Akademia Górniczo-Hutnicza w Krakowie
Wydział Informatyki, Elektroniki i Telekomunikacji

Podstawy baz danych - projekt i implementacja systemu bazodanowego

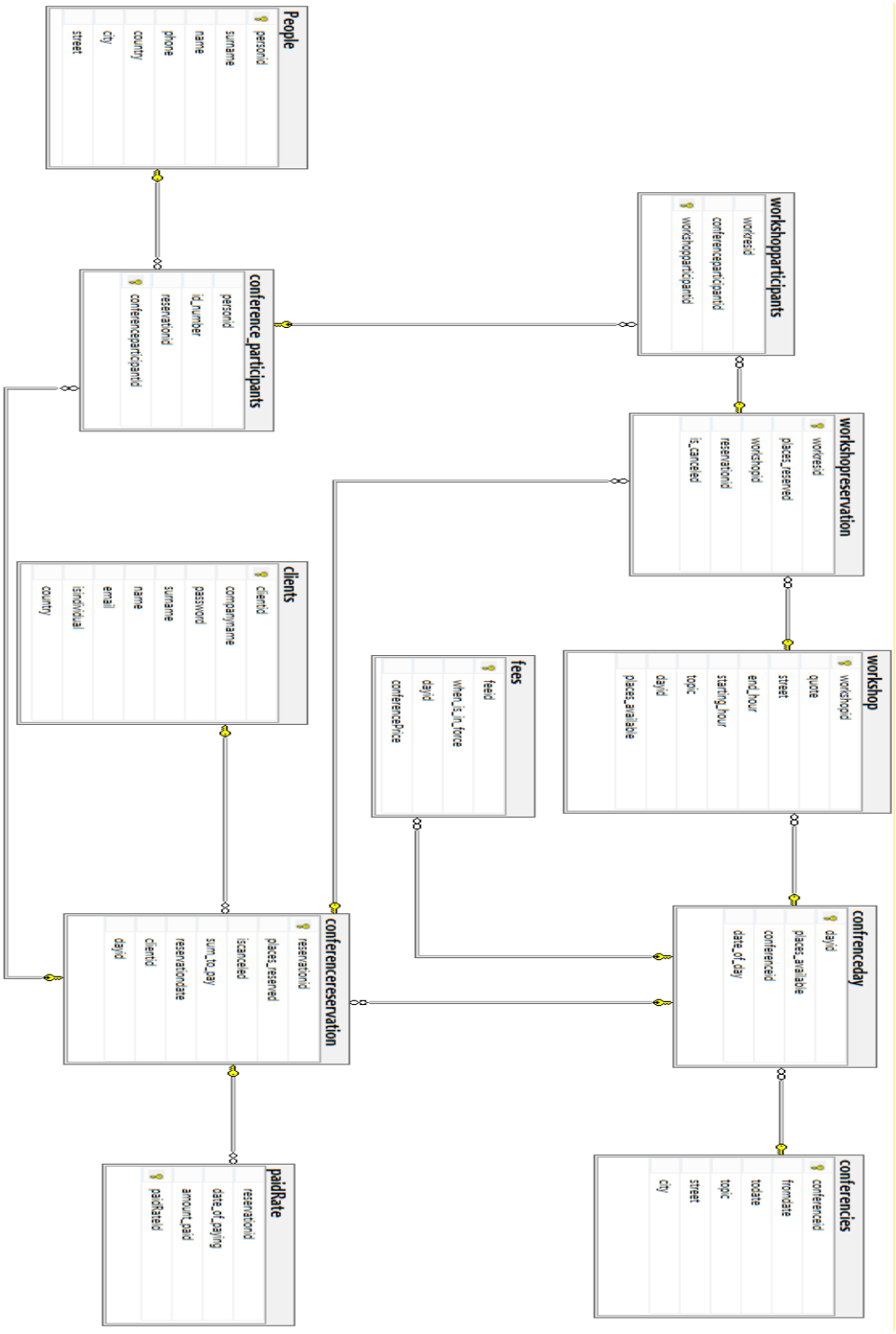
Katedra Informatyki
Studia inżynierskie
Drugi rok

Robert Bielas
Dominik Hendzel

1. Opis systemu bazodanowego

Firma organizuje konferencje, które mogą być jedno lub kilkudniowe. Klientami mogą być zarówno indywidualne osoby jak i firmy, natomiast uczestnikami konferencji są osoby (firma nie musi podawać od razu przy rejestracji listy uczestników może zarezerwować odpowiednią ilość miejsc na określone dni oraz na warsztaty, natomiast na 2 tygodnie przed rozpoczęciem musi te dane uzupełnić). Dla konferencji kilkudniowych, uczestnicy mogą rejestrować na dowolne z tych dni, dowolną liczbę osób. Klient może zmienić liczbę osób na rezerwacji, lub całkiem ją anulować (do 2 tygodni przed konferencją). System obsługiwany jest przez serwis www, który powinien korzystać z procedur i funkcji udostępnianych przez bazę. Baza dostarcza danych do wyświetlenia na stronie, natomiast poprzez serwis użytkownik może obsługiwać bazę bez bezpośredniego dostępu do niej.

2. Schemat bazy



3. Plik szkielet_bazy.sql, służący do stworzenia tabel

```
IF OBJECT_ID('dbo.paidRate', 'U') IS NOT NULL
      drop table paidRate
IF OBJECT_ID('dbo.workshopparticipants', 'U') IS NOT NULL
      drop table workshopparticipants
IF OBJECT_ID('dbo.conference_participants', 'U') IS NOT NULL
      drop table conference_participants
IF OBJECT_ID('dbo.workshopreservation', 'U') IS NOT NULL
      drop table workshopreservation
IF OBJECT_ID('dbo.workshop', 'U') IS NOT NULL
      drop table workshop
IF OBJECT_ID('dbo.fees', 'U') IS NOT NULL
      drop table fees
IF OBJECT_ID('dbo.conferencereservation', 'U') IS NOT NULL
      drop table conferencereservation
IF OBJECT_ID('dbo.confrenceday', 'U') IS NOT NULL
      drop table confrenceday
IF OBJECT_ID('dbo.conferencies', 'U') IS NOT NULL
      drop table conferencies
IF OBJECT_ID('dbo.People', 'U') IS NOT NULL
      drop table People
IF OBJECT_ID('dbo.clients', 'U') IS NOT NULL
      drop table clients

create table fees(
      feeid int primary key identity(0,1),
      when_is_in_force date,
      dayid int,
      conferencePrice real,
      INDEX fees_FKIndex1(dayid)
)

create table conferencies(
      conferenceid int primary key identity(0,1),
      fromdate date,
      todate date,
      topic varchar(50),
      street varchar(50),
      city varchar(50)
)

create table workshop(
      workshopid int primary key identity(0,1),
      quote real check(quote >=0),
      street varchar(50),
      end_hour time,
      starting_hour time,
      topic varchar(50),
```

```
	dayid int,
	places_available int check(places_available >= 0),
	INDEX workshop_FKIndex1(dayid)
)

create table conferencereservation(
	reservationid int primary key identity(0,1),
	places_reserved int ,
	iscanceled bit,
	sum_to_pay real check(sum_to_pay >= 0),
	reservationdate date,
	clientid int,
	dayid int,
	INDEX conferencereservation_FKIndex1(clientid),
	INDEX conferencereservation_FKIndex2(dayid)
)

create table confrenceday(
	dayid int primary key identity(0,1),
	places_available int check(places_available >= 0),
	conferenceid int,
	date_of_day date
	INDEX confrenceday_FKIndex1(conferenceid)
)


create table workshopreservation(
	workresid int primary key identity(0,1),
	places_reserved int check (places_reserved >= 0),
	workshopid int,
	reservationid int,
	is_canceled bit,
	INDEX workshopreservation_FKIndex1(workshopid),
	INDEX workshopreservation_FKIndex2(reservationid)
)


create table People(
	personid int primary key identity(0,1),
	surname varchar(15),
	name varchar(15),
	phone char(14) check(phone like '[0-9][0-9][0-9][0-9] [0-9][0-9][0-
9] [0-9][0-9] [0-9][0-9]'),
	country varchar(50),
	city varchar(50),
	street varchar(50)
)

create table paidRate(
		reservationid int,
		date_of_paying date,
```

```
            amount_paid real,
            paidRateId int primary key identity(0,1),
            INDEX conferencereservation_FKIndex1(reservationid)
)

create table conference_participants(
      personid int,
      id_number char(6) check(id_number like '[0-9][0-9][0-9][0-9][0-9]
[0-9]'),
      reservationid int,
      conferenceparticipantId int primary key identity(0,1),
      INDEX conference_participants_FKIndex1(reservationid),
      INDEX conference_participants_FKIndex2(personid)
)

create table workshopparticipants(
      workresid int,
      conferenceparticipantid int,
      workshopparticipantid int primary key identity(0,1),
      INDEX workshopparticipants_FKIndex1(workresid),
      INDEX workshopparticipants_FKIndex2(conferenceparticipantid)
)

create table clients(
      clientid int primary key identity(0,1),
      companyname varchar(80),
      password varchar(50) not null,
      surname varchar(50),
      name varchar(50),
      email varchar(50) not null,
      isindividual bit not null,
      country varchar (50) not null
)

alter table conferencereservation
add foreign key (clientid)
references clients(clientid)

alter table conferencereservation
add foreign key (dayid)
references confrenceday(dayid)

alter table fees
add foreign key (dayid)
references confrenceday(dayid)

alter table workshop
add foreign key (dayid)
references confrenceday(dayid)

alter table workshopreservation
```

```
add foreign key (workshopid)
references workshop(workshopid)

alter table workshopreservation
add foreign key (reservationid)
references conferencereservation(reservationid)

alter table workshopparticipants
add foreign key (workresid)
references workshopreservation(workresid)

alter table workshopparticipants
add foreign key (conferenceparticipantid)
references conference_participants(conferenceparticipantid)

alter table conference_participants
add foreign key (personid)
references People(personid)

alter table conference_participants
add foreign key (reservationid)
references conferencereservation(reservationid)

alter table paidRate
add foreign key (reservationid)
references conferencereservation(reservationid)

alter table confrenceday
add foreign key (conferenceid)
references conferencies(conferenceid)
```

4. Opis tabel:
Clients – tabelka opisująca klientów składających rezerwacje na konferencje i warsztaty. Kluczem głowym jest clientId. Ponieważ rejestracja rezerwacji odbywa się za pomocą serwisu internetowego, w bazie przechowujemy email klienta i hasło. Klient, gdy jest osobą prywatną, pole isindividual ma ustawione na 1 oraz podaje imię i nazwisko, companyname zostaje wypełnione wartością null. W przeciwnym wypadku pola name i surname nie zostają wypełnione, natomiast uzupełniana jest komórka companyname zaś isindividual przechowuje wartość 0.

ConferenceReservation – tabelka przechowująca informacje o rezerwacjach konferencji klientów.
Kluczem głównym jest reservationId. Należy podać ilość miejsc, którą pragnie się zarezerwować (places_reserved). Tabela posiada klucze obce clientid oraz dayid wiążące każdy jej wiersz z odpowiednim wierszem odpowiednio tabel clients oraz confrenceday. Sum_to_pay jest kwotą do zapłacenia za rezerwację warsztatu i za zarezerwowany dzień konferencji. IsCanceled domyślnie ustawiany jest na wartość 0.

PaidRate – tabela informująca o płatnościach klientów. Każdą płatność identyfikuje klucz główny paidRateId. Amount_paid jest kwotą lub częścią należnej kwoty sum_to_pay w odpowiednim wierszu conferenceReservation, z którym krotka w paidRate jest związana kluczem obcym reservationId. Tabela przechowuje informacje o dacie wpłaty(date_of_paying), która determinuje, czy klient ma zaległości.

ConfrenceDay – tabela, która identyfikuje każdy z dni danej konferencji. Posiada atrybuty: dayid (klucz główny), ilość miejsc dostępnych (places_available), datę w której ten dzień się rozgrywa (dare_of_day) oraz klucz obcy conferenceId wiążący ją z tabelą conferencies.

Fees – tabela zestawiająca progi cenowe dla danego dnia konferencji. Atrybuty: feeid (klucz główny), datę, od której obowiązuje (when_is_in_force), klucz obcy dayid wiążący ją z tabelą confrenceday oraz cenę dnia konferencji obowiązującą od daty when_is_in_force.

Conferencies – tabela identyfikująca konferencje. Kluczem głównym jest conferenceId. Tabela posiada atrybuty fromdate oraz todate, informujące jak długo odbywa się dana konferencja (od dnia do dnia). Każda konferencja posiada myśl przewodnią (temat/nazwę) topic, oraz ulicę i miasto (adres, który pokazuje, gdzie ma miejsce dana konferencja).

Workshop – tabela opisująca warsztaty. Posiada klucz główny workshopId. Każdy wiersz zawiera informację o cenie warsztatu(quote), przy jakiej ulicy ma miejsce warsztat (street), godzinę od której się zaczyna oraz godzinę o której się kończy (starting_hour, end_hour), temat/nazwę topic, ilość dostępnych miejsc places_available oraz klucz obcy dayId będący referencją na odpowiednią krotkę w tabeli confrenceday.

WorkshopReservation – tabela, która zawiera informacje o rezerwacjach warsztatów. Kluczem głównym jest workresid. Tabela posiada atrybuty places_reserved (ilość miejsc zarezerwowanych w ramach danej rezerwacji warsztatu), klucze obce workshopId oraz reservationId wiążące ją odpowiednio z tabelami Workshop i ConferenceReservation, a także informację is_canceled (czy rezerwacja jest anulowana).

