**WOJSKOWA AKADEMIA TECHNICZNA**

**im. Jarosława Dąbrowskiego**

****

**WYDZIAŁ CYBERNETYKI**

****

Sprawozdanie

z zajęć laboratoryjnych z przedmiotu

Bazy Danych 2017

**Sprawozdanie wykonał:**  Piotr Jastrzębski I6X5S1

### Prowadzący zajęcia: mgr inż. Marcin Dąbkiewicz

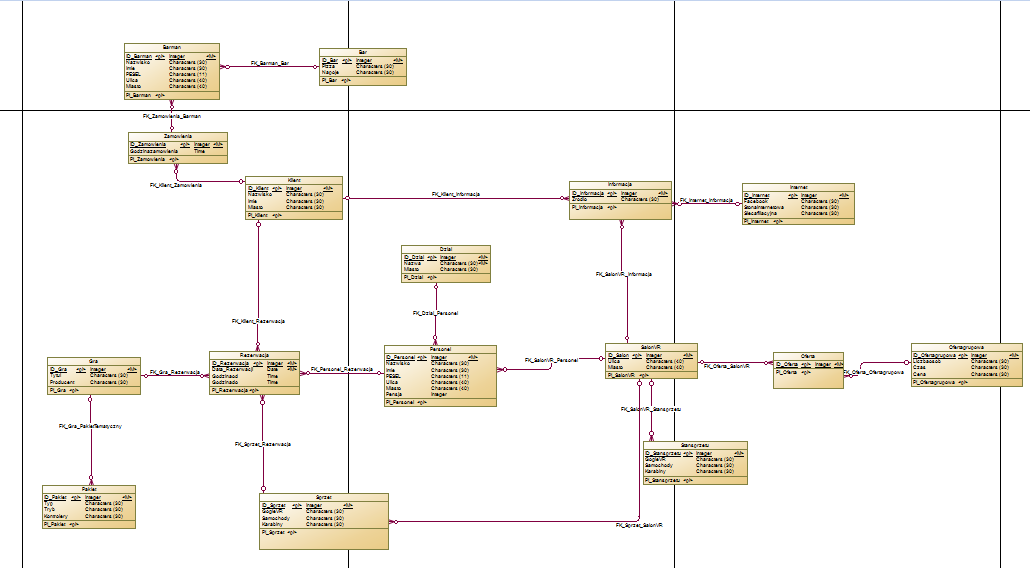
# Model biznesowy:

Klient znajduje w internecie informację na temat salonu VR. Wybiera wygodną dla siebie lokalizację, żeby odwiedzić salon jednego z franczyzobiorców. Klient dokonuje rezerwacji (telefonicznej bądź online) odnotowanej przez jednego z pracowników salonu. Każdy z pracowników przypisany jest do działu pracy. Klient po wejściu do lokalu dostaje od personelu odpowiednie gogle VR i sprzęt do symulacji doświadczeń. Każdy salon ma określoną liczbę dostępnego sprzętu.

Klient może samodzielnie wybrać w jaką grę ma ochotę zagrać. Wybór gry jest łatwo dostępny dzięki podziałowi na pakiety. Osoby odwiedzające salon mają możliwość skorzystanie z korzystnej oferty grupowej.

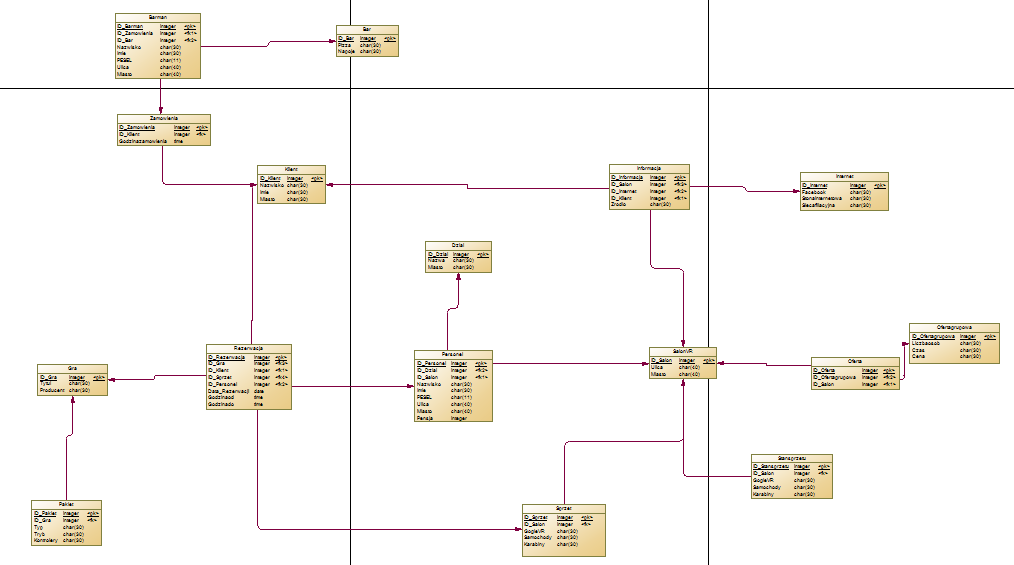
Klienci przebywający w salonie mogą również złożyć zamówienie u barmana na pizzę lub napoje.

