

Data Warehouse & Business Intelligence

Proiect

1. Descrierea modelului ales și a obiectivelor aplicației

Modelul ales în dezvoltarea aplicației de Data Warehouse reprezintă o bază de date ce conține zborurile companiilor aeriene împreună cu rezervările făcute de clienți, pentru a evidenția diverse rezultate cu privire la zboruri efectuate, destinațiile cele mai frecventate sau numărul de zboruri lunare pentru anumiți operatori de zbor.

În cadrul proiectului a fost folosit drept punct de plecare setul de date [2015 Flight Delays and Cancellations](#) de pe platforma Kaggle. Acesta este populat cu date reale, ce prezintă detalii despre zborurile efectuate de companiile aeriene din Statele unite ale Americii în primele 3 luni ale anului 2015. Coloanele din tabele, precum destinația de plecare și cea de sosire, timpul programat al plecării, respectiv al sosirii, durata zborului cât și distanța efectuată, sunt colectate de către *U.S. Department of Transportation's (DOT) Bureau of Transportation Statistics*, ulterior fiind publicate în raportul lunar *Air Travel Consumer Report*.

Setul de date menționat conține următoarele tabele, sub formă de fișiere csv:

- *Airlines.csv* - conține un cod unic al companiei aeriene, împreună cu numele acesteia
- *Airports.csv* - conține codul unic al aeroportului, numele acestuia, orașul, statul, țara (fiind USA în toate cazurile) și coordonatele acestuia
- *Flights.csv* - conține data, ora de plecare, respectiv de sosire, destinația de plecare, respectiv de sosire, reprezentate prin codul aeroportului, codul liniei aeriene, numărul zborului, numărul aeronavei, durata și distanța parcursă, împreună cu alte date referitoare la întârzieri și anulări.

Conținutul tabelelor menționate este folosit în cadrul aplicației în modul următor:

- Fișierul *airlines.csv* devine sursă de date pentru tabelul OPERATOR_ZBOR, care va conține codul operatorului, împreună cu denumirea acestuia
- Fișierul *airports.csv* definește destinațiile posibile pentru zboruri în tabela DESTINATIE, din care vom prelua codul aeroportului, orașul și statul
- Fișierul *flights.csv* devine sursa de date pentru tabelul ZBOR, iar coloanele preluate sunt: codul operatorului, codul aeronavei, durata, distanța, coloana anulat, cu valoarea 0 dacă zborul a avut loc, respectiv 1 dacă a fost anulat, data de plecare împreună cu

ora de plecare, data de sosire împreună cu ora de sosire și locația de plecare și de sosire.

Deoarece fișierul *flights.csv* nu conținea toată informația pe care doream să o folosim în baza de date, am creat un script în Python care modifică fișierul csv în următorul mod:

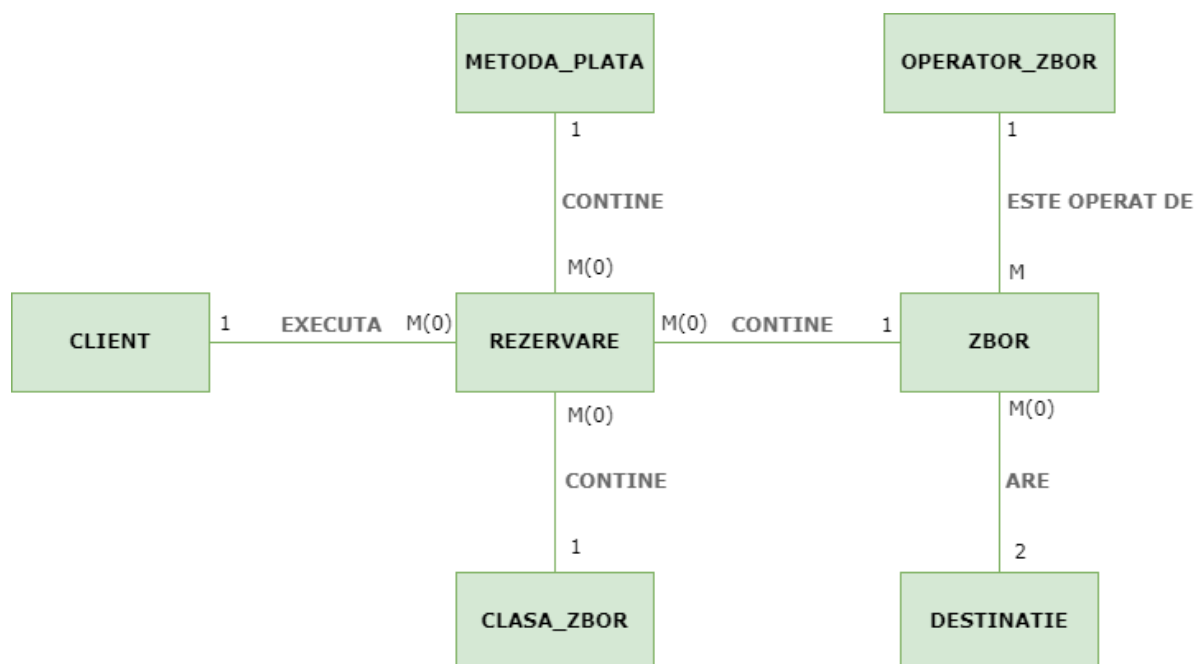
- Adaugă coloana *zbor_id*, creată incremental, pentru a o folosi în continuare cu ușurință
- Generează coloana *total_locuri*, cu valori între 50 și 100
- Modifică coloanele *data_plecare* și *data_sosire* de tip timestamp, pentru a conține și ora