Conference_participants - tabelka przechowująca spis osobowy zarejestrowanych uczestników danego dnia konferencji w ramach rezerwacji, na którą istnieje referencja poprzez klucz obcy reservationId. Kluczem głównym jest conferenceParticipantId. Jeżeli uczestnik jest studentem, to przy rejestracji okazuje legitymację, której numer jest przechowywany w atrybucie id_number.
Klucz obcy personId wiąże każdego uczestnika z jego danymi osobowymi w tabeli People.

People – tabela przechowująca dane osobowe uczestników rezerwacji.

WorkshopParticipants – tabelka łącznikowa, która przechowuje uczestników w ramach danej rezerwacji warsztatu. Identyfikowani są przez id uczestnika danego dnia konferencji. Z tabelką workshopReservation łączy ją klucz obcy workresid. Posiada klucz główny workshopParticipantId.

5. Implementacja systemu:
   a) polecenia, widoki

```
--zestawia dayid, ilosc dostepnych miejsc i ilosc zapelnionych
miejsc
select cd.dayid,places_available,sum(places_reserved)
from conferencereservation cr join confrenceday cd
on cr.dayid = cd.dayid
group by cd.dayid,places_available

--wyswietla klientow, rezerwacje konferencji i odpowiadajace im
rezerwacje warsztatow
select c.clientid,cr.reservationid conf_res,wr.workresid work_res
from clients as c join conferencereservation as cr
on cr.clientid = c.clientid
join conference_participants as cp
on cp.reservationid = cr.reservationid
join workshopparticipants as wp
on wp.conferenceparticipantid = cp.conferenceparticipantid
join workshopreservation as wr
on wr.workresid = wp.workresid
group by c.clientid,cr.reservationid,wr.workresid
order by 1,2

--jw, wariant II
select c.clientid,cr.reservationid conf_res,wr.workresid work_res
from clients as c join conferencereservation as cr
on cr.clientid = c.clientid
join workshopreservation as wr
on wr.reservationid = cr.reservationid

--dla warsztatu podaje miejsca dostepne i miejsca zarezerwowane
select w.workshopid,places_available,sum(places_reserved)
from workshop as w join workshopreservation as wr
on w.workshopid = wr.workshopid
group by w.workshopid,places_available

-- najczesciej korzystajacy z uslug - skladajacy najwieksza ilosc
rezerwacji konferencji
select top 1 cr.clientid, count(reservationid)
from clients as c
join conferencereservation as cr
on c.clientid = cr.clientid
group by cr.clientid
order by 2 desc
```

```sql
 -- polecenie, ktore sortuje konferencje  po procentowym udziale miejsc
-- zarezerwowanych we wszystkich dniach konferencji do sumy miejsc
dostepnych
-- w tych dniach
select conferenceid, topic,
convert(real,dbo.conf_places_res(conferenceid))/dbo.conf_places_av(confer
enceid) * 100--places_reserved, places_available--
sum(convert(real,places_reserved)/places_available) * 100 popularity
from conferencies
order by 3 desc


-- polecenie sortujace warsztaty po procentowym stosunku ilosci
zarezerwowanych
-- miejsc do ilosci dostepnych na nim miejsc
select w.workshopid, topic,
sum(places_reserved)/convert(real,places_available) * 100 popularity
from workshop w join workshopreservation wr
on w.workshopid = wr.workshopid
group by w.workshopid, places_available,topic
order by 3 desc

--polecenie pokazujace tych klientow, ktorzy sa firmami
select clientid, dbo.getIndividualOrCompany(clientid),email,password,
country
from clients
where isindividual = 0

--polecenie pokaujace klientow, ktore sa osobami prywatnymi
select clientid, dbo.getIndividualOrCompany(clientid),email,password,
country
from clients
where isindividual = 1

--polecenie pokazujace anulowane rezerwacje konferencji
select c.clientid,dbo.getIndividualOrCompany(c.clientid)
name,reservationid,places_reserved,sum_to_pay,reservationdate,
dayid,email,country
from conferencereservation cr join clients c
on c.clientid = cr.clientid
where iscanceled = 1

--pokaz anulowane rezerwacje konferencji i warsztatow, zestawione razem
select distinct  cr.reservationid,wr.workresid,is_canceled
from conferencereservation as cr join conference_participants as cp
on cp.reservationid = cr.reservationid
join workshopparticipants as wp
on wp.conferenceparticipantid = cp.conferenceparticipantid
join workshopreservation as wr
```

```
on wr.workresid = wp.workresid
where cr.iscanceled = 1
order by 1

--dla kazdego klienta zestawia odpowiednio laczna ilosc dokonanych rez.
konferencji oraz laczna ilosc rez. warsztatow
select distinct c.clientid,count(distinct
cr.reservationid),count(distinct wr.workresid)
from clients as c join conferencereservation as cr
on cr.clientid = c.clientid
join conference_participants as cp
on cp.reservationid = cr.reservationid
join workshopparticipants as wp
on wp.conferenceparticipantid = cp.conferenceparticipantid
join workshopreservation as wr
on wr.workresid = wp.workresid
--where cr.clientid = 0
group by c.clientid
order by 1

--jw, wariant II
select distinct c.clientid,count(distinct cr.reservationid),count(
distinct wr.workresid)
from clients as c join conferencereservation as cr
on cr.clientid = c.clientid
join workshopreservation as wr
on wr.reservationid = cr.reservationid
group by c.clientid
go


--kwota za zarezerwowane warsztaty zestawiona z rezerwacja konferencji
select cr.reservationid,sum(quote *
dbo.check_if_student(wp.conferenceparticipantid) )
from workshopreservation as wr left join workshopparticipants as wp
on wr.workresid = wp.workresid
join conference_participants as cp
on cp.conferenceparticipantid = wp.conferenceparticipantid
join workshop as w
on w.workshopid = wr.workshopid
join conferencereservation as cr
on cr.reservationid = wr.reservationid
group by cr.reservationid
order by 1,2

-- ilosc studentow bioracych udzial w warsztacie
select wr.workresid,count(id_number)
from workshopreservation as wr join workshopparticipants as wp
on wr.workresid = wp.workresid
join conference_participants as cp
on cp.conferenceparticipantid = wp.conferenceparticipantid
```

```sql
group by wr.workresid

--pomocniczy select do testu powyzszego
select *
from workshopreservation as wr join workshopparticipants as wp
on wr.workresid = wp.workresid
join conference_participants as cp
on cp.conferenceparticipantid = wp.conferenceparticipantid
order by 1

-- select zwracajacy cene za konferencje, wziawszy pod uwage date
najblizszego progu cenowego (usunac topa dla testu)
select top 1 conferencePrice,reservationdate, when_is_in_force
from conferencereservation as cr join confrenceday as cd
on cd.dayid = cr.dayid
join fees as f
on f.dayid = cd.dayid
where  cd.dayid = 5 and reservationdate >= when_is_in_force
order by when_is_in_force desc

go

-- zwraca dla reservationid ilosc zarezerwowanych miejsc i ilosc
studentow
select  places_reserved, count(id_number) students
from conferencereservation as cr left join conference_participants as cp
on cp.reservationid = cr.reservationid
where  cr.reservationid = 2
group by cr.reservationid,places_reserved
go


-- dla kazdej rezerwacji pokaz id uczestnikow zarejestrowanych
select cr.reservationid,
places_reserved,cp.conferenceparticipantid,cp.personid
from conferencereservation as cr join conference_participants as cp
on cr.reservationid = cp.reservationid
where cr.reservationid = 0
order by 1,2

-- dla warsztatu pokaz id uczestnikow zarejestrowanych
select wr.workresid, wp.conferenceparticipantid
from workshopreservation as wr join workshopparticipants as wp
on wp.workresid = wr.workresid
join conference_participants as cp
on cp.conferenceparticipantid = wp.conferenceparticipantid
join conferencereservation as cr
on cp.reservationid = cr.reservationid
where cr.reservationid = 512
order by 1,2
```

```
--pokazuje sume zarezerwowanych miejsc w danym dniu
select cd.dayid,date_of_day,places_available,sum(distinct
places_reserved) [suma miejsc kazdej rezerwacji dot. okreslonego dnia]
from confrenceday as cd join conferencereservation as cr
on cr.dayid= cd.dayid
group by cd.dayid,places_available,date_of_day
go


-- pokazuje ilosc zarezerwowanych miejsc w danym warsztacie i zestawia z
dostepnymi miejscami
select w.workshopid,places_available,sum(places_reserved) [suma miejsc
kazdej rezerwacji dot. okreslonego warsztatu]
from workshopreservation as wr join workshop as w
on w.workshopid = wr.workshopid
where w.workshopid = 183
group by w.workshopid,places_available

go


-- wykorzystujac date pokaz najblizsza nadciagajaca konferencje
-- (UWAGA!! Data ma by mniejsza od daty rozpoczecia konferencji!!)
select top 1 conferenceid
from conferencies
where fromdate >= '2000-06-16'
order by fromdate asc

-- select pomocniczy do powyższej procedury
select *
from conferencies as c join confrenceday as cd
on c.conferenceid = cd.conferenceid
join conferencereservation as cr
on cr.dayid = cd.dayid
join paidRate as pr
on pr.reservationid = cr.reservationid
where c.conferenceid = 12
order by 1,clientid,date_of_paying

go
------------------------------------------------------------------------
-----------------------------------------------
--widoki
create view VIPS_reservations
as
     select c.clientid, dbo.getIndividualOrCompany(c.clientid) name,
sum(dbo.get_res_amount(reservationid)) 'amount of reservations'
     from clients c join conferencereservation cr
     on cr.clientid = c.clientid
     group by c.clientid
go
```

```
select * from VIPS_reservations order by 3 desc
go


-- widok => zestawia klienta, jego calkowita kwote do zaplaty za dni
konferencji i warsztaty
-- dodatkowo sume zaplacona do tej pory oraz date rezerwacji zestawiona z
data ostatnio zaplaconej raty

IF OBJECT_ID('costs', 'V') IS NOT NULL
    DROP VIEW costs;
go
create view costs
as
     select c.clientid, sum_to_pay [sum to pay],  sum(amount_paid)
sum_paid,iscanceled,reservationdate, max(date_of_paying) last_payment
     from clients as c join conferencereservation as cr
     on c.clientid = cr.clientid
     join paidRate as pr
     on pr.reservationid = cr.reservationid
     group by c.clientid, sum_to_pay, iscanceled, reservationdate
go

     --test
     select * from costs where iscanceled = 0 and abs([sum to pay] -
sum_paid) > 0.5
     go

     select * from workshop_costs order by 1 go

b) triggery

--trigger ktory uniewaznia transakcje dodania rezerwacji, jesli
zamowienie
--przekroczylo maksymalna ilosc miejsc na dniu konferencji

IF OBJECT_ID ('stop_if_too_many_conf', 'TR') IS NOT NULL
   DROP TRIGGER stop_if_too_many_conf;
GO
CREATE TRIGGER stop_if_too_many_conf on conferencereservation
after INSERT,UPDATE  AS
     if exists
          (select places_available
          from conferencereservation as cr
          join confrenceday as cd
          on cd.dayid = cr.dayid
          group by cd.dayid, places_available
          having places_available < sum(cr.places_reserved) )
     BEGIN
          RAISERROR('More places reserved on conference day than
```

```
available', 16, 1)
      ROLLBACK TRANSACTION END
go

--test triggera
declare @sum real,@date date, @places int
set @places = 20
set @date = '2000-01-06'
set @sum =
      (select top 1 conferencePrice
      from fees as f join confrenceday as cd
      on f.dayid = cd.dayid
      where cd.dayid = 5 and when_is_in_force <= @date
      order by when_is_in_force desc)
insert into conferencereservation values(@places,0,@sum,@date,0,5)
go

select * from conferencereservation
delete from conferencereservation where reservationid = 602

--trigger, ktory dba o to, aby po ewentualnej zmianie ilosci dostepnych
miejsc
--na konferencji suma zarezerwowanych miejsc na dany dzien nie
przekroczyla
-- uaktualnionej liczby miejsc

IF OBJECT_ID ('watch_res_more_than_available', 'TR') IS NOT NULL
   DROP TRIGGER watch_res_more_than_available;
GO
create trigger watch_res_more_than_available
on confrenceday
for update
as
      if exists
            (select cd.places_available
            from confrenceday as cd join conferencereservation as cr
            on cd.dayid = cr.dayid
            group by cd.dayid,cd.places_available
            having cd.places_available < sum(places_reserved))
      BEGIN
            RAISERROR('Less available places than reserved ones', 16, 1)
      ROLLBACK TRANSACTION END
go

update confrenceday
set places_available = 70
where dayid = 0


IF OBJECT_ID ('stop_if_too_many_workres', 'TR') IS NOT NULL
   DROP TRIGGER stop_if_too_many_workres;
```

```
GO
create trigger stop_if_too_many_workres
on  workshopreservation
for insert,update
as
     if exists
          (select w.places_available
          from workshop as w join workshopreservation as wr
          on w.workshopid = wr.workshopid
          group by w.workshopid, places_available
          having places_available < sum(wr.places_reserved))
     BEGIN
          RAISERROR('Less available places than reserved ones', 16, 1)
     ROLLBACK TRANSACTION END
go

insert into workshopreservation values(200,0,82,0)
delete from workshopreservation where workresid = 1794
select *
from conferencereservation cr join workshopreservation wr
on wr.reservationid = cr.reservationid
where workshopid = 0


--trigger, ktory dba o to, aby po ewentualnej zmianie ilosci dostepnych
miejsc
--na warsztacie suma zarezerwowanych miejsc na dany warsztat nie
przekroczyla
-- uaktualnionej liczby miejsc
IF OBJECT_ID ('stop_if_too_little_workres', 'TR') IS NOT NULL
   DROP TRIGGER stop_if_too_little_workres;
GO
create trigger stop_if_too_little_workres
on  workshop
after update
as
     if exists
          (select w.places_available
          from workshop as w join workshopreservation as wr
          on w.workshopid = wr.workshopid
          group by w.workshopid, places_available
          having places_available < sum(wr.places_reserved))
     BEGIN
          RAISERROR('Less available places than reserved ones', 16, 1)
     ROLLBACK TRANSACTION END
go

update workshop
set places_available = 70
where workshopid = 0
```

```
--trigger dbajacy o to, aby osoba nie zapisala sie na ten sam warsztat
dwukrotnie
IF OBJECT_ID ('check_same_workshop_tr ', 'TR') IS NOT NULL
   DROP TRIGGER check_same_workshop_tr ;
GO
create trigger check_same_workshop_tr
on workshopparticipants
for insert
as
     if exists
          (select p.personid
          from workshop as w join workshopreservation as wr
          on wr.workshopid = w.workshopid
          join workshopparticipants as wp
          on wp.workresid = wr.workresid
          join conference_participants as cp
          on cp.conferenceparticipantid = wp.conferenceparticipantid
          join people as p
          on p.personid = cp.personid
          group by w.workshopid,p.personid
          having count(cp.conferenceparticipantid) > 1)
     BEGIN
          RAISERROR('A person already registered for this workshop', 16,
1)
     ROLLBACK TRANSACTION END
go


select p.personid
from workshop as w join workshopreservation as wr
on wr.workshopid = w.workshopid
join workshopparticipants as wp
on wp.workresid = wr.workresid
join conference_participants as cp
on cp.conferenceparticipantid = wp.conferenceparticipantid
join people as p
on p.personid = cp.personid
group by w.workshopid,p.personid
having count(cp.conferenceparticipantid) > 1


insert into workshopparticipants values(45,500)
select * from workshopparticipants
delete from workshopparticipants where workshopparticipantid = 57608

--trigger dbajacy o to, aby osoba nie zapisala sie na warsztat trwajacy w
tym samym czasie

IF OBJECT_ID ('watch_if_same_time_work ', 'TR') IS NOT NULL
   DROP TRIGGER watch_if_same_time_work ;
```

```
GO
create trigger watch_if_same_time_work
on workshopparticipants
for insert,update
as
      if (SELECT COUNT(*)
            FROM  workshopparticipants wp JOIN  workshopreservation wr
            ON wp.workresid = wr.workresid
            INNER JOIN workshop w
            ON w.workshopid = wr.workshopid
            INNER JOIN confrenceday cd
            ON cd.dayid = w.dayid
            INNER JOIN conference_participants cp
            ON cp.conferenceparticipantid = wp.conferenceparticipantid
            INNER JOIN  confrenceday as cd2
            ON cd.dayid = cd2.dayid
            INNER JOIN  workshop w1
            ON w1.dayid = cd2.dayid
            join inserted
            on inserted.workshopparticipantid = wp.workshopparticipantid

            WHERE (w.starting_hour <= w1.end_hour AND w.end_hour >=
w1.starting_hour) and inserted.conferenceparticipantid =
wp.conferenceparticipantid)>1
      BEGIN
           RAISERROR('A person already registered for the workshop
lasting at the same time', 16, 1)
      ROLLBACK TRANSACTION END
go

select
w.dayid,w.workshopid,wr.workresid,workshopparticipantid,starting_hour,end
_hour
from workshopparticipants as wp join workshopreservation wr
on wp.workresid = wr.workresid
join workshop w
on w.workshopid = wr.workshopid
join confrenceday cd
on cd.dayid = w.dayid
where workshopparticipantid = 1884
order by 1,2,3

select w.workshopid, wr.workresid,conferenceparticipantid
from workshopparticipants as wp join workshopreservation wr
on wp.workresid = wr.workresid
join workshop w
on w.workshopid = wr.workshopid
where  w.workshopid = 0 and conferenceparticipantid = 1282

insert into workshopparticipants values(247,1282)
select * from workshopparticipants
```

```
delete from workshopparticipants where workshopparticipantid = 57611


--trigger, ktory pilnuje, zeby zadna osoba nie zapisala sie wiecej niz
raz na dzien konferencji
IF OBJECT_ID ('watch_if_same_day', 'TR') IS NOT NULL
   DROP TRIGGER watch_if_same_day ;
GO
create trigger watch_if_same_day
on conference_participants
for insert,update
as
     if exists
          (select distinct p.personid
          from confrenceday as cd join conferencereservation as cr
          on cd.dayid = cr.dayid
          join conference_participants as cp
          on cp.reservationid = cr.reservationid
          join people p
          on p.personid = cp.personid
          group by cd.dayid,p.personid
          having count(cp.conferenceparticipantid) > 1)
     BEGIN
          RAISERROR('A person already registered for this conference
day', 16, 1)
     ROLLBACK TRANSACTION END
go

select cd.dayid,cr.reservationid,cp.personid
from confrenceday as cd join conferencereservation cr
on cd.dayid = cr.dayid
join conference_participants cp
on cp.reservationid = cr.reservationid
where cd.dayid = 0 and personid = 18
order by 1,2,3

insert into conference_participants values(2,null,82)
select * from conference_participants
delete from conference_participants where conferenceparticipantid = 38606

--trigger, ktory dba o to, aby nie przekroczyc ilosci zarezerwowannych
miesc przy formowaniu listy zarejestrowanych na dzien konferencji
IF OBJECT_ID ('watch_if_places_res', 'TR') IS NOT NULL
   DROP TRIGGER watch_if_places_res;
GO
create trigger watch_if_places_res
on conference_participants
for insert,update
as
     if exists
          (select places_reserved
```

```
          from conferencereservation cr join conference_participants cp
          on cp.reservationid = cr.reservationid
          group by cr.reservationid,places_reserved
          having count(personid) > places_reserved )
     BEGIN
          RAISERROR('Limit of places reserved has been reached, you
cannot register through this reservation', 16, 1)
     ROLLBACK TRANSACTION END
go

select cr.reservationid, places_reserved
from conferencereservation cr join conference_participants cp
on cp.reservationid = cr.reservationid
group by cr.reservationid,places_reserved

insert into conference_participants values(200,null,0)


-- trigger sprawdzajacy, czy po zmianie zarezerwowanych miejsc nie
dojdzie do
-- przepelnienia list osobowych
IF OBJECT_ID ('watch_if_places_res_change', 'TR') IS NOT NULL
   DROP TRIGGER watch_if_places_res_change;
GO
create trigger watch_if_places_res_change
on conferencereservation
for update
as
     if exists
          (select places_reserved
          from conferencereservation cr join conference_participants cp
          on cp.reservationid = cr.reservationid
          group by cr.reservationid,places_reserved
          having count(personid) > places_reserved )
     BEGIN
          RAISERROR('Cannot change  reserved places amount to maintain
consistence', 16, 1)
     ROLLBACK TRANSACTION END
go

-- test
update conferencereservation
set places_reserved = 5
where reservationid = 0

-- trigger dbajacy o to, by nie przekroczono limitu zarezerwowanych
miejsc
-- dla danej rezerwacji warsztatu
IF OBJECT_ID ('watch_if_places_work', 'TR') IS NOT NULL
   DROP TRIGGER watch_if_places_work;
GO
```

```
create trigger watch_if_places_work
on workshopparticipants
for insert,update
as
      if exists
            (select places_reserved
            from workshopreservation wr join workshopparticipants wp
            on wp.workresid = wr.workresid
            group by wr.workresid,places_reserved
            having count(conferenceparticipantid) > places_reserved )
      BEGIN
            RAISERROR('Limit of places reserved has been reached, you
cannot register through this reservation', 16, 1)
      ROLLBACK TRANSACTION END
go

select wr.workresid, places_reserved, workshopparticipantid
from workshopreservation wr join workshopparticipants wp
on wr.workresid = wp.workresid
--group by wr.workresid,places_reserved

insert into workshopparticipants values(0,200)

-- trigger dbajacy o to, by po zmianie ilosci zarezerwowanych miejsc w
rezerwacji warsztatu nie nastapila
-- utrata spojnosci danych
IF OBJECT_ID ('watch_if_places_work_change', 'TR') IS NOT NULL
   DROP TRIGGER watch_if_places_work_change;
GO
create trigger watch_if_places_work_change
on workshopreservation
for update
as
      if exists
            (select places_reserved
            from workshopreservation wr join workshopparticipants wp
            on wp.workresid = wr.workresid
            group by wr.workresid,places_reserved
            having count(conferenceparticipantid) > places_reserved )
      BEGIN
            RAISERROR('Change canceled in order to maintain data
consistency on workshop reservation', 16, 1)
      ROLLBACK TRANSACTION END
go

update workshopreservation
set places_reserved = 1
where workresid = 0


--trigger pilnujacy, czy osoba pragnaca zarejestrowac sie na warsztat
```

```
jest zapisana
--tego dnia na konferencje

IF OBJECT_ID ('watch_if_registered_onday_work', 'TR') IS NOT NULL
   DROP TRIGGER watch_if_registered_onday_work;
GO
create trigger watch_if_registered_onday_work
on workshopparticipants
for insert,update
as
      if exists
            (select personid
            from inserted as i join conference_participants cp
            on i.conferenceparticipantid =  cp.conferenceparticipantid
            join  workshopreservation wr
            on i.workresid = wr.workresid
            join workshop w
            on w.workshopid = wr.workshopid
            join conferencereservation cr
            on cr.reservationid = cp.reservationid
            where w.dayid != cr.dayid)
      BEGIN
            RAISERROR('This person is not registered on the conference day
when this workshop takes place', 16, 1)
      ROLLBACK TRANSACTION END
go

select *
from conference_participants cp join conferencereservation cr
on cp.reservationid = cr.reservationid
join workshopreservation wr
on wr.reservationid = cr.reservationid
where dayid = 1

insert into workshopparticipants values(258,217)
select * from workshopparticipants
delete from workshopparticipants where workshopparticipantid = 57620

select * from conference_participants where personid = 183

--trigger, ktory pilnuje, czy wplacona rata nie bedzie przekraczala
naleznej sumy
IF OBJECT_ID ('watch_if_paid_too_much', 'TR') IS NOT NULL
   DROP TRIGGER watch_if_paid_too_much;
GO
create trigger watch_if_paid_too_much
on paidRate
for insert
as
      if exists
            (select sum_to_pay
```

```
            from paidRate pr join conferencereservation cr
            on cr.reservationid = pr.reservationid
            group by cr.reservationid, sum_to_pay
            having sum_to_pay < sum(amount_paid))
      BEGIN
            RAISERROR('Client paid more than it is worth', 16, 1)
      ROLLBACK TRANSACTION END
go

select *
from conferencereservation cr join paidrate pr
on pr.reservationid = cr.reservationid
where iscanceled = 0

insert into paidrate values(0,'2000-06-29',200)
```