# Screen modelu konceptualnego bazy danych:



Screen z modelu fizycznego bazy:

# Screen modelu fizycznego bazy:



# Zapytania:

--Przekrojowe zapytania(proste,średnie,trudne)- 24 zapytania lacznie-czyli po 8 każdego

1. SELECT \* FROM Personel LEFT OUTER JOIN SalonVR ON Personel.ID\_Salon=SalonVR.ID\_Salon --wszystkie kolumny z tabel personel i salonVR

2. SELECT Inf.ID\_Internet, Inf.ID\_Klient, I.ID\_Internet,I.Facebook

FROM Informacja as Inf RIGHT OUTER JOIN Internet as I ON Inf.ID\_Internet=I.ID\_Internet

3. SELECT Nazwisko, Imie, Zrodlo

FROM Klient INNER JOIN Informacja ON Klient.ID\_Klient=Informacja.ID\_Klient

4. SELECT Inf.Zrodlo, I.Siecafiliacyjna

FROM Informacja as Inf INNER JOIN Internet as I ON Inf.ID\_Internet=I.ID\_Internet

WHERE Siecafiliacyjna='tak' --pokazuje ilosc klientow skonwertowanych przez internet za pomoca sieci afiliacyjnej (kolumny- zrodlo+siecafiliacyjna)

5. SELECT TOP 3 Imie + ' ' +Nazwisko as Dane, Data\_Rezerwacji

FROM Klient INNER JOIN Rezerwacja ON Klient.ID\_Klient=Rezerwacja.ID\_Klient

ORDER BY Data\_Rezerwacji ASC --3 najblizsze datyrezerwacji z danymi osoby rezerwujacej

6. SELECT Tytul, Producent, Tryb

FROM Gra LEFT JOIN Pakiet P ON Gra.ID\_Gra=P.ID\_Gra

WHERE Tryb='Multiplayer' AND Producent like 'E%' AND Producent like '%s' --gry ktore mozna pograc multiplayer i zaczynaja sie na e i koncza na s

7. SELECT Imie + ' ' +Nazwisko as Dane, P.Miasto, S.Ulica

FROM Personel P

INNER JOIN SalonVR S ON P.ID\_Salon=S.ID\_Salon

WHERE P.Miasto='Wroclaw' --pokazuje ktory pracownik we wroclawiu pracuje w ktorym oddziale

8. SELECT S.ID\_Salon, Miasto, Liczbaosob, Cena

FROM SalonVR S

INNER JOIN Oferta ON S.ID\_Salon=Oferta.ID\_Salon

INNER JOIN Ofertagrupowa O ON O.ID\_Ofertagrupowa=Oferta.ID\_Ofertagrupowa

WHERE Miasto='Krakow' AND Czas='1,5h' --ceny w salonach w krakowie za 1,5h

1. SELECT P.Imie, P.Nazwisko, ID\_Rezerwacja, K.Nazwisko, K.Imie

FROM Personel P

INNER JOIN Rezerwacja R ON P.ID\_Personel=R.ID\_Personel

INNER JOIN Klient K ON R.ID\_Klient=K.ID\_Klient

WHERE SUBSTRING (P.Nazwisko,1,1)=SUBSTRING(K.Nazwisko,1,1)

AND SUBSTRING (P.Imie,1,1)=SUBSTRING(K.Imie,1,1) --wyswietla numer rezerwacji w ktorej klient i pracownik maja ta sama pierwsza litere w nazwisku i w imieniu

2. SELECT P.Nazwisko, P.Imie, K.Nazwisko, K.Imie, P.Miasto

FROM Personel P

INNER JOIN Rezerwacja R ON P.ID\_Personel=R.ID\_Personel

INNER JOIN Klient K ON R.ID\_Klient=K.ID\_Klient

WHERE P.Miasto='Wroclaw'