Pentru a genera clienții care vor face ulterior rezervări, a fost folosit setul de date [people](#) de 10000 înregistrări din care am preluat id-ul, numele, prenumele, email-ul și numărul de telefon.

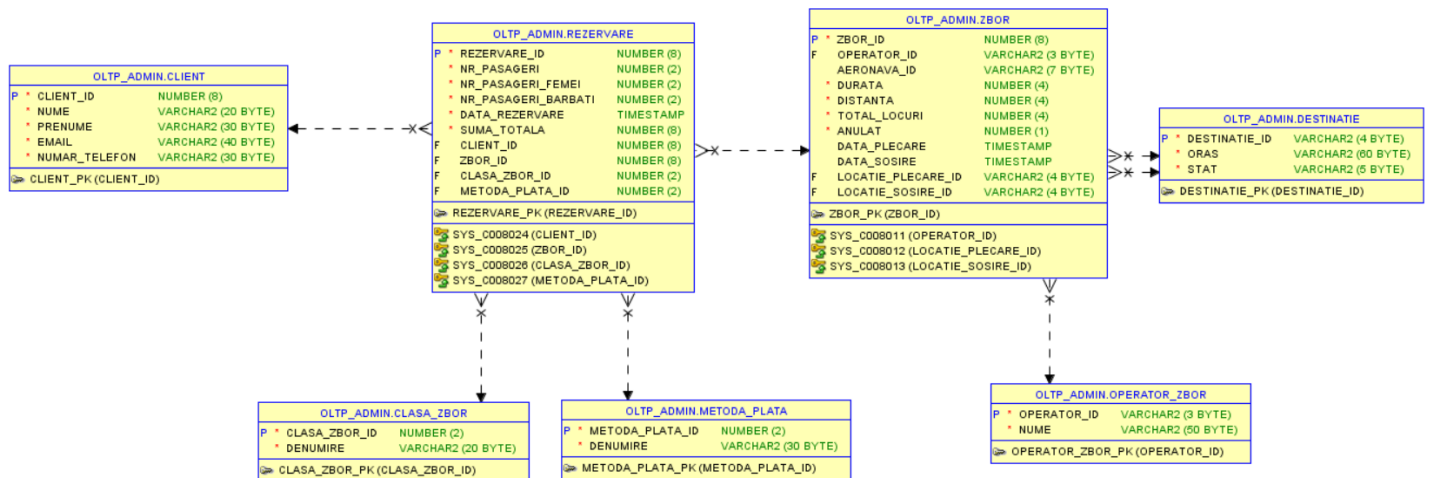
Rezervările clienților au fost generate în totalitate folosind un script de python și conțin: id-ul, numărul total de pasageri conținut de rezervare, numărul de pasageri care sunt femei, respectiv bărbați, data rezervării, suma totală platită de client, id-ul clientului, id-ul zborului, clasa la care au fost rezervate locurile și metoda de plată. Atât clasele de zbor, cât și metodele de plată posibile sunt stocate în tabele separate.