c) funkcje i procedury:

```
      --funkcja zwracajaca sume miejsc dostepnych we wszystkich dniach
konferencji o
--danym id
IF OBJECT_ID ('conf_places_av') IS NOT NULL
   DROP function conf_places_av;
GO
create function conf_places_av(@conferenceid int)
returns int
begin
      return
            (select sum(places_available)
            from confrenceday
            where conferenceid = @conferenceid)
end
go

-- funkcja zwrcajaca sume miejsc zarezerwowanych w obrebie wszystkich
-- dni danej konferencji
IF OBJECT_ID ('conf_places_res') IS NOT NULL
   DROP function conf_places_res;
GO
create function conf_places_res(@conferenceid int)
returns int
begin
      return
            (select sum(places_reserved)
            from confrenceday cd join conferencereservation cr
            on cr.dayid = cd.dayid
            where conferenceid = @conferenceid)
end
go

-- funkcja zwracająca ilość rezerwacji warsztatów, łącznie z
```

```
odpowiadającą im rezerwacją konferencji
IF OBJECT_ID ('get_res_amount') IS NOT NULL
   DROP function get_res_amount;
GO
create function get_res_amount(@resid int)
returns int
begin
      return
            (select count(workresid) + 1 -- 1 oznacza rezerwacje
konferencji
            from workshopreservation
            where reservationid = @resid)
end
go


-- funkcja ktora osadza, czy uzytkownik o id participanta jest studentem
czy nie - gdy jest, zwraca 0.5, czyli polowe stawki do zaplacenia, gdy
nie jest, zwraca 1, czyli cala stawke
IF OBJECT_ID('check_if_student') IS NOT NULL
   DROP FUNCTION check_if_student
GO
create function check_if_student(@conf_part_id int)
returns real
as
begin
      if
      (select id_number
      from conference_participants
      where @conf_part_id = conferenceparticipantid) is null
            return 1
      return 0.5
end
go

      --testy
      select * from conference_participants
      select dbo.check_if_student(1)
      go

-- funkcja zwracajaca dane klienta, ktore nie sa nullem
IF OBJECT_ID('getIndividualOrCompany') IS NOT NULL
   DROP function getIndividualOrCompany
GO
create function getIndividualOrCompany(@clientid int)
returns varchar(100)
as
begin
      declare @isindividual bit,  @companyname varchar(50), @name_surname
varchar(50)
      set @companyname =
```

```
            (select companyname
            from clients
            where @clientid = clientid)
      set @name_surname =
            (select name + ' ' + surname
            from clients
            where @clientid = clientid)
      set @isindividual =
            (select isindividual
            from clients
            where @clientid = clientid)
      if @isindividual = 1 return @name_surname
      return @companyname
end
go

      select * from clients
      select dbo.getIndividualOrCompany(1)
      select dbo.getIndividualOrCompany(22)
go

--funkcja zwrcajaca id warsztatu trwajacego w tym samym czasie co
warsztat o id podanym na wejsciu
IF OBJECT_ID('getSameWorkshopsID') IS NOT NULL
   DROP function getSameWorkshopsID
GO
create function getSameWorkshopsID (@dayid int, @workshopId int)
returns int
as
begin
      declare @startime time, @endtime time
      set @startime =
            (select starting_hour
            from workshop
            where workshopid = @workshopId)
      set @endtime =
            (select end_hour
            from workshop
            where workshopid = @workshopId)
      return
            (select workshopid
            from workshop
            where @workshopId != workshopid
            and end_hour = @endtime
            and starting_hour = @startime
            and @dayid = dayid)
end
go

select * from workshop where dayid = 159
select dbo.getSameWorkshopsID (0, 0)
```

```
go

-------------------------------------------------------------------------
--------------------------------
--procedury

--procedura pokazujaca dla danego dayid progi cenowe
create procedure show_fees @dayid int
as
      select *
      from fees
      where dayid = @dayid
go


-- procedura => oblicz calosc naleznosci za rezerwacje dnia konferencji i
zwiazanych z dniem warsztatow i zestaw to z polem sum_to_pay w conf.res.
IF OBJECT_ID('total_cost') IS NOT NULL
   DROP procedure total_cost
GO
create procedure total_cost @resId int
as
      declare @fee_id int, @price real, @unitprice real, @places int,
@wr_places int, @student_places int, @w_price real, @sum real,@select_sum
real
      set @unitprice =
            (select top 1 conferencePrice
            from conferencereservation as cr join confrenceday as cd
            on cd.dayid = cr.dayid
            join fees as f
            on f.dayid = cd.dayid
            where reservationdate >= when_is_in_force and cr.reservationid
= @resId
            order by when_is_in_force desc)
      set @student_places =
            (select  count(conferenceparticipantid) students
            from conferencereservation as cr left join
conference_participants as cp
            on id_number is not null and cp.reservationid =
cr.reservationid
            where  cr.reservationid = @resId
            group by cr.reservationid)
      set @places =
            (select places_reserved
            from conferencereservation
            where reservationid = @resId) - @student_places
      set @price = @unitprice * @places + 0.5 * @unitprice *
@student_places
      set @w_price =
            (select sum(quote *
dbo.check_if_student(wp.conferenceparticipantid) )
```

```
            from workshopreservation as wr left join workshopparticipants
as wp
            on wr.workresid = wp.workresid
            join conference_participants as cp
            on cp.conferenceparticipantid = wp.conferenceparticipantid
            join workshop as w
            on w.workshopid = wr.workshopid
            join conferencereservation as cr
            on cr.reservationid = wr.reservationid
            where cr.reservationid = @resId
            group by cr.reservationid)
      set @sum = @price + @w_price
      set @select_sum =
            (select sum_to_pay
            from conferencereservation
            where reservationid = @resId)
      select @resId [reservation id],@select_sum [select sum], @sum
[procedure]
go

      -- test
      execute total_cost 0
      go


-- procedura pokazujaca warsztaty trwajace w tym samym czasie w danym
dniu co warsztat o id podanym na wejsciu

IF OBJECT_ID('getSameWorkshops') IS NOT NULL
   DROP procedure getSameWorkshops
GO
create procedure getSameWorkshops @dayid int, @workshopId int
as
      declare @startime time, @endtime time
      set @startime =
            (select starting_hour
            from workshop
            where workshopid = @workshopId)
      set @endtime =
            (select end_hour
            from workshop
            where workshopid = @workshopId)
      print @endtime
      print @startime
      select *
      from workshop
      where @workshopId != workshopid
      and end_hour = @endtime
      and starting_hour = @startime
      and @dayid = dayid
go
```

```
      -- test getSameWorkshops
      select * from workshop where dayid = 159
      execute getSameWorkshops 159, 477
      go


--procedura => lista osobowa uczestnikow na kazdy dzien konferencji
IF OBJECT_ID('daylist') IS NOT NULL
   DROP procedure daylist
GO
create procedure daylist @dayid int
as
select  @dayid,surname,name, phone, country,city,p.street
from confrenceday as cd join conferencereservation cr
on cd.dayid = cr.dayid
join conference_participants as cp
on cp.reservationid = cr.reservationid
join People as p
on cp.personid = p.personid
where cd.dayid = @dayid
order by 1,2
go

      --test
      execute daylist 200

-- procedura zalatwiajaca 'informacje o kliencie' w poleceniu projektu
IF OBJECT_ID('generate_business_card') IS NOT NULL
   DROP procedure generate_business_card
GO

create procedure generate_business_card @dayid int
as
      select  @dayid,p.surname,p.name,
dbo.getIndividualOrCompany(c.clientid) info_about_client
      from confrenceday as cd join conferencereservation cr
      on cd.dayid = cr.dayid
      join conference_participants as cp
      on cp.reservationid = cr.reservationid
      join People as p
      on cp.personid = p.personid
      join clients c
      on c.clientid = cr.clientid
      where cd.dayid = @dayid and iscanceled = 0
      order by 1,2
go

      execute generate_business_card 200

--procedura pokazujaca liste osobowa uczestnikow warsztatu o danym id
```

```
IF OBJECT_ID('workshop_list') IS NOT NULL
   DROP procedure workshop_list
GO
create procedure workshop_list @workshopid int
as
      select @workshopid,surname,name, phone, country,city,p.street
      from workshop as w join workshopreservation as wr
      on w.workshopid = wr.workshopid
      join workshopparticipants as wp
      on wp.workresid = wr.workresid
      join conference_participants as cp
      on wp.conferenceparticipantid = cp.conferenceparticipantid
      join People as p
      on p.personid = cp.personid
      where w.workshopid = @workshopid
go

      execute workshop_list 183

-- procedura, ktora dla danego id rezerwacji pokazuje liste osobowa
uczestnikow zgloszonych + informacja o zglaszajacym
IF OBJECT_ID('reservation_list') IS NOT NULL
   DROP procedure reservation_list
GO
create procedure reservation_list @reservationid int
as
      select p.surname participant_surname, p.name
participant_name,places_reserved, dbo.getIndividualOrCompany(c.clientid)
client_name, isindividual 'registered by individual person?'
      from conference_participants cp join people p
      on p.personid = cp.personid
      join conferencereservation cr
      on cr.reservationid = cp.reservationid
      join clients c
      on c.clientid = cr.clientid
      where cr.reservationid = @reservationid
go

      execute reservation_list 0

--procedura, ktora dla danego id rezerwacji warsztatu pokazuje liste
osobowa uczestnikow warsztatu + informacje i zglaszajacym
IF OBJECT_ID('workres_list') IS NOT NULL
   DROP procedure workres_list
GO
create procedure workres_list @workresid int
as
      select p.surname 'participant surname', p.name 'participant name',
wr.places_reserved, dbo.getIndividualOrCompany(c.clientid) client_name,
isindividual 'Registered by an individual person?'
      from workshopparticipants wp join conference_participants cp
```

```
      on wp.conferenceparticipantid = cp.conferenceparticipantid
      join people p
      on p.personid = cp.personid
      join conferencereservation cr
      on cr.reservationid = cp.reservationid
      join clients c
      on c.clientid= cr.clientid
      join workshopreservation wr
      on wr.workresid = wp.workresid
      where wr.workresid = @workresid
go

      execute workres_list 0

-- pokaz klientow, ktorzy maja zaleglosci w platnosciach, na wejsciu
ilosc dni od daty rezerwacji i aktualna data(punkt odniesienia),
-- dzieki ktorej zostanie odszukana najblizsza konferencja
IF OBJECT_ID('delayed') IS NOT NULL
   DROP procedure delayed
GO
create procedure delayed @days_amount int, @recent_date date
as
      declare @recent_conferencyid int

      set @recent_conferencyid =
            (select top 1 conferenceid
            from conferencies
            where fromdate >= @recent_date
            order by fromdate asc)

      select c.conferenceid,dbo.getIndividualOrCompany(clientid),
               reservationdate,max(sum_to_pay) 'to pay',
               isnull(sum(amount_paid),0) 'already paid',
               iscanceled,dateadd(d, @days_amount, reservationdate)
      from conferencies as c join confrenceday as cd
      on c.conferenceid = cd.conferenceid
      join conferencereservation as cr
      on cr.dayid = cd.dayid
      left join paidRate as pr
      on pr.reservationid = cr.reservationid and date_of_paying <=
dateadd(d, @days_amount, reservationdate)
      where c.conferenceid = @recent_conferencyid and  dateadd(d,
@days_amount, reservationdate) = @recent_date
      group by c.conferenceid,clientid,reservationdate,iscanceled
      having abs(isnull(sum(amount_paid),0) - max(sum_to_pay) ) >0.5

go

      execute delayed 2,'2000-06-15'

-- procedura pokazujaca dane klientow, ktorzy siedem dni od daty
```

```
rezerwacji nie uregulowali
-- naleznosci sum_to_pay, na wejsciu data, ktora bedzie punktem
odniesienia pokazujaca identyfikujaca najblizsza konferencje
IF OBJECT_ID('really_delayed') IS NOT NULL
   DROP procedure really_delayed
GO
create procedure really_delayed @recent_date date
as
      declare @recent_conferencyid int
      set @recent_conferencyid =
            (select top 1 conferenceid
            from conferencies
            where fromdate >= @recent_date
            order by fromdate asc)
      select c.conferenceid,dbo.getIndividualOrCompany(clientid),
               reservationdate,max(sum_to_pay) 'to pay',
               isnull(sum(amount_paid),0) 'already paid',
               iscanceled
      from conferencies as c join confrenceday as cd
      on c.conferenceid = cd.conferenceid
      join conferencereservation as cr
      on cr.dayid = cd.dayid
      left join paidRate as pr
      on pr.reservationid = cr.reservationid and date_of_paying <=
dateadd(d, 7, reservationdate) -- 7 dni od rezerwacji jest ostatecznym
progiem, pozniejsze oplaty nie sa brane pod uwage
      where c.conferenceid = @recent_conferencyid and  dateadd(d, 7,
reservationdate) <= @recent_date
      group by c.conferenceid,clientid,reservationdate,iscanceled
      having abs(isnull(sum(amount_paid),0) - max(sum_to_pay) ) >0.5

go

      execute  really_delayed '2002-09-23'

--procedura pokazujaca id warsztatow na ktory uczestnik byl kiedykolwiek
zapisany + informacje o nim i zestawia to z id dnia konferencji, na
wejsciu id uczestnika
IF OBJECT_ID('my_events') IS NOT NULL
   DROP procedure my_events
GO
create procedure my_events @personid int
as
      select w.workshopid, topic, street, starting_hour,end_hour, quote,
w.dayid,places_available
      from conference_participants cp join conferencereservation cr
      on cp.reservationid = cr.reservationid
      join workshopparticipants wp
      on wp.conferenceparticipantid = cp.conferenceparticipantid
      join workshopreservation wr
      on wr.workresid = wp.workresid
```

```
      join workshop w
      on w.workshopid = wr.workshopid
      where @personid = personid
      order by 1
go

      execute my_events 0

--procedura pokazujaca dane klientow, ktorzy na dwa tygodnie przed data
najblizszej
--konferencji (wyznaczonej przez date na wejsciu) nie uzupelnili
wszystkich sanych osobowych uczestnikow
IF OBJECT_ID('show_delaying_data') IS NOT NULL
   DROP procedure show_delaying_data
GO
create procedure show_delaying_data @recent_date date
as
SELECT cr.reservationid, dbo.getIndividualOrCompany(c.clientid)
'Name', c.email, co.topic AS 'Conferency Subject',
        cr.places_reserved AS 'Places reserved',  cr.places_reserved -
(SELECT COUNT(*)

                           FROM conference_participants AS cp

                           WHERE cp.reservationid = cr.reservationid) AS
'How many lacking'
FROM conferencereservation cr LEFT JOIN clients c
ON cr.clientid = c.clientid
LEFT JOIN confrenceday cd
ON cd.dayid = cr.dayid
LEFT JOIN conferencies co
ON co.conferenceid = cd.conferenceid
WHERE   cr.iscanceled = 0
          AND (cr.places_reserved - (SELECT COUNT(*)
                                        FROM conference_participants cp
                                     WHERE cp.reservationid = cr.reser
vationid)) > 0
