

Projekt **Bazy Danych 2**

Wydział Elektrotechniki Automatyki i Informatyki

Politechnika Świętokrzyska

Studia: **Stacjonarne I stopnia**

Kierunek: **Informatyka**

Temat:

**Hurtownia danych dla sieci
saloników prasowych**

Grupa: **2ID13A**

Skład osobowy:

1. Klusek Wojciech
2. Kostecki Michał
3. Kowalczyk Tomasz

1. Wprowadzenie.

Tematem jaki otrzymaliśmy jest hurtownia danych dla sieci saloników prasowych.

Saloniki prasowe znajdują się w różnych miastach i sprzedają gazety.

Zadaniem hurtowni jest między innymi zinterpretowanie informacji o sprzedaży gazet w konkretnych salonikach przez konkretnych pracowników tak aby otrzymać informacje przydatne kierownictwu firmy. Dane służą zwiększeniu sprzedaży oraz optymalizacji zasobów. poprzez zamawianie większej ilości gazet, które sprzedają się lepiej od innych tak aby podaż był w stanie odpowiednio zaspokoić popyt.

Innym przykładem jest analiza jak pracownicy przyczyniają się do sukcesu firmy poprzez analizę przychodów generowanych przez każdego z nich. Łatwo można wtedy wyłować pracowników, którzy świetnie się sprawują na swoim stanowisku i wynagrodzić ich dając tym samym przykład innym co przekłada się na zwiększenie produktywności kadry.

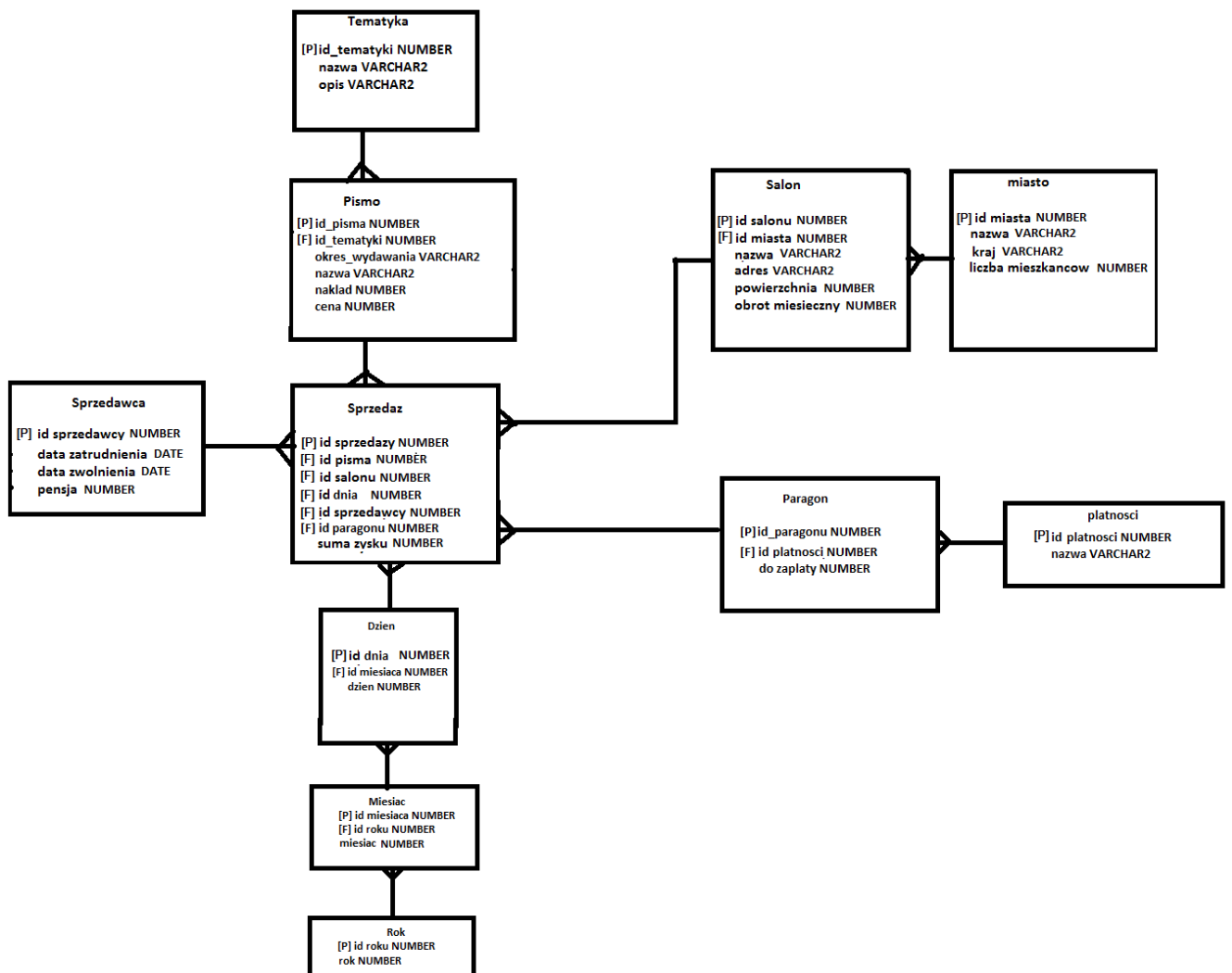
Śledzenie sprzedaży gazet w konkretnych salonikach pozwala firmie być na bieżąco z informacjami o ilości klientów co pozwala określić rentowność saloników gdyż można określić sensowność utrzymania saloniku lub inwestycji w jego rozwój.