2. Diagramele bazei de date OLTP

- Diagrama entitate - relație a bazei de date OLTP



- Diagrama conceptuală a bazei de date OLTP



3. Crearea bazei de date OLTP și a utilizatorilor

Crearea tabelilor din baza de date OLTP:

CLIENT:

```
CREATE TABLE CLIENT
(client_id NUMBER(8) PRIMARY KEY,
nume VARCHAR2(20) NOT NULL,
prenume VARCHAR2(30) NOT NULL,
email VARCHAR2(40) NOT NULL,
numar_telefon VARCHAR2(30) NOT NULL);

SELECT * FROM CLIENT;
```

DESTINATIE:

```
CREATE TABLE DESTINATIE
(destinatie_id VARCHAR2(4) PRIMARY KEY,
oras VARCHAR2(60) NOT NULL,
stat VARCHAR2(5) NOT NULL);

SELECT * FROM DESTINATIE;
```

ZBOR:

```
CREATE TABLE ZBOR
(zbor_id NUMBER(8) PRIMARY KEY,
operator_id VARCHAR2(3) REFERENCES OPERATOR_ZBOR(operator_id)
ON DELETE CASCADE,
aeronava_id VARCHAR2(7),
durata NUMBER(4) NOT NULL,
distanta NUMBER(4) NOT NULL,
total_locuri NUMBER(4) NOT NULL,
anulat NUMBER(1) CHECK (anulat IN (0, 1)),
data_plecure TIMESTAMP NOT NULL,
data_sosire TIMESTAMP NOT NULL,
locatie_plecure_id VARCHAR2(4) REFERENCES
DESTINATIE(destinatie_id) ON DELETE CASCADE,
locatie_sosire_id VARCHAR2(4) REFERENCES
DESTINATIE(destinatie_id) ON DELETE CASCADE);

SELECT * FROM ZBOR;
```

REZERVARE:

```
CREATE TABLE REZERVARE
(rezervare_id NUMBER(8) PRIMARY KEY,
nr_pasageri NUMBER(2) NOT NULL CHECK(nr_pasageri > 0),
nr_pasageri_femei NUMBER(2) NOT NULL CHECK(nr_pasageri_femei
>= 0),
nr_pasageri_barbati NUMBER(2) NOT NULL
CHECK(nr_pasageri_barbati >= 0),
data_rezervare TIMESTAMP NOT NULL,
suma_totala NUMBER(8) NOT NULL CHECK(suma_totala >= 0),
client_id NUMBER(8) REFERENCES CLIENT(client_id) ON DELETE
CASCADE,
zbor_id NUMBER(8) REFERENCES ZBOR(zbor_id) ON DELETE CASCADE,
clasa_zbor_id NUMBER(2) REFERENCES CLASA_ZBOR(clasa_zbor_id)
ON DELETE CASCADE,
metoda_plata_id NUMBER(2) REFERENCES
METODA_PLATA(metoda_plata_id) ON DELETE CASCADE);

SELECT * FROM REZERVARE;
```

OPERATOR ZBOR:

```
CREATE TABLE OPERATOR_ZBOR
(operator_id VARCHAR2(3) PRIMARY KEY,
nume VARCHAR2(50) NOT NULL);

SELECT * FROM OPERATOR_ZBOR;
```

METODA_PLATA:

```
CREATE TABLE METODA_PLATA
(metoda_plata_id NUMBER(2) PRIMARY KEY,
denumire VARCHAR2(30) NOT NULL);
```

CLASA_ZBOR:

```
CREATE TABLE CLASA_ZBOR
(clasa_zbor_id NUMBER(2) PRIMARY KEY,
denumire VARCHAR2(20) NOT NULL);
```

Crearea utilizatorilor OLTP și oferirea drepturilor de acces pentru următoarele cerințe:

```
SHOW con_name;
ALTER SESSION SET CONTAINER=orclpdb;

CREATE USER oltp_admin IDENTIFIED BY pass_oltp;
GRANT CREATE SESSION TO oltp_admin;
GRANT CREATE TABLE TO oltp_admin;
GRANT CREATE SEQUENCE TO oltp_admin;
ALTER USER oltp_admin QUOTA UNLIMITED ON USERS;
```