          AND (cd.date_of_day >= DATEADD(d,14,@recent_date) )
go

      execute show_delaying_data '2000-01-01'

--zestawia id rezerwacji konferencji, ilosc zarezerwowanych miejsc i id
uczestnikow
select  cr.reservationid, places_reserved, conferenceparticipantId
from conferencereservation as cr join conference_participants as cp
on cr.reservationid = cp.reservationid
where cr.reservationid = 0

--jw
select  distinct cr.reservationid,wr.workresid, wr.places_reserved,
```

```
wp.conferenceparticipantId,workshopparticipantid
from conferencereservation as cr join conference_participants as cp
on cr.reservationid = cp.reservationid
join workshopreservation wr
on wr.reservationid = cr.reservationid
join workshopparticipants as wp
on wp.workresid = wr.workresid
where cr.reservationid = 12
order by 1,2,3,4

--procedura pokazujaca osobie, w ktorych warsztatach uczestniczy
IF OBJECT_ID ('show_workshop_person') IS NOT NULL
   DROP procedure show_workshop_person;
GO
create procedure show_workshop_person @id int
as
      select @id,w.workshopid
      from workshop as w join workshopreservation wr
      on w.workshopid = wr.workshopid
      join workshopparticipants wp
      on wp.workresid = wr.workresid
      join conference_participants as cp
      on cp.conferenceparticipantid = wp.conferenceparticipantid
      where personid = @id
go

execute show_workshop_person 183
select *
from workshopreservation as wr
where workshopid = 1


select *
from conference_participants cp join workshopparticipants wp
on cp.conferenceparticipantid = wp.conferenceparticipantid
join workshopreservation as wr
on wr.workresid = wp.workresid
where personid = 183

--procedura pokazujaca osobie, w ktorych dniach konferencji uczestniczy
IF OBJECT_ID ('show_day_person') IS NOT NULL
   DROP procedure show_day_person;
GO
create procedure show_day_person @id int
as
      select @id,cd.dayid
      from confrenceday as cd join conferencereservation cr
      on cd.dayid= cr.dayid
      join conference_participants cp
      on cp.reservationid = cr.reservationid
      where personid = @id
```

```
go

execute show_day_person 183

--procedura, ktora wypelnia tabelke conferencereservation, jesli trigger
nie zabroni
IF OBJECT_ID ('fill_conf_res') IS NOT NULL
   DROP procedure fill_conf_res;
GO
create procedure fill_conf_res @placesreserved int,@reservationdate
date,@clientid int, @dayid int
as
      declare @sum real,@unitprice real

      set @unitprice =
            (select top 1 conferencePrice
            from fees f join confrenceday cd
            on cd.dayid = f.dayid
            where f.dayid = @dayid and @reservationdate >=
when_is_in_force
            order by when_is_in_force desc)
      set @sum = @unitprice * @placesreserved
      insert into conferencereservation
values(@placesreserved,0,@sum,@reservationdate,@clientid,@dayid)
go

select * from conferencereservation
delete from conferencereservation where reservationid = 603
execute fill_conf_res 20,'2000-01-06',0,5

--procedura, ktora wypelnia tabelke workshopreservation, jesli trigger
nie zabroni
IF OBJECT_ID ('fill_work_res') IS NOT NULL
   DROP procedure fill_work_res;
GO
create procedure fill_work_res @placesreserved int,@workshopid
int,@reservationid int
as
      declare @sum real,@quote real
      set @quote =
            (select quote
            from workshop
            where workshopid = @workshopid)
      set @sum = @quote * @placesreserved
      insert into workshopreservation
values(@placesreserved,@workshopid,@reservationid,0)
      if @@error = 0
            update conferencereservation
            set sum_to_pay += @sum
            where reservationid = @reservationid
go
```

```
-- procedura dodajaca uczestnika rezerwacji, do danych nalezy podac date,
w ktorej
-- go zarejestrowano. Jesli data przekracza o tydzien date rezerwacji, a
uczestnik
-- jest studentem, suma nie ulegnie zmianie

IF OBJECT_ID ('fill_conf_partic') IS NOT NULL
   DROP procedure fill_conf_partic;
GO
create procedure fill_conf_partic @personid int, @id_number char(6) =
Null, @reservationid int, @date_of_adding date
as
      declare @unitprice real,@res_date date,@dayid int
      insert into conference_participants
values(@personid,@id_number,@reservationid)
      set @res_date =
                  (select reservationdate
                  from conferencereservation
                  where reservationid = @reservationid)
      if @@error = 0 and @id_number is not null and @date_of_adding <=
dateadd(d,7,@res_date)
      begin
            set @dayid =
                  (select dayid
                  from conferencereservation
                  where reservationid = @reservationid)
            set @unitprice =
                  (select top 1 conferencePrice
                  from fees f join confrenceday cd
                  on cd.dayid = f.dayid
                  where f.dayid = @dayid and @res_date >= when_is_in_force
                  order by when_is_in_force desc)
            update conferencereservation
            set sum_to_pay -= @unitprice * 0.5
            where reservationid = @reservationid
      end
go

-- procedura dodajaca uczestnika warsztatu, do danych nalezy podac date,
w ktorej
-- go zarejestrowano. Jesli data przekracza o tydzien date rezerwacji, a
uczestnik
-- jest studentem, suma nie ulegnie zmianie

IF OBJECT_ID ('fill_work_partic') IS NOT NULL
   DROP procedure fill_work_partic;
GO
create procedure fill_work_partic @workresid int,
@conferenceparticipantid int,@date_of_adding date
```

```
as
      declare @reservationid int,@unitprice real,@res_date date,@dayid
int,@id_number char(6)
      insert into workshopparticipants
values(@workresid,@conferenceparticipantid)
      set @reservationid =
            (select reservationid
            from conference_participants
            where conferenceparticipantid = @conferenceparticipantid)
      set @res_date =
            (select reservationdate
            from conferencereservation
            where reservationid = @reservationid)
      set @id_number =
            (select id_number
            from conference_participants cp
            where conferenceparticipantid = @conferenceparticipantid)
      if @@error = 0 and @id_number is not null and @date_of_adding <=
dateadd(d,7,@res_date)
      begin
            set @dayid =
                  (select dayid
                  from workshopreservation wr join workshop w
                  on w.workshopid = wr.workshopid
                  where workresid = @workresid)
            set @unitprice =
                  (select top 1 conferencePrice
                  from fees f join confrenceday cd
                  on cd.dayid = f.dayid
                  where f.dayid = @dayid and @res_date >= when_is_in_force
                  order by when_is_in_force desc)
            update conferencereservation
            set sum_to_pay -= @unitprice * 0.5
            where reservationid = @reservationid
      end
go

-- procedura pokazujaca platnosci klientow, ktorzy zlozyli rezerwacje na
ktorys z dni konferencji, ktorej id podajemy na wejsciu;
-- drugim argumentem jest data, ktora wskazuje na moment, w ktorym chcemy
poznac stan platnosci klientow
IF OBJECT_ID ('show_payments_for_conf') IS NOT NULL
   DROP procedure show_payments_for_conf;
GO
create procedure show_payments_for_conf @conferenceid int,@recent_date
date
as
      select c.clientid,co.conferenceid,
dbo.getIndividualOrCompany(c.clientid) 'client name', reservationdate,
            sum_to_pay,
            (select sum(isnull(amount_paid,0))
```

```
            from conferencereservation cr1 left join paidRate pr
            on pr.reservationid = cr1.reservationid and date_of_paying
<= @recent_date
            where cr1.clientid = c.clientid and cr1.reservationid =
cr.reservationid) paid,
            sum_to_pay -
                  (select sum(isnull(amount_paid,0))
                  from conferencereservation cr1 left join paidRate pr
                  on pr.reservationid = cr1.reservationid and
date_of_paying <= @recent_date
                  where cr1.clientid = c.clientid and cr1.reservationid
= cr.reservationid) how_much_left
      from conferencereservation as cr join clients c
      on c.clientid = cr.clientid and reservationdate <= @recent_date
      join confrenceday cd
      on cd.dayid = cr.dayid
      join conferencies co
      on co.conferenceid = cd.conferenceid
      where co.conferenceid = @conferenceid
go

execute show_payments_for_conf 1,'2000-01-10'



--procedura pokazujaca typ kolumny w tabeli
IF OBJECT_ID('getDataType') IS NOT NULL
   DROP procedure getDataType
GO
create procedure getDataType @table varchar(50), @column varchar(50)
as
      SELECT DATA_TYPE
      FROM INFORMATION_SCHEMA.COLUMNS
      WHERE
            TABLE_NAME = @table AND
            COLUMN_NAME = @column
go
      --test getDataType
      execute getDataType 'workshop','end_hour'
go

--Addclients
create  PROCEDURE [dbo].[Addclients]
      @companyname varchar(80),
      @password varchar(50),
      @surname varchar(50),
      @name varchar(50),
      @email varchar(50),
      @isindividual bit,
      @country varchar (50)
```

```
AS

BEGIN
SET NOCOUNT ON
INSERT INTO clients(
companyname ,
password ,
surname,
name ,
email ,
isindividual ,
country
 )
VALUES(
@companyname ,
@password ,
@surname,
@name ,
@email ,
@isindividual ,
@country

)
END



--Addconferencies
go
create  PROCEDURE [dbo].[Addconferencies]

      @fromdate date,
      @todate date,
      @topic varchar(50),
      @street varchar(50),
      @city varchar(50)



AS

BEGIN
SET NOCOUNT ON
INSERT INTO conferencies(
      fromdate,
      todate  ,
      topic   ,
      street  ,
      city
```

```
)

VALUES(
      @fromdate,
      @todate ,
      @topic ,
      @street ,
      @city

)
END


--Addconfrenceday
go
create procedure [dbo].[Addconfrenceday]
      @places_available int,
      @conferenceid int,
      @date_of_day date

AS
BEGIN
SET NOCOUNT ON

INSERT INTO confrenceday(
      places_available ,
      conferenceid,
      date_of_day)

VALUES(

      @places_available ,
      @conferenceid,
      @date_of_day

)


END


go
--Addfees
create  PROCEDURE [dbo].[Addfees]
          @when_is_in_force date,
          @dayid int,
          @conferencePrice real


AS
```

```
BEGIN
SET NOCOUNT ON
INSERT INTO fees(when_is_in_force,dayid,conferencePrice)
VALUES(
      @when_is_in_force ,
      @dayid ,
      @conferencePrice
)
END


--AddpaidRate
go
create  PROCEDURE [dbo].[AddpaidRate]
           @reservationid int,
           @date_of_paying date,
           @amount_paid real


AS

BEGIN
SET NOCOUNT ON
INSERT INTO paidRate(reservationid,date_of_paying,amount_paid)
VALUES(
@reservationid,
@date_of_paying,
@amount_paid
)
END

go
--AddPeople

create PROCEDURE [dbo].[AddPeople]
 @surname varchar(15),
 @name varchar(15),
 @phone char(14),
 @country varchar(50),
 @city varchar(50),
 @street varchar(50)


AS
BEGIN
SET NOCOUNT ON
INSERT INTO People(surname,name,phone,country,city,street)
VALUES(
      @surname ,
      @name ,
      @phone ,
```

```
      @country,
      @city ,
      @street
)
END

--Addworkshop
go
create  PROCEDURE [dbo].[Addworkshop]
      @quote real ,
      @street varchar(50),
      @end_hour time,
      @starting_hour time,
      @topic varchar(50),
      @dayid int,
      @places_available int

AS

BEGIN
SET NOCOUNT ON

INSERT INTO workshop(

quote,
street,
end_hour,
starting_hour,
topic,
dayid,
places_available

 )
VALUES(
      @quote,
      @street,
      @end_hour,
      @starting_hour,
      @topic,
      @dayid,
      @places_available

)
END

---updatePeople
----People
---()Zmien_dane
go
CREATE PROCEDURE [dbo].[UpdatePeople]
```

```
      @personid int              = NULL,/* po tym aktualizujemy  */
      @surname varchar(15)  = NULL,
      @name varchar(15)          = NULL,
      @phone char(14)       = NULL,
      @country varchar(50)  = NULL,
      @city varchar(50)          = NULL,
      @street varchar(50)        = NULL


AS
BEGIN
      SET NOCOUNT ON


      UPDATE dbo.People
      SET
                  surname    =ISNULL(@surname,surname)   ,
                  name          =ISNULL(@name,name)         ,
                  phone     =ISNULL(@phone,phone)       ,
                  country    =ISNULL(@country,country)   ,
                  city       =ISNULL(@city,city)         ,
                  street    =ISNULL(@street,street)


      FROM   dbo.People
      WHERE
      personid=@personid

END
go

--Update_places_reserved_IN_workshopreservation
--()Zmien_ilosc_miejsc_na_warsztaty

create procedure [dbo].[Update_places_reserved_IN_workshopreservation]
      @workresid int ,
      @places_reserved int
AS
BEGIN
      SET NOCOUNT ON
      UPDATE dbo.workshopreservation
      SET places_reserved=@places_reserved
      FROM dbo.workshopreservation
      where workresid=@workresid
END
go

--Update
--Zmien_ilosc_miejsc_w_rezerwacji(
create procedure [dbo].[Update_palces_reserved_IN_conferencereservation]
```

```
      @reservationid int,
      @places_reserved int

AS
BEGIN
      SET NOCOUNT ON
      UPDATE dbo.conferencereservation
      SET   places_reserved=@places_reserved
      FROM dbo.conferencereservation
      where reservationid=@reservationid
END
go

-- Zmien_limit_miejsc_dnia_konf(
--Update_places_available_IN_confrenceday
create PROCEDURE [dbo].[Update_places_available_IN_confrenceday]
      @dayid int ,
      @places_available int
AS
BEGIN
      SET NOCOUNT ON
      UPDATE dbo.confrenceday
      SET places_available=@places_available
      FROM dbo.confrenceday
      WHERE dayid=@dayid
END
go

--Zmien_limit_miejsc_warsztatu(
--Update_places_available_IN_workshop

create PROCEDURE [dbo].[pdate_places_available_IN_workshop]
      @workshopid int ,
      @places_available int

AS
BEGIN
      SET NOCOUNT ON
      UPDATE dbo.workshop
      SET
      places_available=@places_available
      FROM dbo.workshop
      WHERE  workshopid=@workshopid
END
go



--Cancel_conferencereservation
create PROCEDURE [dbo].[Cancel_conferencereservation] @reservationid int
AS
```

```
BEGIN
	SET NOCOUNT ON
	--- anuluj podlegle workshopreservation
	UPDATE  dbo.workshopreservation
	SET is_canceled=1
	FROM  dbo.workshopreservation
	WHERE reservationid=@reservationid
	--anuluj conferencereservation
	UPDATE  dbo.conferencereservation
	SET iscanceled=1
	FROM  dbo.conferencereservation
	WHERE reservationid=@reservationid

END

go

--Cancel_workshopreservation
create PROCEDURE [dbo].[Cancel_workshopreservation] @workresid int
AS
BEGIN
	SET NOCOUNT ON
	UPDATE  dbo.workshopreservation
	SET is_canceled=1
	FROM  dbo.workshopreservation
	WHERE workresid=@workresid
END
go

--Restore_conferencereservation
create PROCEDURE [dbo].[Restore_conferencereservation]
@reservationid int
AS
BEGIN
	SET NOCOUNT ON
	UPDATE  dbo.conferencereservation
	SET iscanceled=0
	FROM  dbo.conferencereservation
	WHERE reservationid=@reservationid
	--przywroc podlegle rezerwaje warsztatu
	update workshopreservation
	set is_canceled = 0
	where reservationid = @reservationid
END
go

--Restore_workshopreservation
create PROCEDURE [dbo].[Restore_workshopreservation]
@workresid int
AS
BEGIN
```

```
	SET NOCOUNT ON
	if
		(select iscanceled
		from dbo.conferencereservation
		where reservationid=(
					select reservationid
					from dbo.workshopreservation
					where workresid=@workresid))=0

		BEGIN
				UPDATE  dbo.workshopreservation
				SET is_canceled=0
				FROM  dbo.workshopreservation
				WHERE workresid=@workresid
		END
END
go
```