Hurtownia pozwala także łatwo śledzić trendy dotyczące gazet w każdym saloniku. Dzięki analizie tych danych można poszerzać asortyment o kolejne pisma obracające się w gestii zainteresowań klientów przyczyniając się do zwiększenia sprzedaży.

Dzięki takim danym zarząd firmy może łatwiej podejmować trafne decyzje pozwalające na realne korzyści. Analiza jest prosta, łatwo zrozumieć wyniki i wprowadzić w życie odpowiednie reakcje na nadarzające się okazje lub zapobiec stratom wszystko na poziomie konkretnego saloniku. Obsługa także jest prosta ponieważ można śledzić wszystko z jednego centralnego ośrodka danych.

Podczas tworzenia projektu korzystaliśmy z narzędzia Git do synchronizacji danych. Repozytorium projektu znajduje się pod tym linkiem: [HIPERŁĄCZE](#)

2. Schemat ERD.

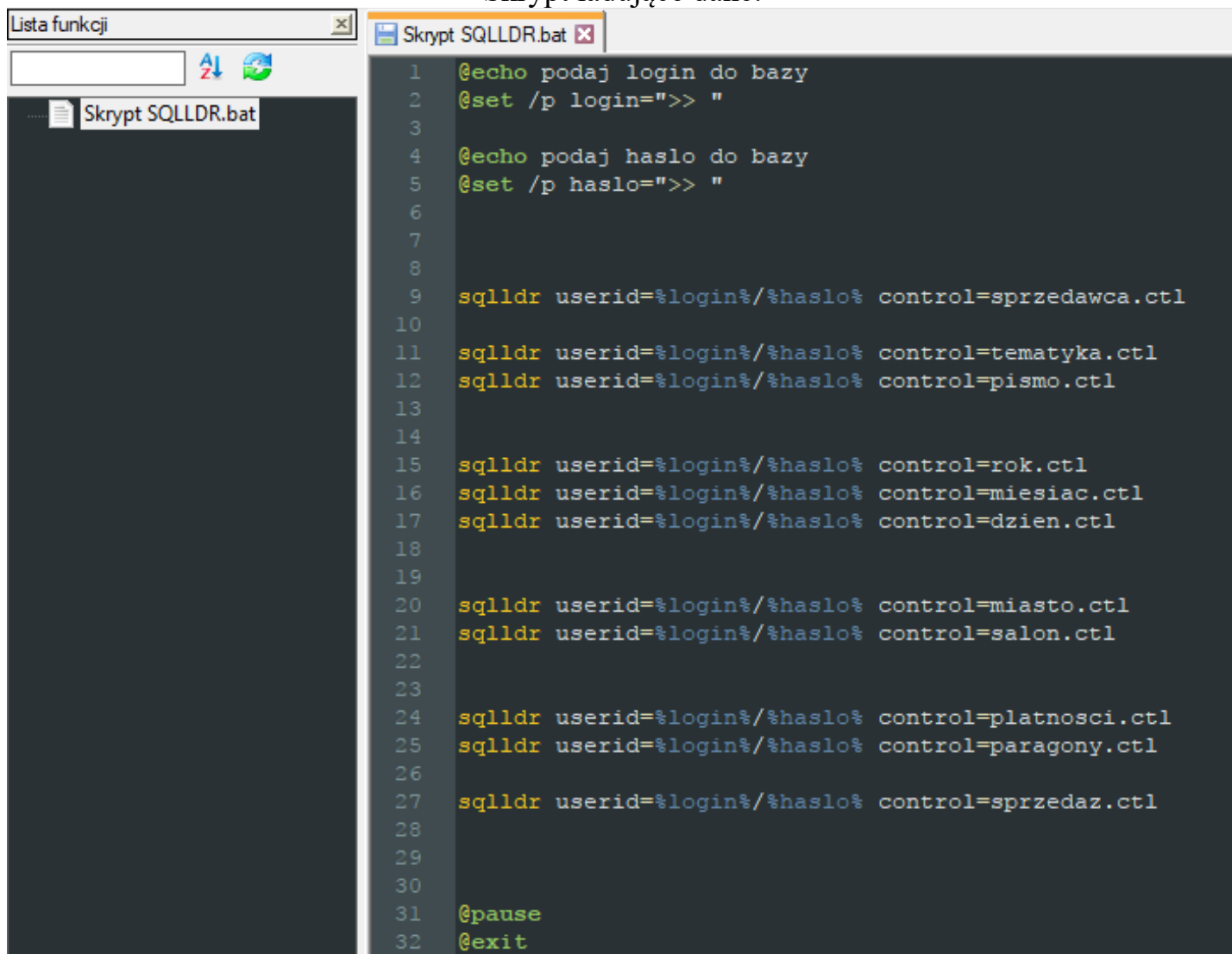


Schemat ERD bazy danych w większej rozdzielczości znajduje się w folderze z obecnie otworzonym plikiem sprawozdania pod nazwą: schemat bazy.png