GROUP BY P.Nazwisko, P.Imie, K.Nazwisko, K.Imie, P.Miasto, Data\_Rezerwacji

HAVING Data\_Rezerwacji > 2018-04-05 -- imie i nazwisko personelu obsługującego, imie nazwisko klienta z Wroclawia obslugiwanych po 05.04.2018

3. SELECT MAX(Data\_Rezerwacji), Miasto

FROM Rezerwacja R

INNER JOIN Klient K ON R.ID\_Klient=K.ID\_Klient

GROUP BY Data\_Rezerwacji, Miasto

HAVING Data\_Rezerwacji > 2018-04-01 --??? znajduje kiedy i w jakim miescie jest ostatni termin ktory jest zarezerowowany przez klienta

4. SELECT Zrodlo, COUNT(Stonainternetowa) as Liczbaosobzestronyinternetowej

FROM Internet I

INNER JOIN Informacja Inf ON I.ID\_Internet=Inf.ID\_Internet

INNER JOIN Klient K ON K.ID\_Klient=Inf.ID\_Klient

WHERE Stonainternetowa='tak'

GROUP BY Zrodlo --liczy ilu klientow mialo informacje ze strony internetowej

5. SELECT MAX(SS.GogleVR) as GogleVRmaxliczba, SS.ID\_Salon, ID\_Rezerwacja

FROM Stansprzetu SS

INNER JOIN Sprzet S ON S.ID\_Salon=SS.ID\_Salon

INNER JOIN Rezerwacja R ON S.ID\_Sprzet=R.ID\_Sprzet

GROUP BY SS.ID\_Salon, ID\_Rezerwacja,SS.GogleVR

HAVING SS.GogleVR > 30 --numer rezerwacji, które zawiera numer salonnu w ktorym jest wiecej jak 30 googli

6. SELECT TOP 3 MAX(Gra.ID\_Gra),Tytul, Producent, Kontrolery

FROM Gra

INNER JOIN Pakiet P ON P.ID\_Gra=Gra.ID\_Gra

GROUP BY Kontrolery, Tytul, Producent, Gra.ID\_Gra

HAVING Gra.ID\_Gra > 5

ORDER BY Gra.ID\_Gra DESC --zwraca wlasciwosci gier 3 gier z największym indeksem z id gry większym od 5 (malejąco)

7. SELECT Nazwisko, Imie, COUNT(GogleVR)

FROM Klient K

INNER JOIN Rezerwacja R ON K.ID\_Klient=R.ID\_Klient

INNER JOIN Sprzet S ON R.ID\_Sprzet=S.ID\_Sprzet

WHERE GogleVR=2

GROUP BY Nazwisko, Imie --przyznaj po punkcie klientowi ktory zarezerwowal 2x okulary

8. SELECT Tytul, COUNT(Typ) as TypGier

FROM Pakiet P

INNER JOIN Gra ON P.ID\_Gra=Gra.ID\_Gra

WHERE Typ='Adrenalin'

GROUP BY Tytul, ID\_Pakiet

HAVING ID\_Pakiet>1 --przypisuje wartosc 1 do kazdej gry z trybem adrenalin, gdzie pakiet zaczynamy 'dolaczac' od pakiet>1

1. SELECT S.Samochody as SamochodyOperacja, SS.Samochody, S.ID\_Salon

FROM Sprzet S

INNER JOIN Stansprzetu SS ON S.ID\_Salon=SS.ID\_Salon

WHERE S.Samochody < 4

UNION ALL

SELECT S.Samochody, SS.Samochody, S.ID\_Salon

FROM Sprzet S

INNER JOIN Stansprzetu SS ON S.ID\_Salon=SS.ID\_Salon

WHERE S.Samochody > 0 --dane Samochody których ilosc jest mniejsza od 4 lub większa od 0 (tabela sprzet) + odpowiadajaca im ilos Samochody z tabeli stansprzetu

2. SELECT S.GogleVR, SS.GogleVR as GogleVRoperowane, S.ID\_Salon

FROM Sprzet S

INNER JOIN Stansprzetu SS ON S.ID\_Salon=SS.ID\_Salon

WHERE SS.GogleVR < 25

EXCEPT

SELECT S.GogleVR, SS.GogleVR, S.ID\_Salon

FROM Sprzet S

INNER JOIN Stansprzetu SS ON S.ID\_Salon=SS.ID\_Salon

WHERE SS.GogleVR > 15 --dane GogleVR których ilosc jest mniejsza od 25 (roznica zbiorow) większa od 15 (tabela stansprzet) + odpowiadajaca im ilos GogleVR z tabeli sprzet z danego salonu