6. Uczestnicy, role, uprawnienia

```
DROP role Administrator
CREATE ROLE Administrator
/*
Administrator bazy ma dostęp do wszystkich jej składników
*/

drop role Owner
CREATE ROLE Owner
/*Owner (orgnizator) - może tworzyć konferencje i warsztaty, dostęp do
wszystkich widoków i procedur związanych z
utworzonymi konferencjami + wszystko to co WebApiCustomerSupport)
*/
drop role WebApiCustomerSupport
create role WebApiCustomerSupport
-- WebApiCustomerSupport(recepcja) - trzyma rękę na pulsie, obserwując
zaległości w płaceniu
drop role WebApiCustomer
CREATE ROLE WebApiCustomer
/*
WebApiCustomer(Klient zarejestrowany) - moze dokonywac rezerwacji na
konferencje i warsztaty modyfikować dane rezerwacji
dodawac osoby do rezerwacji
*/
drop role WebApi
CREATE ROLE WebApi
-- WebApi(Klient niezarejestrowany) - Rejestracja konta w systemie
drop role participant
create role participant
-- participant - dostęp do procedur pokazujących na co jest zapisany

--dziedziczenie po WebApiCustomerSupport
```

```
ALTER ROLE WebApiCustomerSupport ADD member Owner;
--WebApiCustomerSupport
--dziedziczy po WebApiCustomer
ALTER ROLE WebApiCustomer ADD MEMBER WebApiCustomerSupport;
--dodatkowe widoki

--dodajemy db_datareaders i db_writers - prawa do calej bazy danych
EXEC sp_addrolemember N'db_datareader', N'Owner'  /*  odczyt tabel  i
widokow */
EXEC sp_addrolemember N'db_datawriter', N'Owner'  /*  zapis do tabel  */
EXEC sp_addrolemember N'db_owner', N'Administrator'

--WebApi
GRANT EXECUTE ON dbo.Addclients  TO WebApi
GRANT select,insert  ON  dbo.clients       TO WebApi

--WebApiCustomers
GRANT EXECUTE ON  dbo.Addconference_participants TO WebApiCustomer
GRANT EXECUTE ON  dbo.Addconferencereservation        TO WebApiCustomer
GRANT EXECUTE ON  dbo.AddPeople                            TO
WebApiCustomer
GRANT EXECUTE ON  dbo.Addworkshopparticipants         TO WebApiCustomer
GRANT EXECUTE ON  dbo.Addworkshopreservation          TO WebApiCustomer
GRANT EXECUTE ON  dbo.UpdatePeople
                TO WebApiCustomer
GRANT EXECUTE ON  dbo.Update_places_reserved_IN_workshopreservation
     TO WebApiCustomer
GRANT EXECUTE ON  dbo.Update_palces_reserved_IN_conferencereservation
     TO WebApiCustomer
GRANT EXECUTE ON  dbo.Cancel_conferencereservation
TO WebApiCustomer
GRANT EXECUTE ON  dbo.Cancel_workshopreservation
TO WebApiCustomer
GRANT EXECUTE ON  dbo.Restore_conferencereservation
TO WebApiCustomer
GRANT EXECUTE ON  dbo.Restore_workshopreservation
TO WebApiCustomer
grant execute on dbo.fill_conf_partic to webapicustomer
grant execute on dbo.fill_conf_res to webapicustomer
grant execute on dbo.fill_work_partic to webapicustomer
grant execute on dbo.fill_work_res to webapicustomer
grant execute on dbo.getFee to webapicustomer
grant execute on dbo.show_fees to webapicustomer
grant execute on dbo.workres_list to webapicustomer
Grant insert, update on dbo.conference_participants   to WebApiCustomer
Grant insert, update on dbo.conferencereservation     to WebApiCustomer
Grant insert, update on dbo.People                    to WebApiCustomer
Grant insert, update on dbo.workshopparticipants      to WebApiCustomer
Grant insert, update on dbo.workshopreservation       to WebApiCustomer
grant execute on dbo.conf_places_av to webapicustomer
grant execute on dbo.conf_places_res to webapicustomer
```

```
grant execute on getIndividualOrCompany to webapicustomer
grant execute on getSameWorkshopsID to webapicustomer
grant execute on minimum_date to webapicustomer

--Owner
GRANT EXECUTE ON dbo.Addconferencies  TO Owner
GRANT EXECUTE ON dbo.Addconfrenceday  TO Owner
GRANT EXECUTE ON dbo.Addfees               TO Owner
GRANT EXECUTE ON dbo.AddpaidRate           TO Owner
GRANT EXECUTE ON dbo.Addworkshop      TO Owner
grant select on dbo.costs to owner
grant select on dbo.vips to owner
grant select on dbo.vips_reservations to owner
grant execute on dbo.daylist to owner
grant execute on dbo.reservation_list to owner
grant execute on dbo.show_payments_for_conf to owner
grant execute on dbo.workshop_list to owner
grant execute on get_res_amount to owner
GRANT EXECUTE ON  dbo.Update_places_available_IN_confrenceday TO owner
GRANT EXECUTE ON  dbo.Update_places_available_IN_workshop TO owner

--participant
grant execute on dbo.getSameWorkshops to participant
grant execute on dbo.my_events to participant
grant execute on dbo.show_day_person to participant
grant execute on dbo.show_workshop_person to participant
grant execute on getSameWorkshopsID to participant
grant execute on minimum_date to participant
grant execute on dbo.fill_work_partic to participant

-- webapicustomersupport
grant execute on delayed to Webapicustomersupport
grant execute on dbo.generate_business_card to webapicustomersupport
grant execute on dbo.getFee to webapicustomersupport
grant execute on dbo.really_delayed to webapicustomersupport
grant execute on show_delaying_data to webapicustomersupport
grant execute on dbo.total_cost to webapicustomersupport
grant execute on dbo.check_if_student to webapicustomersupport
```