3. Ładowanie danych

W folderze Pliki danych znajdującym się w głównym katalogu tej płyty znajduje się plik „Skrypt SQLLDR.bat” będący skrypcem powłoki systemu Windows. Pomaga on w szybkim ładowaniu danych do bazy. Jego działanie polega na tym, że pobiera od użytkownika najpierw login a potem hasło do bazy danych. Następnie za pomocą odpowiednich poleceń wywołuje plik sqlldr.exe ładując osobno dla każdej z tabel jej dane na podstawie jej pliku kontrolnego.

Skrypt ładujące dane:



```
Lista funkcji
Skrypt SQLLDR.bat

1 @echo podaj login do bazy
2 @set /p login=">> "
3
4 @echo podaj haslo do bazy
5 @set /p haslo=">> "
6
7
8
9 sqlldr userid=%login%/%haslo% control=sprzedawca.ctl
10
11 sqlldr userid=%login%/%haslo% control=tematyka.ctl
12 sqlldr userid=%login%/%haslo% control=pismo.ctl
13
14
15 sqlldr userid=%login%/%haslo% control=rok.ctl
16 sqlldr userid=%login%/%haslo% control=miesiac.ctl
17 sqlldr userid=%login%/%haslo% control=dzien.ctl
18
19
20 sqlldr userid=%login%/%haslo% control=miasto.ctl
21 sqlldr userid=%login%/%haslo% control=salon.ctl
22
23
24 sqlldr userid=%login%/%haslo% control=platnosci.ctl
25 sqlldr userid=%login%/%haslo% control=paragony.ctl
26
27 sqlldr userid=%login%/%haslo% control=sprzedaz.ctl
28
29
30
31 @pause
32 @exit
```

Pliki kontrolne znajdują się w tym samym folderze i posiadają nazwy *.ctl

Przykładowy plik o nazwie sprzedaz.ctl :

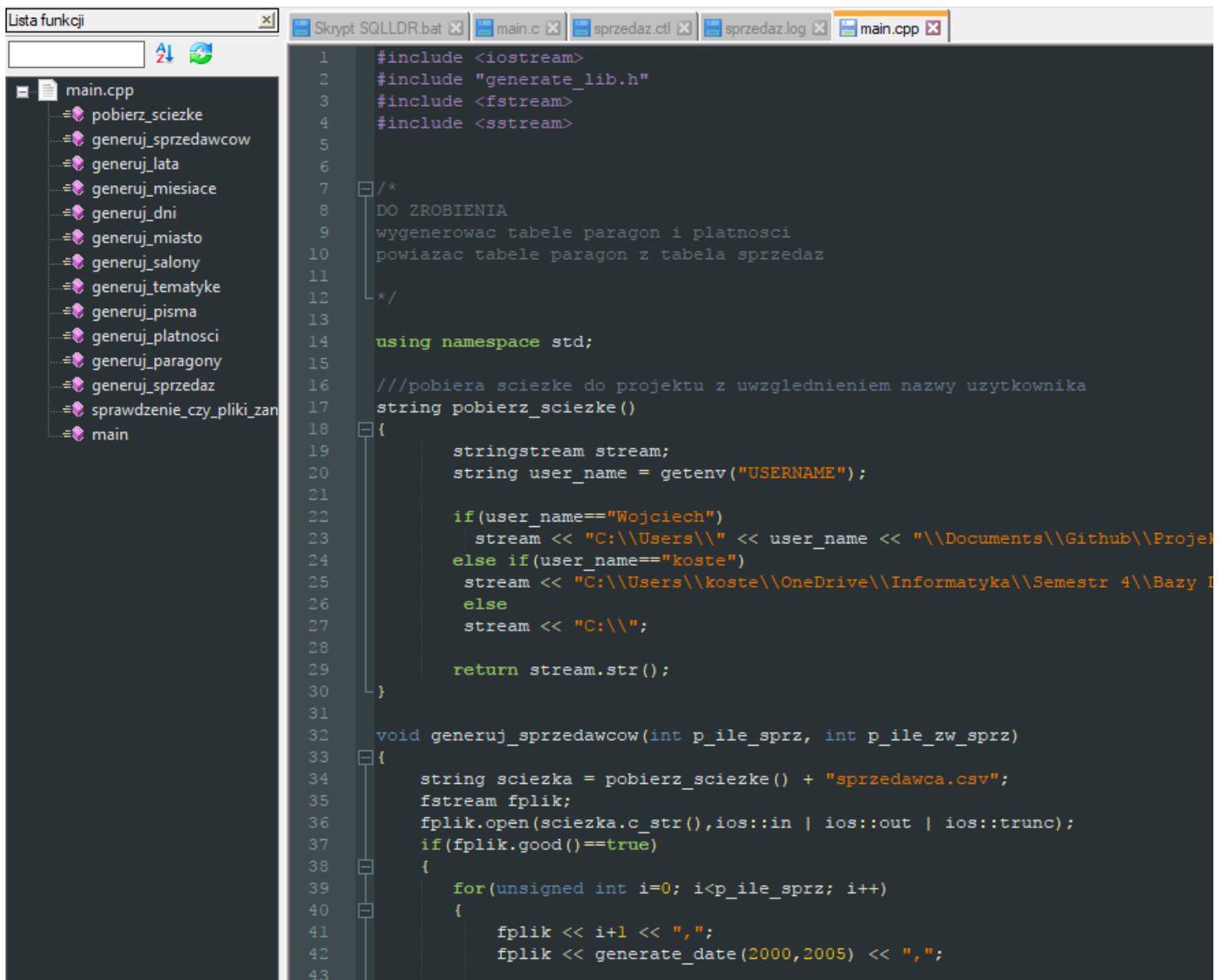
```
1 LOAD DATA
2 INFILE 'sprzedaz.csv'
3 BADFILE 'sprzedaz.bad'
4 DISCARDFILE 'sprzedaz.dsc'
5 REPLACE INTO TABLE Sprzedaz
6 FIELDS TERMINATED BY "," TRAILING NULLCOLS
7 ( id_sprzedazy, id_pisma, id_salonu, id_dnia, id_sprzedawcy, id_paragonu, suma_zysku )|
```