--roznica zbiorow

3. SELECT S.Karabiny, SS.Karabiny as KarabinyOperowane, SS.ID\_Salon

FROM Sprzet S

INNER JOIN Stansprzetu SS ON S.ID\_Salon=SS.ID\_Salon

WHERE SS.Karabiny < 11

UNION ALL

SELECT S.Karabiny, SS.Karabiny, SS.ID\_Salon

FROM Sprzet S

INNER JOIN Stansprzetu SS ON S.ID\_Salon=SS.ID\_Salon

WHERE SS.Karabiny > 5 --dane Karabiny których ilosc jest mniejsza od 6 lub większa od 1(tabela stansprzetu) + odpowiadajaca im ilos Karabiny z tabeli sprzet z danego salonu

--suma zbiorow

4. SELECT S.GogleVR as GogleVRoperowane, SS.GogleVR, S.ID\_Salon

FROM Sprzet S

INNER JOIN Stansprzetu SS ON S.ID\_Salon=SS.ID\_Salon

WHERE S.GogleVR < 6

UNION ALL

SELECT S.GogleVR, SS.GogleVR, S.ID\_Salon

FROM Sprzet S

INNER JOIN Stansprzetu SS ON S.ID\_Salon=SS.ID\_Salon

WHERE S.GogleVR > 1 --dane googli których ilosc jest mniejsza od 6 lub większa od 1(tabela sprzet) + odpowiadajaca im ilos googli z tabeli stansprzetu

--suma zbiorow

5. SELECT ID\_Stansprzetu, ID\_Salon, GogleVR

FROM Stansprzetu

WHERE GogleVR > (SELECT AVG(GogleVR) FROM Stansprzetu) --salon w którym gogli jest więcej niż średnia liczba ze wszystkich gogli

6.SELECT ID\_Sprzet, Karabiny, ID\_Salon

FROM Sprzet

WHERE Karabiny = (SELECT MAX(Karabiny) FROM Sprzet) --wyswietla salon z rezerwacji w ktorej jest najwiecej karabinow sposrod wszystkich rezerwacji

7. SELECT S.Karabiny as Karabinyoperowane, SS.Karabiny

FROM Sprzet S

INNER JOIN Stansprzetu SS ON S.ID\_Salon=SS.ID\_Salon

WHERE S.Karabiny < 6

INTERSECT

SELECT S.Karabiny, SS.Karabiny

FROM Sprzet S

INNER JOIN Stansprzetu SS ON S.ID\_Salon=SS.ID\_Salon

WHERE S.Karabiny > 1 --dane Karabiny których ilosc jest mniejsza od 6 && większa od 1(tabela sprzet) + odpowiadajaca im ilos Karabiny z tabeli stansprzetu

--czesc wspolna zbiorow

8. SELECT ID\_Sprzet, Samochody, ID\_Salon

FROM Sprzet

WHERE Samochody = (SELECT MIN(Karabiny) FROM Sprzet) --wyswietla salon z rezerwacji w ktorej jest najmniej karabinow sposrod wszystkich rezerwacji

9. SELECT SS.GogleVR, SS.ID\_Salon, ID\_Rezerwacja

FROM Stansprzetu SS

INNER JOIN Sprzet S ON S.ID\_Salon=SS.ID\_Salon

INNER JOIN Rezerwacja R ON S.ID\_Sprzet=R.ID\_Sprzet

WHERE SS.GogleVR = 35 --wyswietla salony w których gogli jest 35

10. SELECT SS.GogleVR, SS.ID\_Salon, ID\_Rezerwacja

FROM Stansprzetu

WHERE SS.GogleVR=(

SELECT SS.GogleVR, SS.ID\_Salon, ID\_Rezerwacja

FROM Stansprzetu SS

INNER JOIN Sprzet S ON S.ID\_Salon=SS.ID\_Salon

INNER JOIN Rezerwacja R ON S.ID\_Sprzet=R.ID\_Sprzet

WHERE SS.GogleVR = 35 ) - 34

--wyswietla rezerwacje, w których gogli jest o 34 mniej niż w salonach w których gogli jest 35

