dodaje je do bazy. Wstawiana data jest z dnia aktualnego a umowa ustawiana na 2 lata.

```
create or replace procedure dod_kl(
   imie klienci.imie_klienta%type, nazwisko klienci.nazwisko_klienta%type, adres klienci.adres_klienta%type,
   miastoo klienci.miasto%type, pes klienci.pesel%type, rodzaj_u umowa.rodzaj_umowy%type,
   id_f umowa.ID_FIRMY%type, id_lin UMOWA.ID_LINII%type, cena_p rachunki.cena_pradu%type,
   zuzycie rachunki.zuzycie_w_kwh%type)
is
begin
insert into klienci values(klient_seq.nextval, imie, nazwisko, adres, miastoo, pes);
insert into umowa values(umowa_seq.nextval,rodzaj_u, klient_seq.currval,id_f,id_lin,sysdate, sysdate +
INTERVAL '2' YEAR);
insert into rachunki values(rachunek_seq.nextval, cena_p, zuzycie,umowa_seq.currval);
end;
```

  Dodaj pracownika – po wypełnieniu textboxów przekazywane są one do procedury, która dodaje je do bazy.

```
create or replace procedure dod_pracow(
   imie PRACOWNICY.IMIE_PRACOWNIKA%type,
   nazwisko PRACOWNICY.NAZWISKO_PRACOWNIKA%type,
   pensja PRACOWNICY.pensja%type,
   id_o  PRACOWNICY.id_oddzialu%type )
is
begin
insert into pracownicy values(pracownik_seq.nextval, imie, nazwisko, pensja, id_o);
end;
```

- DELETE – Podstrona pozwala usunąć klienta, pracownika lub firmę. Procedury które są wywoływane są prawie identyczne, należy podać id elementu, który chcemy usunąć. W przypadku usuwania klienta lub firmy wywoływane są trigery, które kasują zależne od nich dane w bazie.

```
create or replace procedure us_kl(idd klienci.id_klienta%type)
is
begin
delete from klienci where id_klienta = idd;
end;
```

- TRIGERY – Uruchamiają się gdy wywoływana jest jakaś procedura usuwania.

  usun_kl_um – po usunięciu klienta o id usuwa umowy powiązane z klientem o tym id.

```
CREATE  or replace  TRIGGER usun_kl_um
AFTER DELETE ON klienci
FOR EACH ROW
BEGIN
DELETE FROM umowa WHERE id_klienta = :OLD.id_klienta;
END;
```

usun_kl_RACH – po usunięciu umowy o id usuwa rachunki powiązane z tym id.

```
CREATE  or replace  TRIGGER usun_kl_RACH
AFTER DELETE ON UMOWA
FOR EACH ROW
BEGIN
DELETE FROM rachunki WHERE id_umowy = :OLD.id_umowy;
END;
```

usun_pracownikow_od – po usunięciu firmy o danym id usuwa odziały powiązane z tym id.

```
CREATE  or replace  TRIGGER usun_pracownikow_od
AFTER DELETE ON firma
 for EACH ROW
BEGIN
DELETE FROM oddzial where id_firmy = :OLD.id_firmy;
END;
```

usun_pracownikow_umo – po usunięciu firmy o danym id usuwa umowy powiązane z tym id.

```
CREATE  or replace  TRIGGER usun_pracownikow_umo
AFTER DELETE ON firma FOR EACH ROW
BEGIN
DELETE FROM umowa where id_firmy = :OLD.id_firmy;
END;
```

usun_pracownikow – po usunięciu oddziału o danym id usuwa pracowników powiązane z tym id.

```
CREATE  or replace  TRIGGER usun_pracownikow
AFTER DELETE ON oddzial
FOR EACH ROW
BEGIN
DELETE FROM pracownicy where id_oddzialu = :OLD.id_oddzialu;
END;
```