Pliki *.log opisują jak przebiegł proces ładowania danych do tabeli
Przykładowy plik sprzedaz.log :

```
1 SQL*Loader: Release 11.2.0.2.0 - Production on Pt Maj 5 20:11:46 2017
2
3 Copyright (c) 1982, 2009, Oracle and/or its affiliates. All rights reserved.
4
5 Control File:      sprzedaz.ctl
6 Data File:        sprzedaz.csv
7 Bad File:         sprzedaz.bad
8 Discard File:     sprzedaz.dsc
9 (Allow all discards)
10
11 Number to load: ALL
12 Number to skip: 0
13 Errors allowed: 50
14 Bind array:       64 rows, maximum of 256000 bytes
15 Continuation:     none specified
16 Path used:        Conventional
17
18 Table SPRZEDAZ, loaded from every logical record.
19 Insert option in effect for this table: REPLACE
20 TRAILING NULLCOLS option in effect
21
22
23      Column Name          Position  Len  Term Encl Datatype
24 -----
25 ID_SPRZEDAZY             FIRST    *    ,      CHARACTER
26 ID_PISMA                 NEXT     *    ,      CHARACTER
27 ID_SALONU                NEXT     *    ,      CHARACTER
28 ID_DNIA                  NEXT     *    ,      CHARACTER
29 ID_SPRZEDAWCY            NEXT     *    ,      CHARACTER
30 ID_PARAGONU              NEXT     *    ,      CHARACTER
31 SUMA_ZYSKU               NEXT     *    ,      CHARACTER
32
33 Record 12769: Rejected - Error on table SPRZEDAZ.
34 ORA-02291: integrity constraint (SYSTEM.SPRZEDAZ_DZIEN_FK) violated - parent key not found
35
36
37 Table SPRZEDAZ:
38 12769 Rows successfully loaded.
39 1 Row not loaded due to data errors.
40 0 Rows not loaded because all WHEN clauses were failed.
41 0 Rows not loaded because all fields were null.
42
43
44 Space allocated for bind array:          115584 bytes(64 rows)
45 Read buffer bytes: 1048576
46
47 Total logical records skipped:           0
48 Total logical records read:             12770
49 Total logical records rejected:          1
50 Total logical records discarded:         0
51
52 Run began on Pt Maj 05 20:11:46 2017
53 Run ended on Pt Maj 05 20:11:46 2017
54
55 Elapsed time was:      00:00:00.28
56 CPU time was:          00:00:00.03
```

Generator danych do tabel znajduje się w katalogu \Pliki danych\Generator danych
gdzie \ jest głównym katalogiem płyty.
Generator został napisany przy użyciu języka C++. Tworzy on pliki .csv zawierające
dane, które wprowadzamy do tabel.

Przykładowy zrzut ekranu z początku pliku źródłowego main.c :



```
1  #include <iostream>
2  #include "generate_lib.h"
3  #include <fstream>
4  #include <sstream>
5
6
7  /*
8   DO ZROBIENIA
9   wygenerowac tabele paragon i platnosci
10  powiazac tabele paragon z tabela sprzedaz
11
12  */
13
14  using namespace std;
15
16  ///pobiera sciezke do projektu z uwzglednieniem nazwy uzytkownika
17  string pobierz_sciezke()
18  {
19      stringstream stream;
20      string user_name = getenv("USERNAME");
21
22      if(user_name=="Wojciech")
23          stream << "C:\\Users\\" << user_name << "\\Documents\\Github\\Proje"
24      else if(user_name=="koste")
25          stream << "C:\\Users\\koste\\OneDrive\\Informatyka\\Semestr 4\\Bazy I
26      else
27          stream << "C:\\";
28
29      return stream.str();
30  }
31
32  void generuj_sprzedawcow(int p_ile_sprz, int p_ile_zw_sprz)
33  {
34      string sciezka = pobierz_sciezke() + "sprzedawca.csv";
35      fstream fplik;
36      fplik.open(sciezka.c_str(),ios::in | ios::out | ios::trunc);
37      if(fplik.good()==true)
38      {
39          for(unsigned int i=0; i<p_ile_sprz; i++)
40          {
41              fplik << i+1 << ",";
42              fplik << generate_date(2000,2005) << ",";
43          }
44      }
45  }
```