# Procedury:

1. #AKTUALIZUJE CENE OFERTY GRUPOWEJ O PODANYM ID i czasie trwania

ALTER PROCEDURE "DBA"."UPDATE\_OFERTA\_GRUPOWA"(IN ID\_Ofertagrupowa INTEGER, IN czas VARCHAR(30), IN UCENA FLOAT) /\* [IN | OUT | INOUT] parameter\_nameparameter\_type [DEFAULT default\_value], ... \*/

/\* RESULT( column\_namecolumn\_type, ... ) \*/

BEGIN

UPDATE Ofertagrupowa

SET Ofertagrupowa.Cena = UCENA

WHERE Ofertagrupowa.ID\_Ofertagrupowa = ID\_Ofertagrupowa;

END

2.#AKTUALIZUJE DANE ADRESOWE Personelu

ALTER PROCEDURE "DBA"."UPDATE\_ADRES\_PERSONEL"(IN ID\_Per INTEGER, IN Ulica VARCHAR(40), IN Miasto VARCHAR(40) /\* [IN | OUT | INOUT] parameter\_nameparameter\_type [DEFAULT default\_value], ... \*/ )

/\* RESULT( column\_namecolumn\_type, ... ) \*/

BEGIN

UPDATE Personel

SET Personel.Ulica =Ulica, Personel.Miasto = Miasto

WHERE ID\_Personel = ID\_Per;

END

3.#ZWIEKSZENIE PENSJI PERSONELU O DANY PROCENT

ALTER PROCEDURE "DBA"."PODWYZKA\_PERSONEL"( IN N varchar(30),I varchar(30), procent int)

BEGIN

UPDATE Personel

SET PENSJA =((100+procent)\*PENSJA)/100

WHERE NAZWISKO = N AND IMIE = I

END

4.#PROCEDURA DODAJACA PRACOWNIKA

ALTER PROCEDURE "DBA"."DODANIE\_PRACOWNIKA"( IN Nazwisko VARCHAR(30), IN Imie VARCHAR(30), IN Pesel VARCHAR(11),

IN Ulica VARCHAR(30), IN MIASTO varchar(20), IN PENSJA NUMERIC)

/\* [IN | OUT | INOUT] parameter\_nameparameter\_type [DEFAULT default\_value], ... \*/

/\* RESULT( column\_namecolumn\_type, ... ) \*/

BEGIN

DECLARE @LICZNIK INTEGER;

DECLARE @ID\_P INTEGER;

DECLARE KURSOR SCROLL CURSOR

FOR SELECT ID\_Personel FROM Personel;

OPEN KURSOR;

SET @LICZNIK=0;

lp: LOOP

FETCH KURSOR INTO @ID\_P;

if sqlcode<> 0 then leave lp end if;

SET @LICZNIK = @LICZNIK + 1

END LOOP;

CLOSE KURSOR;

INSERT INTO "Personel" ("ID\_Personel", "Nazwisko", "Imie", "Pesel", "Ulica", "Miasto", "Pensja")

VALUES (@LICZNIK, Nazwisko, Imie, Pesel, Ulica, Miasto, Pensja);

END

5.#PROCEDURA DODAJACA KILIENTA

ALTER PROCEDURE "DBA"."DODANIE\_KLIENTA"( IN NAZWISKO VARCHAR(30), IN IMIE VARCHAR(20),IN Miasto varchar(20))

/\* [IN | OUT | INOUT] parameter\_nameparameter\_type [DEFAULT default\_value], ... \*/

/\* RESULT( column\_namecolumn\_type, ... ) \*/

BEGIN

DECLARE @LICZNIK INTEGER;

DECLARE @ID\_K INTEGER;

DECLARE KURSOR SCROLL CURSOR

FOR SELECT ID\_Klient FROM Klient ;

OPEN KURSOR;

SET @LICZNIK=0;

lp: LOOP

FETCH KURSOR INTO @ID\_K;

if sqlcode<> 0 then leave lp end if;

SET @LICZNIK = @LICZNIK + 1

END LOOP;

CLOSE KURSOR;

INSERT INTO "Klient" ( "ID\_Klient", "Nazwisko", "Imie", "Miasto" )

VALUES (@LICZNIK, Nazwisko, Imie, Miasto );

END

6.#PROCEDURA DODAJACA gre

ALTER PROCEDURE "DBA"."DODANIE\_GRY"( IN Tytul VARCHAR(30), IN Producent VARCHAR(30) )

/\* [IN | OUT | INOUT] parameter\_nameparameter\_type [DEFAULT default\_value], ... \*/