4. Generarea datelor și inserarea acestora în tabele

Pentru generarea datelor din tabelele REZERVARE, respectiv ZBOR, s-au utilizat următoarele script-uri de *python*.

```
import csv
import numpy as np
import datetime
```

```

rows = []
with open('flights-2.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            print(f'Column names are {", ".join(row)}')
            line_count += 1
        else:
            line_count += 1

            year, month, day = row[:3]
            month = '0' + month if len(month) == 1 else month
            day = '0' + day if len(day) == 1 else day

            date = f"{year}-{month}-{day}"

            rows.append([date] + row[3:])

ids = np.arange(len(rows)) + 1

for index in range(len(rows)):
    rows[index].append(str(ids[index]))
    rows[index].append(str(np.random.choice([50,70,80,90,100])))
    datetime_object = datetime.datetime.strptime(rows[index][0],
'%Y-%m-%d') + datetime.timedelta(minutes=int(rows[index][5]))

    if len(rows[index][6]) == 0:
        rows[index][6] = '0'

    arrival = datetime_object + datetime.timedelta(minutes =
int(rows[index][6])) if len(rows[index][6]) != 0 else
datetime_object

    rows[index][5] = datetime_object.strftime("%Y-%m-%d %H:%M:%S")
    rows[index][8] = arrival.strftime("%Y-%m-%d %H:%M:%S")

    rows[index] = rows[index][1:]

```

```
print(2)
```

```
with open('flights_updated.csv','w') as fout:
```

```
fout.write('operator_id,aeronava_id,locatie_plecare_id,locatie_sos  
ire_id,SCHEDULED_DEPARTURE,durata,distanta,SCHEDULED_ARRIVAL,anula  
t,id_zbor,total_locuri\n')
```

```
    for row in rows:  
        fout.write(','.join(row))  
        fout.write('\n')
```

```
import csv  
from pydoc import cli  
import numpy as np  
import datetime  
from faker import Faker  
fake = Faker()
```

```
rows = []  
N = 1040000  
for index in range(N):
```

```
    rezervare_id = index + 1  
    nr_pasageri = np.random.randint(1, 11)  
    nr_pasager_femei = np.random.randint(0, nr_pasageri + 1)  
    nr_pasager_barbati = nr_pasageri - nr_pasager_femei
```

```
    start_date = datetime.date(year=2015, month=1, day=1)  
    end_date = datetime.date(year=2015, month=12, day=31)
```

```
    data_rezervare = fake.date_between(start_date=start_date,  
end_date=end_date).strftime("%Y-%m-%d %H:%M:%S")  
    suma_totala = np.random.randint(100, 10001)
```

```
    client_id = np.random.randint(1, 10001)
```

```

zbor_id = np.random.randint(1, 1048575 + 1)

clasa_id = np.random.randint(1, 4)

plata = np.random.randint(1, 4)

rows.append([str(rezervare_id), str(nr_pasageri),
str(nr_pasager_femei), str(nr_pasager_barbati), data_rezervare,
str(suma_totala), str(client_id), str(zbor_id), str(clasa_id),
str(plata)])

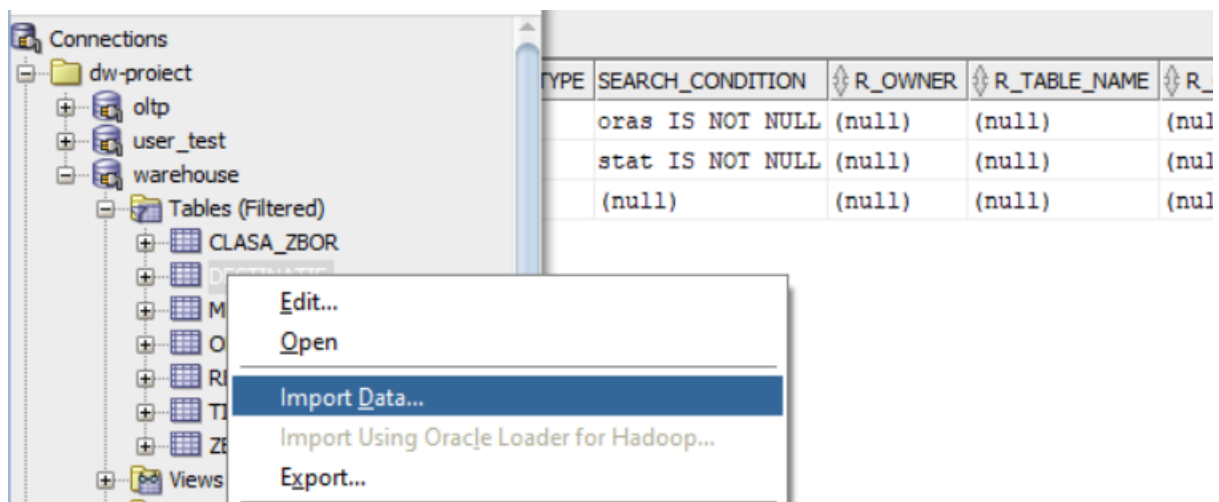
with open('reservation.csv','w') as fout:

fout.write('rezervare_id,nr_pasageri,nr_pasageri_femei,nr_pasageri
_barbati,data_rezervare,suma_totala,client_id,zbor_id,clasa_id,pla
ta\n')

for row in rows:
    fout.write(','.join(row))
    fout.write('\n')