4. Zapytania.

- **ROLLUP**

```
SELECT Nvl(To_Char(id_salonu), 'Wszystkie salony') AS id_salonu,  
       Nvl(To_Char(id_pisma), 'Ogolem pism') AS id_pisma,  
       Count(id_pisma) AS liczba_sprzedanych  
FROM sprzedaz  
GROUP BY rollup (id_salonu, id_pisma);
```

Zapytanie do analizy sprzedaży poszczególnych gazet dla każdego z salonów osobno, Pozwala precyzyjnie określić sprzedaż każdej gazety (czasopisma itd.) w każdym salonie oraz ogólną liczbę sprzedanych gazet. Dzięki temu można dowiedzieć się jakie jest zapotrzebowanie oraz pośrednio jak wysoka jest sprzedaż dla każdego z saloników.

id_salonu	id_pisma	liczba_sprzedanych
1	1	14
1	2	22
1	3	10
1	4	14
1	5	14
1	6	22
1	7	17
1	8	19
1	9	14
1	10	13
1	11	25
1	12	18
1	13	18
1	14	16
1	15	13
1	16	10
1	17	15
1	18	19
1	19	14
1	20	30
1	Ogolem pism	337
2	1	13
2	2	20
2	3	13
2	4	13
2	5	18
2	6	18
2	7	15
2	8	16
2	9	15
2	10	26
2	11	20
2	12	19
2	13	14
2	14	13
2	15	19
2	16	15
2	17	25
2	18	13
2	19	16
2	20	15
2	Ogolem pism	336

```

SELECT Nvl(To_Char(id_salonu),'Wszystkie salony') AS id_salonu,
       Nvl(To_Char(id_sprzedawcy),'Sprzedawcy') AS id_sprzedawcy_w_salonie,
       count(id_paragonu) AS liczba_obsłużonych_klientow
FROM sprzedaz
GROUP BY rollup (id_salonu,id_sprzedawcy);

```

Zapytanie pozwala określić liczbę obsługiwanych przez każdego sprzedawcę klientów z podziałem na poszczególne salony oraz całkowitą liczbę obsługiwanych klientów. Pokazuje to na ocenę pracownika. W sytuacji gdy w danym sklepie byłoby obsługiwanych wielu klientów można by się zastanowić nad otwarciem następnego salonu w okolicy.

Dane zostały wygenerowane przez nasz własny generator stąd wartości które są jednakowe.

id_salonu	id_sprzedawcy_w_salonie	liczba_obsłużonych_klientow
22	34	336
22	Sprzedawcy	336
23	35	336
23	Sprzedawcy	336
24	36	336
24	Sprzedawcy	336
25	37	336
25	Sprzedawcy	336
26	38	336
26	Sprzedawcy	336
27	39	336
27	Sprzedawcy	336
28	40	336
28	Sprzedawcy	336
29	41	336
29	Sprzedawcy	336
30	42	336
30	Sprzedawcy	336
31	43	336
31	Sprzedawcy	336
32	44	336
32	Sprzedawcy	336
33	45	336
33	Sprzedawcy	336
34	46	336
34	Sprzedawcy	336
35	47	336
35	Sprzedawcy	336
36	48	336
36	Sprzedawcy	336
37	49	336
37	Sprzedawcy	336
38	50	336
38	Sprzedawcy	336
Wszystkie salony	Sprzedawcy	12769

- **CUBE**

```
SELECT Nvl(To_Char(id_salonu),'salon') AS id_salonu,
       Nvl(To_Char(id_sprzedawcy),'sprzedawcy') AS id_sprzedawcy,
       Nvl(To_Char(id_pisma),'pism') AS id_pisma,
       Count(id_pisma) AS ilosc_sprzedanych FROM sprzedaz
GROUP BY cube(id_salonu,id_sprzedawcy,id_pisma)
ORDER BY id_salonu,id_sprzedawcy,id_pisma asc;
```

Jest to bardziej szczegółowe zapytanie i stanowi połączenie dwóch poprzednich. Pozwala jednocześnie na określenie sprzedaży dla każdego salonu, pracownika i egzemplarza pisma. Dzięki temu można kreślić już szczegółowo jak wygląda sprzedaż, które pismo sprzedaje się lepiej, a które gorzej, jak wydajnie pracują pracownicy.