/\* RESULT( column\_namecolumn\_type, ... ) \*/

BEGIN

DECLARE @LICZNIK INTEGER;

DECLARE @ID\_G INTEGER;

DECLARE KURSOR SCROLL CURSOR

FOR SELECT ID\_Gra FROM Gra ;

OPEN KURSOR;

SET @LICZNIK=0;

lp: LOOP

FETCH KURSOR INTO @ID\_G;

if sqlcode<> 0 then leave lp end if;

SET @LICZNIK = @LICZNIK + 1

END LOOP;

CLOSE KURSOR;

INSERT INTO "Gra" ("ID\_Gra", "Tytul", "Producent")

VALUES (@LICZNIK, Tytul, Producent );

END

7.#PROCEDURA DODAJACA NAPOJE DO BARU

ALTER PROCEDURE "DBA"."DODANIE\_NAPOJE"( IN Pizza VARCHAR(30), IN Napoje VARCHAR(30) )

/\* [IN | OUT | INOUT] parameter\_nameparameter\_type [DEFAULT default\_value], ... \*/

/\* RESULT( column\_namecolumn\_type, ... ) \*/

BEGIN

DECLARE @LICZNIK INTEGER;

DECLARE @ID\_B INTEGER;

DECLARE KURSOR SCROLL CURSOR

FOR SELECT ID\_Bar FROM Bar ;

OPEN KURSOR;

SET @LICZNIK=0;

lp: LOOP

FETCH KURSOR INTO @ID\_B;

if sqlcode<> 0 then leave lp end if;

SET @LICZNIK = @LICZNIK + 1

END LOOP;

CLOSE KURSOR;

INSERT INTO "Bar" ( "ID\_Bar", "Pizza", "Napoje" )

VALUES (@LICZNIK, Pizza, Napoje );

END

8.#Aktualizowanie daty rezerwacji dokonanej przez klienta

ALTER PROCEDURE "DBA"."Data\_Rezerwacji"(IN ID\_Klient INTEGER, IN new\_date date)

BEGIN

UPDATE Rezerwacja

SET Data\_Rezerwacji = new\_date

WHERE Rezerwacja.ID\_ZAWODNIKA = ID\_Klient;

END

# Funkcje:

1.#FUNKCJA ZWRACAJACA LICZBE NAPOI BEDACYCH W AKTUALNEJ LISCIE ZAMOWIEN, na poszczególnym pojedynczym zamowieniu przyjętym w barze

ALTER FUNCTION "DBA"."NAPOJE"( IN NAZWA1 VARCHAR(30)/\* [IN] parameter\_name parameter\_type [DEFAULT default\_value], ... \*/ )

RETURNS INTEGER

DETERMINISTIC

BEGIN

DECLARE NAPOJE INTEGER;

SET NAPOJE=(SELECT COUNT(bar.Napoje)

FROM Bar AS bar

INNER JOIN Barman AS b ON bar.ID\_Bar= b.ID\_Bar

INNER JOIN Zamowienia AS z ON b.ID\_Zamowienia = z.ID\_Zamowienia

WHERE bar.ID\_Bar = NAZWA1);

RETURN NAPOJE;

END

2.#FUNKCJA ZWRACAJACA ILOSC REZERWACJI DOKONANYCH NA GRE O PODANEJ NAZWIE

ALTER FUNCTION "DBA"."ILOSC\_REZERWACJI\_NA\_GRE"( IN NAZWA VARCHAR(30)/\* [IN] parameter\_name parameter\_type [DEFAULT default\_value], ... \*/ )

RETURNS INTEGER

DETERMINISTIC

BEGIN

DECLARE LICZBA\_REZERWACJI INTEGER;

SET LICZBA\_REZERWACJI=(SELECT COUNT(\*)

FROM Rezerwacja AS R

INNER JOIN Gra AS G ON R.ID\_Gra = G.ID\_Gra

WHERE G.Tytul = NAZWA);

RETURN LICZBA\_REZERWACJI;

END

3.#ZWRACA OSTATNIA CYFRE NUMERU PESEL

ALTER FUNCTION "DBA"."PESEL\_OSTATNIA\_CYFRA" (IN PES VARCHAR(11))

RETURNS INTEGER

DETERMINISTIC

BEGIN

DECLARE @OST\_CYFRA INTEGER;

SET @OST\_CYFRA=CAST((SUBSTRING(PES,11,1)) AS INTEGER );

RETURN @OST\_CYFRA;

END

4.#ZWARACA SUME KONTROLNA NUMERU PESEL

ALTER FUNCTION "DBA"."PESEL"(IN PES VARCHAR(11))