7. Opis generatora danych:

Postanowiliśmy napisać generator w języku Python. Kod z podziałem na poszczególne pliki znajduje się poniżej.

Plik main.py (korzeń projektu):

```
import datetime
from random import randint
from conferencies import fill_conferencies
from clients import fill_clients
from Every_Participant import fill_list
```

```
from Cleaning import Storage
import fileinput
def terminate(store):
    store.w.close()
    index = 0
    last =  store.conference_reservationid
    for line in fileinput.input("Workfile.sql", inplace = True):
        compare = line
        change = store.totalSums["XXX" + str(index)]
        line = line.replace("XXX" + str(index), change)
        if compare != line and index != last - 1:
            index += 1
        print line,
store = Storage()
store.w.write("use bielas_a\n\n"
        "delete from paidRate\n"
        "delete from workshopparticipants\n"
        "delete from conference_participants\n"
        "delete from workshopreservation\n"
        "delete from workshop\n"
        "delete from fees\n"
        "delete from conferencereservation\n"
        "delete from confrenceday\n"
        "delete from conferencies\n"
        "delete from People\n"
        "delete from clients\n\n")
fill_conferencies(store)
fill_clients(store)
terminate(store)
```

Plik Cleaning.py zawierający pomocniczą strukturę danych – klasę Storage:

```
import datetime
from random import randint
class Storage:
    def __init__(self):
        #self.w = codecs.open('Workfile.sql',encoding='utf-8', mode='w')
        self.w = open('Workfile.sql','w')
        self.conferencies_amount = 2*12*3
        self.conference_reservationid = 0
        self.paidRateId = 0
        self.workshopid = 0
        self.workresid = 0
        self.conferenceparticipantid = 0
        self.workshopparticipantid = 0
        self.personid = 0
        self.dayid = 0
        self.feeid = 0
        self.global_places = 0
        self.day_desc = {}
        self.workres_desc = {}
```

```
        self.cost_of_workshop = {}
        self.totalSums = {}
        self.recent_workshop_costs = 0
        self.recent_total_sum = 0
        self.recent_conf_part = []  # selected conference participants
who may register to a workshop
        self.recent_selected_students = []
        self.recent_selected_people = []
        self.clients_amount = 170
        self.participants_per_conference = 80
        self.is_canceled_dict = {True: "1", False: "0"}
        self.canceled = self.choose_canceled()
        self.people_reg_day = {}
        self.people_reg_workshop = {}
    def choose_canceled(self):
        canceled_amount = 5
        canceled = []
        for j in range(canceled_amount):
            max_resId = 200
            canceled_id = randint(0,max_resId)
            while canceled_id in canceled:
                canceled_id = randint(0,max_resId)
            canceled.append(canceled_id)
        return canceled
    def string_to_date(self,ev_date):
        return datetime.date(*(int(s) for s in ev_date.split('-')))
```

Plik conferencies.py (tu i w poniższych plikach nazwy modułów sugerują, jakie tabelki zostały nimi wypełnione).

```
import datetime
from conferenceDay import fill_conference_day
def fill_conferencies(store):
    store.daysamount = 3
    store.delta_between_conf = 15
    start = datetime.date(2000,1,1)
    conferencies_amount = store.conferencies_amount
    w = store.w
    s = open('address_set.csv','r+')
    c = open("conferencies_names","r")
    for conferencyid in range(conferencies_amount):
        address_set = s.readline().split(",")
        street,city =
address_set[0].replace("\"","").replace("\n",""),address_set[1].replace("
\"","").replace("\n","")
        topic = c.readline().replace("\n","")
        w.write("insert into conferencies values(" +
                #str(conferencyid) + "," +
                "\'" + str(start) + "\'" +
                ", \'" + str(start + datetime.timedelta(store.daysamount
- 1)) + "\'"+
```

```
                ",\'" + topic + "\'" +
                ",\'" + street + "\'" +
                ",\'" + city + "\'" +
                ")\n" )
        fill_conference_day(store, conferencyid,start,topic)
        start = start + datetime.timedelta(days =
store.delta_between_conf)
    s.close()
```

Plik conferenceDay.py:

```
from random import randint
from fees import fill_fees
from workshop import fill_workshop
import datetime
def fee_branch(store,dayid,places_available,event_date):
    main_cost_of_day = randint(195,215)
    key = dayid
    value = (places_available,event_date,main_cost_of_day)
    store.day_desc.update({key:value})
    fill_fees(store,dayid)
def workshop_branch(store,s,dayid,places_available,topic,j):
    streets = []
    workshops_amount = 3
    store.workres_desc.update({dayid:[]})
    for q in range(workshops_amount):
        streets.append(s.readline())
    fill_workshop(store,dayid,streets,places_available,topic,j)
def fill_conference_day(store,conferencyid,start,topic):
    daysamount = store.daysamount
    w = store.w
    s = open("workshop_street.csv","r")
    for j in range(daysamount):
        dayid = conferencyid*daysamount + j
        event_date = start+datetime.timedelta(j)
        places_available = randint(190,210)
        store.global_places += places_available
        w.write("insert into confrenceday values("
                #+ str(dayid) +","
                + str(places_available)+","
                + str(conferencyid) + ","
                + "\'" + str(event_date) +"\'"
                +")\n")
        store.people_reg_day.update({dayid:[]})
        fee_branch(store,dayid,places_available,event_date)
        workshop_branch(store,s,dayid,places_available,topic,j)
```

Plik fees.py:

```
from random import randint

import datetime
def fill_fees(store,dayid):
    w = store.w
    event_date,main_cost = store.day_desc[dayid][1],store.day_desc[dayid]
[2]
    first_step = 14
    second_step = 21
    third_step = 27
    costs = {event_date - datetime.timedelta(first_step) : main_cost,
             event_date - datetime.timedelta(second_step) : main_cost *
0.9,
             event_date - datetime.timedelta(third_step) : main_cost *
0.8}
    feeid = dayid * costs.__len__()
    for i in costs.keys():
        w.write("insert into fees values(" +
                #str(feeid) + "," +
                "'" + str(i) + "'," +
                str(dayid) + "," +
                str(costs[i]) +
                ")\n")
        feeid += 1
```