id_salonu	id_sprzedawcy	id_pisma	ilosc_sprzedanych
11	23	1	14
11	23	10	13
11	23	11	17
11	23	12	22
11	23	13	22
11	23	14	18
11	23	15	26
11	23	16	14
11	23	17	21
11	23	18	10
11	23	19	22
11	23	2	16
11	23	20	20
11	23	3	12
11	23	4	20
11	23	5	12
11	23	6	20
11	23	7	16
11	23	8	8
11	23	9	13
11	23	pism	336
11	sprzedawcy	1	14
11	sprzedawcy	10	13
11	sprzedawcy	11	17
11	sprzedawcy	12	22
11	sprzedawcy	13	22
11	sprzedawcy	14	18
11	sprzedawcy	15	26
11	sprzedawcy	16	14
11	sprzedawcy	17	21
11	sprzedawcy	18	10
11	sprzedawcy	19	22
11	sprzedawcy	2	16
11	sprzedawcy	20	20
11	sprzedawcy	3	12
11	sprzedawcy	4	20
11	sprzedawcy	5	12
11	sprzedawcy	6	20
11	sprzedawcy	7	16
11	sprzedawcy	8	8
11	sprzedawcy	9	13
11	sprzedawcy	pism	336

```

SELECT Nvl(To_Char(id_sprzedawcy),'Wszystkie sprzedawcy') AS id_sprzedawcy,
       Nvl(To_Char(id_dnia),'w miesiacu') AS id_dnia,
       Sum(suma_zysku) AS suma_zysku FROM sprzedaz
GROUP BY cube(id_sprzedawcy,id_dnia)
HAVING id_dnia<=28;
ORDER BY id_sprzedawcy, id_dnia asc;

```

Zapytanie określa kwotę jaką poszczególny pracownik zarobił dla firmy każdego dnia w ciągu każdego miesiąca, oraz sume zarobków każdego dnia. Dzięki temu sprzedawce można ocenić, oraz zorientować się jakie są przychody.

id_sprzedawcy	id_dnia	suma_zysku
13	1	36
14	1	12
15	1	17
16	1	38
17	1	25
18	1	42
19	1	11
20	1	17
21	1	25
22	1	10
23	1	9
24	1	23
25	1	36
26	1	38
27	1	10
28	1	9
29	1	30
30	1	42
31	1	18
32	1	25
33	1	32
34	1	32
35	1	13
36	1	17
37	1	17
38	1	34
39	1	35
40	1	29
41	1	17
42	1	14
43	1	13
44	1	19
45	1	24
46	1	35
47	1	38
48	1	25
49	1	19
50	1	26
Wszystkie sprzedawcy	1	912

- **GRUPPING**

```
SELECT Nvl(To_Char(sprzedaz.id_salonu),'Wszystkie salony') AS id_salonu,
       sprzedaz.id_dnia,
       Count(sprzedaz.id_sprzedazy) AS liczba_sprzedanych_gazet
FROM sprzedaz
GROUP BY Grouping sets ((id_salonu,id_dnia), id_dnia ,())
ORDER BY id_salonu desc, id_dnia;
```

Zapytanie określa liczbę pism sprzedanych każdego dnia w salonach, oraz bardziej szczegółowo w każdym z nich. Rozpatrując te dane tygodniowo lub w ciągu miesiąca możemy się dowiedzieć jak zmienia się sprzedaż w przedziale czasowym.

id_salonu	id_dnia	liczba_sprzedanych_gazet
Wszystkie salony	88	38
Wszystkie salony	89	38
Wszystkie salony	9	38
Wszystkie salony	90	38
Wszystkie salony	91	38
Wszystkie salony	92	38
Wszystkie salony	93	38
Wszystkie salony	94	38
Wszystkie salony	95	38
Wszystkie salony	96	38
Wszystkie salony	97	38
Wszystkie salony	98	38
Wszystkie salony	99	38
Wszystkie salony	Wszystkie salony	12769
9	1	1
9	10	1
9	100	1
9	101	1
9	102	1
9	103	1
9	104	1
9	105	1
9	106	1
9	107	1
9	108	1
9	109	1
9	11	1
9	110	1
9	111	1
9	112	1

```
SELECT Nvl(To_Char(sprzedaz.id_salonu),'Wszystkie salony') AS id_salonu,
       sprzedaz.id_dnia, Sum(sprzedaz.suma_zysku) AS zysk FROM sprzedaz
GROUP BY Grouping sets ((id_salonu,id_dnia), id_dnia ,())
ORDER BY id_salonu desc , id_dnia;
```

Zapytanie to pokazuje zysk ze sprzedaży w danym dniu, jednak tym razem w kontekście konkretnego salonu lub zbiorczo wszystkich salonów w danym dniu.

id_salonu	id_dnia	zysk
Wszystkie salony	85	1128
Wszystkie salony	86	940
Wszystkie salony	87	1016
Wszystkie salony	88	1023
Wszystkie salony	89	1048
Wszystkie salony	9	960
Wszystkie salony	90	1035
Wszystkie salony	91	1028
Wszystkie salony	92	918
Wszystkie salony	93	1008
Wszystkie salony	94	1052
Wszystkie salony	95	1010
Wszystkie salony	96	1082
Wszystkie salony	97	1021
Wszystkie salony	98	944
Wszystkie salony	99	977
Wszystkie salony	wszystkie dni	339736
9	1	25
9	10	37
9	100	11
9	101	9
9	102	15
9	103	15
9	104	31
9	105	34
9	106	18
9	107	11
9	108	17
9	109	17
9	11	20
9	110	17