RETURNS INTEGER

DETERMINISTIC

BEGIN

DECLARE @SUMA INTEGER;

DECLARE @OST\_CYFRA INTEGER;

DECLARE @MODULO INTEGER;

SET @SUMA=CAST((SUBSTRING(PES,1,1)) AS INTEGER ) + 3 \* CAST((SUBSTRING(PES,2,1)) AS INTEGER) + 7 \* CAST((SUBSTRING(PES,3,1)) AS INTEGER) + 9 \* CAST((SUBSTRING(PES,4,1)) AS INTEGER)

+ CAST((SUBSTRING(PES,5,1)) AS INTEGER) + 3 \* CAST((SUBSTRING(PES,6,1)) AS INTEGER) + 7 \* CAST((SUBSTRING(PES,7,1)) AS INTEGER) + 9 \* CAST((SUBSTRING(PES,8,1)) AS INTEGER)

+ CAST((SUBSTRING(PES,9,1)) AS INTEGER) + 3 \* CAST((SUBSTRING(PES,10,1)) AS INTEGER);

SET @MODULO = @SUMA % 10;

SET @OST\_CYFRA=10-@MODULO;

RETURN @OST\_CYFRA;

END

5.#Wypisuje ilu klientow dokonało rezerwacji

ALTER FUNCTION "DBA"."Klient\_Rezerwacja" (IN N INTEGER )

RETURNS INTEGER

BEGIN

DECLARE @SUMA INTEGER ;

DECLARE @REZERWACJE INTEGER;

DECLARE Rezerwacja\_kli SCROLL CURSOR

FOR (SELECT COUNT(ID\_Rezerwacja) FROM Rezerwacja);

OPEN Rezerwacja\_kli;

SET @SUMA=0;

SET @REZERWACJE=0;

lp: LOOP

FETCH Rezerwacja\_kli INTO @REZERWACJE;

if sqlcode<> 0 then leave lp end if;

SET @SUMA=@SUMA + @REZERWACJE;

END LOOP;

CLOSE Rezerwacja\_kli;

RETURN @SUMA;

END

6#Wypisuje ilu pracownikow przyjęło rezerwację

ALTER FUNCTION "DBA"."Personel\_Rezerwacja" (IN N INTEGER )

RETURNS INTEGER

BEGIN

DECLARE @SUMA INTEGER ;

DECLARE @REZERWACJE INTEGER;

DECLARE Rezerwacja\_per SCROLL CURSOR

FOR (SELECT COUNT(ID\_Rezerwacja) FROM Rezerwacja);

OPEN Rezerwacja\_per;

SET @SUMA=0;

SET @REZERWACJE=0;

lp: LOOP

FETCH Rezerwacja\_per INTO @REZERWACJE;

if sqlcode<> 0 then leave lp end if;

SET @SUMA=@SUMA + @REZERWACJE;

END LOOP;

CLOSE Rezerwacja\_per;

RETURN @SUMA;

END

7.#zwraca liczbe klientow o imieniu Wladyslaw

ALTER FUNCTION "DBA"."Klient\_Wladyslaw" (IN N INTEGER )

RETURNS INTEGER

BEGIN

DECLARE @SUMA INTEGER ;

DECLARE @KLIENTW INTEGER;

DECLARE Klient\_naz SCROLL CURSOR

FOR (SELECT COUNT(ID\_Nazwisko) FROM Klient WHERE Nazwisko=Wladyslaw);

OPEN Klient\_naz;

SET @SUMA=0;

SET @KLIENTW=0;

lp: LOOP

FETCH Klient\_naz INTO @KLIENTW;

if sqlcode<> 0 then leave lp end if;

SET @SUMA=@SUMA + @KLIENTW;

END LOOP;

CLOSE Klient\_naz;

RETURN @SUMA;

END

8.#zwraca liczbe salonów otwartych w krakowie

ALTER FUNCTION "DBA"."Salon\_Krakow" (IN N INTEGER )

RETURNS INTEGER

BEGIN

DECLARE @SUMA INTEGER ;

DECLARE @SALONKR INTEGER;

DECLARE Salon\_mia SCROLL CURSOR

FOR (SELECT COUNT(ID\_Salon) FROM SalonVR WHERE Miasto=Krakow);

OPEN Salon\_mia;

SET @SUMA=0;

SET @SALONKR=0;

lp: LOOP

FETCH Salon\_mia INTO @SALONKR;

if sqlcode<> 0 then leave lp end if;