Plik workshop.py:

```
from random import randint
import datetime
def fill_workshop(store,dayid,streets,places_available,topic,j):
    dict = {
0 : (datetime.time(8,0,0),datetime.time(10,0,0),randint(0,25), " -
introduction no 1"),
1 : (datetime.time(8,0,0),datetime.time(10,0,0),randint(0,25), " -
introduction no 2"),
2 : (datetime.time(12,0,0),datetime.time(14,0,0),0, " - advanced
aspects") }
    for i in range(3):
        start_hour,end_hour,quote,suffix = dict[i]
        store.w.write("insert into workshop values("+
                #str(store.workshopid) + "," +
                str(quote) + "," +
                "'" + streets[i].replace("\n", "").replace("\"", "") +
```

```
"'," +
                "'" + str(end_hour) + "'," +
                "'" + str(start_hour) + "'," +
                "'" + topic + " part " + str(j) + suffix + "'," +
                str(dayid) + "," +
                str(places_available) +
                ")\n")
        store.cost_of_workshop.update({store.workshopid: quote})
        #store.people_reg_workshop.update({store.workshopid : []})
        store.workres_desc[dayid].append(store.workshopid)  # for
filling workshopreservation
        store.workshopid += 1
```

Plik clients.py:

```
from random import randint
from conferencereservation import fill_conferencereservation_person,
fill_conferencereservation_client
from Every_Participant import fill_list
def gen_passwd():
    minlen = 15
    maxlen = 30
    res = ""
    beginint = 97
    endint = 122
    scope = randint(minlen,maxlen)
    for i in range(scope):
      res += unichr(randint(beginint,endint)).encode('utf-8')
    return res
def print_people(lp):
    r = 0
    for i in lp:
        print str(r) + ". ",
        i.toString()
        r += 1
    print ""

def fill_clients(store):
    companies = open("companies.csv","r")
    names = open("Names_Surn.csv","r")
    marked_participants = []  # randomly selected ids of people
mentioned in the reservation
    students_amount = 200
    people_amount = 400
    students,people = fill_list(students_amount,people_amount)
    w = store.w
    countries = ['Argentina', "Brazil","Germany","UK"]
    for clientid in range (store.clients_amount):
        passwd = "'" + gen_passwd() + "'"
        country = "'" + countries[randint(0,len(countries)-1)] + "'"
```

```
        boundry = 20 # tells, when individuals stop generating
        if clientid <= boundry:
            line = names.readline().split(",")
            surname, name = line[0],line[1].replace("\n","")
            email = "'" + surname + "." + name + "@gmail.com'"
            w.write("insert into clients values(" +
                    #str(clientid) + ","
                    "null"+","
                    + passwd+","
                    + "\'" + surname + "\',"
                    + "\'" + name + "\',"
                    + email + ","
                    + "1" + ","
                    + country
                    + ")\n")

fill_conferencereservation_person(store,clientid,students,people,marked_p
articipants)
        else:
            line = companies.readline()
            scope = 5
            companyname = "'" +
line.replace("\"","").replace("'","").replace("\n","") + "'"
            email = "'" + companyname[1:scope].replace("'","") +
"@gmail.com'"
            w.write("insert into clients values(" +
                    #str(clientid)+","
                    companyname+","
                    +passwd+","
                    +"null"+","
                    +"null"+","
                    +email+","
                    +"0"+","
                    +country
                    +")\n")

fill_conferencereservation_client(store,clientid,students,people,marked_p
articipants)
```

Plik Every_Participant.py, tworzący listę obiektów People, z których dobieranie są późniejsi uczestnicy konferencji i warsztatów.

```
from Conference import Participant
def fill_list(students_amount,people_amount):
    students = []
    people = []
    part_names = open("names_participants.csv","r")
    part_address = open("city_country_street_part.csv","r")
    id_number = 276000
    phones = open ("phones","r")
    for i in range (students_amount + people_amount):
```

```
        name,surname = part_names.readline().split(",")
        city,country,street = part_address.readline().split(",")
        phone = phones.readline()
        name = name.replace("\n","").replace("\"","")
        surname = surname.replace("\n","").replace("\"","")
        city = city.replace("\n","").replace("\"","")
        country = country.replace("\n","").replace("\"","")
        street = street.replace("\n","").replace("\"","")
        phone = phone.replace("\n","").replace("\"","")
        part = Participant(name,surname,phone,country,city,street,i)
        if(i < students_amount):
            part.id_number_one = id_number
            part.id_number_two = id_number + 1
            id_number += 2
            students.append(part)
        else:
            people.append(part)
    return students,people
```

Plik conferenceReservation.py:

```
import fileinput
import datetime
from random import randint
from paidRate import fill_paidRate
from conferenceparticipants import fill_conference_participants
from workshopreservation import fill_workshopreservation
def randDay(N,marked_days):  # chooses unique dayids
    dayid = randint(3,N)
    while dayid in marked_days:
        dayid = randint(0,N)
    return dayid,marked_days + [dayid]
def choose_participant(store,possible_people,how_many,dayid):
    result = []
    marked = []
    day_participants = store.people_reg_day[dayid]
    for i in range(how_many):
        option = randint(0,len(possible_people)-1)
        while option in marked or option in day_participants:
            option = randint(0,len(possible_people)-1)
        marked.append(option)
        store.people_reg_day[dayid].append(option)
        result.append(possible_people[option])
    return result

def branches
(store,dayid,reservation_date,places_reserved,studentsAmount,selected_stu
dents,selected_people,marked_participants,places_available):
```

```
fill_conference_participants(store,selected_students,selected_people,mark
ed_participants)
    fill_workshopreservation(store,dayid,places_available,
store.conference_reservationid in store.canceled)

fill_paidRate(store,dayid,reservation_date,places_reserved,studentsAmount
)
    store.totalSums.update({"XXX" + str(store.conference_reservationid) :
'%.2f' % store.recent_total_sum})

def fill_conferencereservation_person
(store,clientid,students,people,marked_participants):
    conf_day_amount = int(store.conferencies_amount) * 3 - 1
    marked_days = []
    max_places = 30 # maximum number of places that an individual person
can                 reserve
    reservations_amount = randint(2,6)  # amount of reservations
submitted by                                    this particular
client/person
    w = store.w
    for i in range(reservations_amount):
        dayid,marked_days = randDay(conf_day_amount,marked_days)
        places_available, event_date,main_cost = store.day_desc[dayid]
        places_reserved = min(randint(1, max_places), places_available)
        store.global_places -= places_reserved
        if places_reserved > 0:
            reservation_date = event_date -
datetime.timedelta(randint(15, 27))
            studentsAmount = randint(0,places_reserved)
            selected_students =
choose_participant(store,students,studentsAmount,dayid)
            selected_people =
choose_participant(store,people,places_reserved - studentsAmount,dayid)
            w.write("insert into conferencereservation values(" +
                            #str(store.conference_reservationid) + "," +
                            str(places_reserved) + "," +

store.is_canceled_dict[store.conference_reservationid in store.canceled]
+ "," +
                            "XXX" + str(store.conference_reservationid) +
"," +
                            #str(studentsAmount) + ","+
                            "'" + str(reservation_date) + "'," +
                            str(clientid) + "," +
                            str(dayid) +
                            ")\n")

branches(store,dayid,reservation_date,places_reserved,studentsAmount,sele
cted_students,selected_people,marked_participants,places_available)
            store.day_desc.update({dayid:(places_available-
places_reserved,event_date,main_cost)})
```

```
            store.conference_reservationid += 1

def fill_conferencereservation_client
(store,clientid,students,people,marked_participants):
    conf_day_amount = int(store.conferencies_amount) * 3 - 1
    marked_days = []
    places_per_client = store.global_places / store.clients_amount
    participants_per_conference = store.participants_per_conference
    res_amount = places_per_client / participants_per_conference
    w = store.w
    for i in range(res_amount):
        dayid,marked_days = randDay(conf_day_amount,marked_days)
        places_available, event_date,main_cost = store.day_desc[dayid]
        places_reserved = min(participants_per_conference + randint(3,9),
places_available)
        places_per_client -= places_reserved
        if places_reserved > 0:
            reservation_date = event_date -
datetime.timedelta(randint(15, 27))
            studentsAmount = randint(0,places_reserved)
            selected_students =
choose_participant(store,students,studentsAmount,dayid)
            selected_people =
choose_participant(store,people,places_reserved - studentsAmount,dayid)
            w.write("insert into conferencereservation values(" +
                            #str(store.conference_reservationid) + "," +
                            str(places_reserved) + "," +

store.is_canceled_dict[store.conference_reservationid in store.canceled]
+ "," +
                             "XXX" + str(store.conference_reservationid)
+ "," +
                            #str(studentsAmount) + ","+ #students
                            "'" + str(reservation_date) + "'," +
                            str(clientid) + "," +
                            str(dayid) +
                            ")\n")

branches(store,dayid,reservation_date,places_reserved,studentsAmount,sele
cted_students,selected_people,marked_participants,places_available)
            store.day_desc.update({dayid:(places_available-
places_reserved,event_date,main_cost)})
            store.conference_reservationid += 1

use_remaining_coins(store,places_per_client,marked_days,students,people,c
lientid,marked_participants)

def use_remaining_coins
(store,places_per_client,marked_days,students,people,clientid,marked_part
icipants):
    participants_per_conference = store.participants_per_conference
```

```
    w = store.w
    for dayid in store.day_desc.keys():
        if places_per_client >= participants_per_conference:  # if has
                                                              points/coins
to use
            places_available,event_date,main_cost = store.day_desc[dayid]
            if places_per_client <= places_available and dayid not in
marked_days:
                places_reserved = participants_per_conference
                places_per_client -= places_reserved
                reservation_date = event_date -
datetime.timedelta(randint(15, 27))
                studentsAmount = randint(0,places_reserved)
                selected_students =
choose_participant(store,students,studentsAmount,dayid)
                selected_people =
choose_participant(store,people,places_reserved - studentsAmount,dayid)
                w.write("insert into conferencereservation values(" +
                        #str(store.conference_reservationid) + "," +
                        str(places_reserved) + "," +

store.is_canceled_dict[store.conference_reservationid in store.canceled]
+ "," +  # if canceled, cancelling bit is set to 1
                         "XXX" + str(store.conference_reservationid) +
"," +
                        #str(studentsAmount) + ","+ #students
                        "'" + str(reservation_date) + "'," +
                        str(clientid) + "," +
                        str(dayid) +
                        ")\n"
                        )

branches(store,dayid,reservation_date,places_reserved,studentsAmount,sele
cted_students,selected_people,marked_participants,places_available)
                store.day_desc.update({dayid:(places_available -
places_reserved,event_date,main_cost)})
                store.conference_reservationid += 1
        else: break
```

plik conferenceparticipants.py:

```
from Person import fill_Person
def fill_conference_participants
(store,selected_students,selected_people,marked_participants):
    w = store.w
    reservationid = store.conference_reservationid
    store.recent_conf_part = []
    for student in selected_students:
        id_number = student.id_number_two
        personid,participantid =
store.personid,store.conferenceparticipantid
```

```
        if student.id not in marked_participants:
            student.personid = personid
            fill_Person(store, student)
            store.personid += 1
            marked_participants.append(student.id)
            id_number = student.id_number_two
        student.participantid = participantid
        w.write("insert into conference_participants values("+
                str (student.personid) + "," +
                "'" + str (id_number) + "'," +
                str (reservationid)  +
                #str (participantid) +
                ")\n")
        store.recent_conf_part.append(participantid)
        store.conferenceparticipantid +=1
    for person in selected_people:
        personid,participantid =
store.personid,store.conferenceparticipantid
        if person.id not in marked_participants:
            person.personid = personid
            fill_Person(store,person)
            store.personid += 1
            marked_participants.append(person.id)
        person.participantid = participantid
        w.write("insert into conference_participants values("+
                str (person.personid) + "," +
                "null," +
                str (reservationid) +
                #str (participantid) +
                ")\n")
        store.recent_conf_part.append(participantid)
        store.conferenceparticipantid += 1
    store.recent_selected_students = selected_students
    store.recent_selected_people = selected_people
```

Plik Conference.py - **klasa obiektów które zostaną użyte w wypełnianiu tabeli People:**

```
class Participant:
    def __init__(self,name,surname,phone,country,city,street,id,personid
= None,id_number_one = None, id_number_two = None):
        self.id = id
        self.name = name
        self.surname = surname
        self.phone = phone
        self.country = country
        self.city = city.replace("'","")
        self.street = street.replace("'","")
        self.personid = personid
        self.id_number_one = id_number_one
        self.id_number_two = id_number_two
```

```
    def toString(self):
        print self.name + " " + self.surname + " " + self.phone + " " +
self.country + " " + self.city + " " + self.street + " " + str(self.id)
```

Plik Person.py:

```
def fill_Person(store,person):
    store.w.write("insert into People values("+
            #str(person.personid) + "," +
            "'" + str(person.surname) + "'," +
            "'" + str(person.name) + "'," +
            "'" + str(person.phone) + "'," +
            "'" + str(person.country) + "'," +
            "'" + str(person.city) + "'," +
            "'" + str(person.street) + "'" +
            ")\n")
```