- **Partition i okna przesuwne**

```
SELECT id_salonu, id_dnia, suma_zysku AS zysk_dzienny,
Sum(suma_zysku) OVER(
    PARTITION BY id_salonu
    ORDER BY id_dnia
    RANGE BETWEEN unbounded preceding AND CURRENT ROW
) AS zarobek
FROM sprzedaz
WHERE id_dnia < 29;
```

Zapytanie podobne do poprzedniego jednak z wykorzystaniem podziału na partycje oraz ruchomego okna obliczeniowego. Pokazuje zysk ze sprzedaży w danym dniu w ciągu miesiąca, a w ostatniej kolumnie uzyskany w ten sposób przychód z podziałem na poszczególne salony.

id_salonu	id_dnia	zysk_dzienny	zarobek
1	1	36	36
1	2	44	80
1	3	23	103
1	4	34	137
1	5	21	158
1	6	40	198
1	7	20	218
1	8	17	235
1	9	21	256
1	10	19	275
1	11	13	288
1	12	13	301
1	13	14	315
1	14	35	350
1	15	42	392
1	16	29	421
1	17	17	438
1	18	9	447
1	19	33	480
1	20	18	498
1	21	35	533
1	22	11	544
1	23	43	587
1	24	23	610
1	25	13	623
1	26	30	653
1	27	42	695
1	28	30	725
2	1	12	12
2	2	23	35
2	3	36	71

```

SELECT id_salonu, id_sprzedawcy, id_pisma AS id_pisma,
Count(id_pisma) OVER(
    PARTITION BY id_salonu
    ORDER BY id_dnia, id_pisma asc
    RANGE BETWEEN unbounded preceding AND CURRENT ROW
) AS liczba_sprzedanych_pism
FROM sprzedaz
WHERE id_dnia <29;

```

Zapytanie to wykorzystuje ruchome okno obliczeniowe do pokazania jak w danym dniu wyglądała liczba sprzedanych poszczególnych pism, dodając ją jednocześnie do już do tej pory sprzedanej liczby pism,

Dane zostały wygenerowane przez program przez co nie oddają rzeczywistej sytuacji

id_salonu	id_sprzedawcy	id_pisma	liczba_sprzedanych_pism
1	13	5	1
1	13	1	2
1	13	2	3
1	13	8	4
1	13	17	5
1	13	15	6
1	13	13	7
1	13	4	8
1	13	19	9
1	13	1	10
1	13	20	11
1	13	12	12
1	13	20	13
1	13	11	14
1	13	20	15
1	13	14	16
1	13	8	17
1	13	9	18
1	13	17	19
1	13	19	20
1	13	2	21
1	13	8	22
1	13	10	23
1	13	12	24
1	13	10	25
1	13	2	26
1	13	10	27
1	13	9	28
2	14	10	1
2	14	5	2
2	14	12	3

- **Funkcje rankingowe**

```
SELECT id_sprzedawcy,
       Sum(suma_zysku) AS zysk,
       Rank() OVER (PARTITION BY NULL ORDER BY Sum(suma_zysku) desc) AS ranking
FROM sprzedaz
GROUP BY id_sprzedawcy;
```

Zapytanie przedstawia ranking pracowników na podstawie zysku jaki wygenerowali dla firmy. Pozwala to na ich lepszą ocenę i ewentualne nagrodzenie.

id_sprzedawcy	zysk	ranking
30	9078	6
41	9073	7
17	9067	8
28	9063	9
31	9054	10
27	9009	11
46	8994	12
13	8981	13
25	8974	14
39	8967	15
48	8957	16
43	8952	17
44	8929	18
19	8916	19
50	8915	20
14	8912	21
32	8906	22
35	8900	23
24	8899	24
34	8896	25
42	8893	26
18	8890	27
36	8886	28
29	8843	29
16	8804	30
23	8798	31
40	8784	32
45	8781	33
33	8755	34
22	8714	35
21	8699	36
20	8689	37
47	8579	38
51	33	39

```

SELECT id_pisma,
       Count(id_pisma) AS liczba_sprzedanych,
       Rank() OVER (PARTITION BY NULL ORDER BY Count(id_pisma) desc) AS ranking
FROM sprzedaz
GROUP BY id_pisma;

```

Zapytanie przedstawia ranking liczby sprzedanych pism, pozwala to określić jakie pisma preferują klienci i reagowanie na to np. przez zwiększenie liczby egzemplarzy pisma dostępnych do kupienia w salonikach.

id_pisma	liczba_sprzedanych	ranking
2	665	1
18	665	1
3	662	3
5	658	4
19	657	5
12	654	6
1	652	7
8	648	8
20	646	9
11	643	10
14	640	11
6	632	12
17	624	13
4	624	13
10	624	13
7	620	16
15	617	17
13	615	18
16	612	19
9	611	20

5. Opis interfejsu.

Interfejs sieciowy hurtowni danych został napisany przy użyciu języka PHP, oraz modułu PDO.