SET @SUMA=@SUMA + @SALONKR;

END LOOP;

CLOSE Salon\_mia;

RETURN @SUMA;

END

# Triggery:

1.ALTER TRIGGER "Czas" BEFORE INSERT, UPDATE

ORDER 1 ON "DBA"."Ofertagrupowa"

/\* REFERENCING OLD AS old\_name NEW AS new\_name \*/

FOR EACH ROW /\* WHEN( search\_condition ) \*/

BEGIN

/\* Type the trigger statements here \*/

if(new\_name.Czas > 12)

then

Signal "osoby bedace w salonie jako grupa nie moga byc dluzej niz 12h"

endif

END

2.ALTER TRIGGER "gogle" BEFORE INSERT, UPDATE

ORDER 1 ON "DBA"."Stansprzetu"

/\* REFERENCING OLD AS old\_name NEW AS new\_name \*/

FOR EACH ROW /\* WHEN( search\_condition ) \*/

BEGIN

/\* Type the trigger statements here \*/

if(new\_name.GogleVR <=0)

THEN

SIGNAL "Gogli nie moze byc w salonie mniej niz 0"

ENDIF

END

3.ALTER TRIGGER "Karabiny" AFTER INSERT, UPDATE

ORDER 1 ON "DBA"."Sprzet"

/\* REFERENCING OLD AS old\_name NEW AS new\_name \*/

FOR EACH STATEMENT /\* WHEN( search\_condition ) \*/

BEGIN

/\* Type the trigger statements here \*/

if(new\_name.Karabiny <0)

THEN

SIGNAL "Nie mozna zarezerwowac mniej niz 0 karabinow"

ENDIF

END

4.ALTER TRIGGER "PersonelMiasto" BEFORE INSERT, UPDATE

ORDER 1 ON "DBA"."Personel"

/\* REFERENCING OLD AS old\_name NEW AS new\_name \*/

FOR EACH ROW /\* WHEN( search\_condition ) \*/

BEGIN

/\* Type the trigger statements here \*/

IF (new\_name.Miasto not like 'Warszawa' and new\_name.Miasto not like 'Krakow'

and new\_name.KONTYNENT not like 'Wroclaw')

THEN

SIGNAL "Wprowadz poprawne Miasto"

ENDIF

END

5.ALTER TRIGGER "SalonMiasto" BEFORE INSERT, UPDATE

ORDER 1 ON "DBA"."SalonVR"

/\* REFERENCING OLD AS old\_name NEW AS new\_name \*/

FOR EACH ROW /\* WHEN( search\_condition ) \*/

BEGIN

IF (new\_name.Miasto not like 'Warszawa' and new\_name.Miasto not like 'Krakow'

and new\_name.Miasto not like 'Wroclaw')

THEN

SIGNAL "Wprowadz poprawne Miasto"

ENDIF

/\* Type the trigger statements here \*/

END

6.ALTER TRIGGER "Samochody" BEFORE INSERT, UPDATE

ORDER 1 ON "DBA"."Sprzet"

/\* REFERENCING OLD AS old\_name NEW AS new\_name \*/

FOR EACH ROW /\* WHEN( search\_condition ) \*/

BEGIN

/\* Type the trigger statements here \*/

if(new\_name.Samochody > 3)

THEN

SIGNAL "Klient nie moze zamowic wiecej niz 3 samochody"

ENDIF

END

7.ALTER TRIGGER "Tryb" BEFORE INSERT, UPDATE

ORDER 1 ON "DBA"."Pakiet"

/\* REFERENCING OLD AS old\_name NEW AS new\_name \*/

FOR EACH ROW /\* WHEN( search\_condition ) \*/

BEGIN

/\* Type the trigger statements here \*/

IF (new\_name.Tryb not like 'Singleplayer' and new\_name.Tryb not like 'Multiplayer')

THEN

SIGNAL "Wprowadz poprawny tryb"

ENDIF

END

8.ALTER TRIGGER "Zrodlo" BEFORE INSERT, UPDATE

ORDER 1 ON "DBA"."Informacja"

/\* REFERENCING OLD AS old\_name NEW AS new\_name \*/

FOR EACH ROW /\* WHEN( search\_condition ) \*/

BEGIN

/\* Type the trigger statements here \*/

IF (new\_name.Zrodlo not like 'prasa' and new\_name.Zrodlo not like 'internet')

THEN

SIGNAL "Zakwalifikuj jako prasa lub internet"

ENDIF

END