Plik workshopreservation.py:

```
import math
from random import randint
from workshopparticipants import fill_workshopparticipants
def select_random_participants(collection,amount):
    marked_indecies = []
    res = []
    scope = len(collection) - 1
    for i in range(amount):
        index = randint(0,scope)
        while index in marked_indecies:
            index = randint(0,scope)
        res.append(collection[index])
        marked_indecies.append(index)
    return res

def select_third_group(store):
    people = store.recent_selected_people
    students = store.recent_selected_students
    res = []
    people_amount = len(people)/2
    students_amount = len(students)/2
    marked_people = []
    marked_students = []
    for i in range(students_amount):
        stud_index = randint(0,len(students) - 1)
        while stud_index in marked_students:
            stud_index = randint(0,len(students) - 1)
        res.append(students[stud_index])
        marked_students.append(stud_index)
    for i in range(people_amount):
        person_index = randint(0,len(people) - 1)
```

```
        while person_index in marked_people:
            person_index = randint(0,len(people) - 1)
        res.append(people[person_index])
        marked_people.append(person_index)
    return res,students_amount,people_amount

def fill_workshopreservation(store,dayid,places_available,canceled):
    w = store.w
    first_half_of_students =
int(math.floor(len(store.recent_selected_students)/2.0))
    second_half_of_students =
int(math.ceil(len(store.recent_selected_students)/2.0))
    first_half_of_people =
int(math.floor(len(store.recent_selected_people)/2.0))
    second_half_of_people =
int(math.ceil(len(store.recent_selected_people)/2.0))
    first_group = store.recent_selected_students[:first_half_of_students]
+ store.recent_selected_people[:first_half_of_people]
    second_group =
store.recent_selected_students[first_half_of_students:] +
store.recent_selected_people[first_half_of_people:]
    third_group,third_half_of_students,third_half_of_people =
select_third_group(store)
    workshopids = store.workres_desc[dayid]
    groups_desc = [

[first_group,first_half_of_students,first_half_of_people],

[second_group,second_half_of_students,second_half_of_people],

[third_group,third_half_of_students,third_half_of_people ]
            ]
    store.recent_workshop_costs = 0
    for i in range(len(workshopids)):
        group,students_amount,people_amount = groups_desc[i]
        w.write("insert into workshopreservation values(" +
                #str(store.workresid) + "," +
                str(len(group)) + "," +
                str(workshopids[i]) + "," +
                str(store.conference_reservationid) + "," +
                store.is_canceled_dict[canceled] +
                ")\n")
        store.recent_workshop_costs += (people_amount + 0.5 *
students_amount)* store.cost_of_workshop[workshopids[i]]
        fill_workshopparticipants(store,store.workresid,group)
        store.workresid += 1
```

Plik workshopparticipants.py:

```
def fill_workshopparticipants(store,workresid,participants):

    w = store.w
    for participant in participants:
        w.write("insert into workshopparticipants values(" +
                str(workresid) + "," +
                str(participant.participantid)  +
                #str(store.workshopparticipantid) +
                ")\n")
        store.workshopparticipantid += 1
```

Plik paidRate.py:

```
from random import randint,random
import fileinput
import datetime
def discounts(reservation_date,eventdate,maincost):
    if reservation_date == eventdate - datetime.timedelta(14):
        return maincost
    elif reservation_date >= eventdate - datetime.timedelta(21) and
reservation_date <= eventdate - datetime.timedelta(15):
        return maincost * 0.9
    elif reservation_date >= eventdate - datetime.timedelta(27) and
reservation_date <= eventdate - datetime.timedelta(22):
        return maincost * 0.8
def
fill_paidRate(store,dayid,reservation_date,placesReserved,studentsAmount)
:
    places_available,event_date,cost = store.day_desc[dayid]
    price = discounts(reservation_date,event_date,cost)
    totalSum = price * (placesReserved-studentsAmount + 0.5 *
studentsAmount) + store.recent_workshop_costs
    store.recent_total_sum  = totalSum
    is_canceled = False
    reservationId = store.conference_reservationid
    if reservationId in store.canceled:
        totalSum *=  0.9
        is_canceled = True
    rates_Amount = float(randint(1,3))
    one_rate = totalSum/rates_Amount
    for i in range(int(rates_Amount)):
        date_of_paying = reservation_date + datetime.timedelta(days =
randint(1,6))
        if is_canceled:
             one_rate = one_rate - randint(1,30) - random()
        store.w.write("insert into paidRate values("+
                str(reservationId) + "," +
                "'" + str(date_of_paying) + "'," +
                '%.2f' % one_rate  +
```

```
                #str(store.paidRateId) +
                ")\n")
        store.paidRateId += 1
```

plik Names_Surn.csv do generowania klientów:

```
Dean Gladys
Massey,Teofila
Houck,Rebecca
McCarthy,Denver
Hayes,Carlos
Langlois,Tony
Hunt,Marion
Fernandez,Jerlene
Diep,Robert
Jones,Gwen
Gallagher,Virginia
Gerrish,Albert
Andrews,Robert
Jones,Stephen
Hobbs,Maria
Jones,Diana
Simpson,Caroline
Morris,Robert
Cochran,Sadie
Ferguson,Bernadette
Roberson,Nellie
```

Plik companies.csv do nadania nazw klientom-firmom:

```
Laneco
"Allied City Stores"
"Locost Accessories"
"Matrix Architectural Service "
"Simply Appraisals"
"Hughes Markets"
"Service Merchandise"
"Hollywood Video"
"Alladin's Lamp"
"Eden Lawn Service"
"O.K. Fairbanks"
"Office Warehouse"
"H.C. Bohack"
"Leo's Stereo"
Oshman's
```

"Superior Appraisals"
BASCO
"Laughner's Cafeteria"
"The Royal Canadian Pancake Houses"
Rink's
"Magik Grey"
"Rivera Property Maintenance"
"Custom Lawn Care"
"Scotty's Builders Supply"
"National Tea"
"ABCO Foods"
"Northern Star"
Maxaprofit
"A+ Investments"
Gallenkamp
Ejecta
"Superior Interactive"
Capitalcorp
"Express Merchant Service"
"Laura Ashley"
"Henry's Hamburgers"
"Spaceage Stereo"
"House Of Denmark"
"Bugle Boy"
Fireball
"Quality Event Planner"
Oshman's
"Jacob Reed and Sons"
"Builders Square"
"Hughes Markets"
"Forest City"
"The Spotted Cougar"
"Pro-Care Garden Maintenance"
"Golden's Distributors"
Hechinger
Rink's
"Sam Goody"
"Builders Emporium"
"Elm Farm"
"Pro-Care Garden Maintenance"
"De-Jaiz Mens Clothing"
Muirhead's
"Lindsley's Lumber"
"Maloley's Finer Foods"
"Flagg Bros. Shoes"
"New World"
"Gene Walter's Marketplace"
"Endicott Shoes"
JumboSports
"Maloley's Finer Foods"
"The Record Shops at TSS"

"Planet Profit"
"Future Plan"
"Sunny's Surplus"
"Electronics Source"
"Kenny Rogers Roasters"
"Good Guys"
"Rainbow Life"
"Dick Fischers"
"Star Bright Investment Group"
Rink's
"The Wiz"
"Dee's Drive-In"
"Linens 'n Things"
"Gas Zone"
Prospa-Pal
Wetson's
"Universo Realtors"
"Penn Fruit"
Awthentikz
"Price's Electronics"
"Sunflower Market"
"Solid Future"
"Poore Simon's"
"Independent Investors"
"Bountiful Harvest Health Food Store"
"Today's Man"
"Food Mart"
FamilyMart
Luria's
"Jack Lang"
"Brooks Fashions"
"Buehler Foods"
"Chief Auto Parts"
"The Warner Brothers Store"
"Pay 'N Pak"
Weathervane
Adaptas
"Sam Goody"
Luria's
Lum's
"Blockbuster Music"
"Singer Lumber"
"Liberty Wealth Planner"
McDade's
"Griff's Hamburgers"
"Pay 'N Pak"
Lazysize
Veramons
"Sure Save"
"Manning's Cafeterias"
"Asian Fusion"

"Chief Auto Parts"
"Alpha Beta"
"Omni Source"
"CSK Auto"
"Our Own Hardware"
Weathervane
"Frank and Seder"
"Desert Garden Help"
"Super Saver Foods"
"Suncoast Video"
"The White Rabbit"
"Planet Pizza"
"Earthworks Garden Kare"
"Sounds of Soul Records & Tapes"
"Two Pesos"
"Asian Fusion"
GEX
"Thrift Auto Parts"
"Better Business Ideas and Services"
"Reliable Investments"
"Enrich Garden Services"
"Sky High Financial Advice"
"Techo Solutions"
Ulbrich's
"Rhodes Furniture"
"Quality Merchant Services"
"Strong Life"
"DGS VolMAX"
"Hughes & Hatcher"
"Franklin Music"
"Crandall's Fine Furniture"
Briazz

Plik conferencies_names.txt użyty do wygenerowania topic'ów dla konferencji:

Lambda days
Skyrim fans reunion
The Force Awakens
kinect
control
websites today
house hold appliance
TUIO
openCV
openGL
MMORPG
virtual reality
augmented reality
3d glasses
android

mobile phones today
grenade
laser show
forum
device control
simulator
robot
holography
hologram
internet
python
time lapse
slavic music
virtual
web
vision
motion
mobile
tracking
HMD
touch screens
gallery
audio
sound
alarm
encryption
biometry
fingerprints
quake
digital
analog
cloud
plane
depth
robotics
library
teleop
porn
fight
shadow
effects
particle
genetic
nanotechnology
recognition
draw
physics
scanning
manipulation
gravity
multitouch

space
linux
prototype
algorithms
democracy
detection

plik adress_set.csv wykorzystany do wypełniania atrybutów street i country w tabeli conferencies:

"Gewerbezentrum 64",STARNBERG

"Zeppelinstr 38",WEIXELBERG
"Wiehtestrasse 40",SINZING
"Enzersdorfer Strasse 34",GIESSHÜBL
"Gewerbezentrum 96",STEG
"Kaisergasse 35",LAAS
"Baumgarten 44",NIEDERHOFSTETTEN
"Alpenstrasse 43",STEINBERG
"Pazmaniteng 90",ABTSDORF
"Zistelweg 25",UNTERKLING
"Ernstbrunner Strasse 56",REICHHARTS
"Kuefsteinstrasse 16","SANKT LORENZEN OB MURAU"
"An Der Bundesstrasse 95",HINTERHOLZ
"Auenweg 22",PÖTTSCHING
"Huttenstrasse 46",TSCHAU
"Gumpendorfer Strasse 62",KEFERMARKT
"Salzburgerstrasse 62",GROSS-ENZERSDORF
"Neutorstrasse 2",TÖLTSCHACH
"Zeppelinstrasse 86","ST. PETER"
"Kriemhildstrasse 14",KROTTENTHAL
"Mattersburger Strasse 29",PEHIGEN
"Wiedner Hauptstrasse 64",HÖTZELSDORF
"Obdacher Bundesstrasse 86",OBERDÖRFL
"Feldham 10",DIEMLACH
"Thayapark 63",ROSSA
"Untere Donaulände 97",BÖHLERWERK
"Bräuhof 12",ASBERG
"Spiegelsberg 37",ALTMANNS
"Ziegelgasse 81",GRUB
"Hadikgasse 75",KÖTZING
"Schachterlweg 65",POLZBERG
"Bahnhofstrasse 75",DRASSMARKT
"Kärntner Strasse 5","HASLAU AN DER DONAU"
"Holzstrasse 1",SALLEGG
"Marktplatz 73",RENNERSDORF
"Traunuferstrasse 89",BRUDERNDORF
"Gleichenberger Strasse 4",GSCHWENDTHÄUSER
"Silvrettastrasse 85",SCHWARZENBACH
"Thayapark 65","ROSENAU AM HENGSTPASS"
"Sternhofweg 23",GOLLING

"Huttenstrasse 72",TRENK
"Ernstbrunner Strasse 38",REIGERSBERG
"Stollenstrasse 24",OBERTÖLLERN
"Grössgstötten 72",FLATZ
"Haid 74",MAYRHÖFEN
"Weblinger Gürtel 99",HELBETSCHLAG
"Traungasse 3",FAHRENDORF
"Ortsstrasse 32",WIESMÜHL
"Völkermarkter Strasse 83",PICHLA
"Schönaugasse 31",URTLGRABEN
"Wurmbrandgasse 73",SIEBENBERG
"Neuhofer Strasse 25",WINKL
"Mattersburger Strasse 19",PERLEITEN
"Hauptplatz 67",OBERKURZHEIM
"Schallmooser Hauptstrasse 53","SCHÖNAU AN DER TRIESTING"
"Aspernstrasse 11",SCHEUTZ
"Hausergasse 55",WÜRTING
"Salzburgerstrasse 66",GROLZHAM
"An Der Bundesstrasse 73",HINTERGLEMM
"Feldham 87",DIEDERSDORF
"Gartenweg 32",ALBERTING
"Waldstrasse 34",FEITZING
"Zistelweg 66",UNTERLEHEN
"Silvrettastrasse 18","SCHWARZENBERG AM BÖHMERWALD"
"Baumgarten 78",NIEDERPRISCHING
"Kärntner Strasse 30",HASELSDORFBERG
"Galgenau 29",NEIDLING
"Sonnberg 16",ANZBERG
"Dreimühlenweg 27",SCHLITTERS
"Steinfelden 83",UNTERSEE
"Zeppelinstrasse 84","ST. PETER AM WALLERSBERG"
"Langenloiser Strasse 72",MÖDRING

Z uwagi na zbyt dużą wielkość plików workshop_street.csv uznaliśmy, że nie załączymy go do sprawozdania (ulica, przy której ma miejsce warsztat)

To samo dotyczy plików phones.txt (numery telefonów uczestników), names_participants.csv, city_country_street_part.csv (zestaw adresowy dla wierszy w tabeli People).