Łączenie z bazą danych:

```
$dbh = null;

try {
    $server = "127.0.0.1";
    $db_username = "SYSTEM";
    $db_password = "qwerty";
    $service_name = "";
    $sid = "";
    $port = 1521;
    $dbtns = "(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP) (HOST = $server) (PORT = $port))
(CONNECT_DATA = (SERVICE_NAME = $service_name) (SID = $sid)))";

    $dbh = new PDO("oci:dbname=" . $dbtns . ";charset=utf8", $db_username, $db_password, array(
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_EMULATE_PREPARES => false,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC));
} catch (PDOException $e) {
    echo $e->getMessage();
}
```

Zmienna \$dbh przechowuje połączenie z bazą danych, i jest dostępna w większości plików interfejsu.

```
1  <?php include_once 'top.php'; ?>
2
3  <?php
4  $sql = "SELECT id_pisma,
5          Count(id_pisma) AS liczba_sprzedanych,
6          Rank() OVER (PARTITION BY NULL ORDER BY Count(id_pisma) desc) AS ranking
7  FROM sprzedaz
8  GROUP BY id_pisma";
9
10 $colNames = array(
11     'id_pisma',
12     'liczba_sprzedanych',
13     'ranking',
14 );
15
16 $opis = 'ranking liczby sprzedanych pism';
17
18 displaySql($sql, $opis);
19 echo maketable($dbh->query($sql), $colNames);
20 ?>
21
22 <?php include_once 'back.php'; ?>
23
```

Każdy plik wyświetlający dane z bazy danych oracle ma określoną strukturę. W pierwszej linii tego pliku zostaje dołączony plik "top.php" (wiersz nr 1). W ostatniej linii dołączamy plik "back.php" (wiersz nr 22). Pomiedzy tymi dwoma dołączeniami umieszczamy zapytanie sql które chcemy wykonać, nagłówki tabeli, oraz funkcje wykonujące zapytanie i "rysujące" tabele html (wiersze 4-19).

Plik top.php zawiera w sobie wyświetlanie górnej części kodu html(nagłówki html, dołączenia css/js, tytuł strony, menu), oraz połączenie z bazą danych.

Plik back.php zawiera w sobie jedynie tagi, zamykające kod html.

Do generowania tabeli html z danymi, użyliśmy biblioteki CodeDriller/maketable dostępnej pod adresem <https://github.com/CodeDriller/maketable>

Przykładowe strony interfejsu:

Tabela Sprzedaz	id_paragonu ↕	id_platnosci ↕	do_zaplaty ↕
	1	2	24
Tabela Paragony	2	2	24
	3	1	72
Tabela Platnosci	4	2	16
	5	2	40
Tabela Pismo	6	2	24
	7	2	56
Tabela Tematyka	8	1	80
	9	1	72
Tabela Salon	10	2	8
	11	1	64
Tabela Miasto	12	1	16
	13	2	16
Tabela Dzień	14	2	64
	15	1	40
Tabela Miesiąc	16	1	48
	17	1	24
Tabela Rok	18	2	16
	19	1	40
	20	2	8
		1	32

localhost/Projekt-BD2/Interfejs/selectTable.php?table=Paragony

Tabela Sprzedaz	ile klientow bylo obsluzonych w danym salonie przez jakiego sprzedawce, ogolem w salonie, ogolem obsluzonych klientow
Tabela Paragony	SQL: SELECT Nvl(To_Char(id_salonu),'Wszystkie salony') AS id_salonu, Nvl(To_Char(id_sprzedawcy),'Sprzedawcy') AS id_sprzedawcy, Count(id_paragonu) AS liczba_obsluzonych_klientow FROM sprzedaz GROUP BY rollup (id_salonu,id_sprzedawcy)
Tabela Platnosci	
Tabela Pismo	id_salonu ↕ id_sprzedawcy ↕ liczba_obsluzonych_klientów ↕
Tabela Tematyka	Wszystkie salony Sprzedawcy 12769
Tabela Salon	1 Sprzedawcy 337
	1 13 336
Tabela Miasto	2 14 336
	2 Sprzedawcy 336
Tabela Dzień	3 15 336
	3 Sprzedawcy 336
Tabela Miesiąc	4 16 336
	4 Sprzedawcy 336
Tabela Rok	5 17 336
	5 Sprzedawcy 336
Tabela Sprzedawca	6 18 336
	6 Sprzedawcy 336
rollup1	7 19 336
	7 Sprzedawcy 336
rollup2	8 20 336
	8 Sprzedawcy 336
cube1	9 21 336
	9 Sprzedawcy 336
cube2	10 22 336
	10 Sprzedawcy 336
	11 23 336

6. Podsumowanie.

Podczas tworzenia tego projektu udało nam się zrealizować wszystkie postawione przed nami cele. Nauczyliśmy się po co istnieją hurtownie danych, jak różnią się od relacyjnych baz danych oraz w jaki sposób należy je projektować i implementować.

Zapytania kierowane do hurtowni danych mają dużą wartość podczas kierowania przedsiębiorstwem, pozwalają w łatwy sposób zrozumieć sytuację w jakiej obecnie znajduje się firma i pomagają podjąć prawidłowe decyzje jak dalej nią zarządzać.