```

Pentru tabelele CLIENT, REZERVARE, ZBOR, DESTINATIE și OPERATOR_ZBOR s-au folosit fișiere csv pentru populate, acestea fiind încărcate direct din SQL Developer.



Atât vizualizarea, cât și inserarea datelor pot fi realizate și din modulul aplicație al proiectului, iar rezultatele pot fi vazute în fisierul *aplicație*.

Pentru tabelele CLIENT, REZERVARE, ZBOR, DESTINATIE și OPERATOR_ZBOR trebuie să mai creăm secvențele corespunzătoare, iar celelalte tabele vor fi populate manual cu date.

CLIENT:

```
SELECT * FROM CLIENT;
```

```
CREATE SEQUENCE client_seq
START WITH 10001
INCREMENT BY 1;
```

CLIENT_ID	NUME	PRENUME	EMAIL	NUMAR_TELEFON
348	Butler	Sue	ybeck@example.org	600.569.1342
349	Price	Alex	baxtergreg@example.com	541-691-9625x8495
350	Hodge	Allison	lynningram@example.org	373.061.0884x033
351	Miles	Andrew	perkinscourtney@example.net	085-361-1960
352	Grant	Johnathan	mtaylor@example.com	(600) 098-5586x3837
353	Holland	Dustin	stanley10@example.org	(311) 207-1214x845
354	Sanchez	Christie	sean18@example.net	669.648.4509x782
355	Dixon	Eric	fvelazquez@example.com	-8459
356	Mills	Marisa	katrinaowens@example.com	036-815-4488x20732
357	Allison	Terrance	april89@example.net	(196) 224-1350
358	Ayers	Max	ellisonpamela@example.org	398-052-0380x2662
359	Howard	Adrian	mooneymichele@example.com	846-517-9600x05303
360	Moran	Leonard	kristen44@example.com	896.648.4185x4776

```
SELECT COUNT(*) FROM CLIENT;
```

Output	Explain Plan	Query Result	Query Result 1
SQL All Rows Fetched: 1 in 0.01 seconds			
COUNT(*)	10000		

DESTINATIE:

```
SELECT * FROM DESTINATIE;
```

```
CREATE SEQUENCE destinatie_seq
START WITH 323
INCREMENT BY 1;
```

DESTINATIE_ID	ORAS	STAT
ABE	Allentown	PA
ABI	Abilene	TX
ABQ	Albuquerque	NM
ABR	Aberdeen	SD
ABY	Albany	GA
ACK	Nantucket	MA
ACT	Waco	TX
ACV	Arcata/Eureka	CA
ACY	Atlantic City	NJ
ADW	Adelphi	NY

```
SELECT COUNT(*) FROM DESTINATIE;
```

Output	Explain Plan	Query Result	Query Result 1
SQL All Rows Fetched: 1 in 0.003 seconds			
COUNT(*)			
322			

ZBOR:

```
SELECT * FROM ZBOR;
CREATE SEQUENCE zbor_seq
START WITH 1048576
INCREMENT BY 1;
```

ZBOR_ID	OPERATOR_ID	AERONAVA_ID	DU...	DISTANTA	TOTAL_LOCURI	ANULAT	DATA_PLECARE	DATA_SOSIRE	LOCATIE_PLECARE_ID	LOCATIE_SOSIRE_ID	
394 B6	N247JB	50	266	90	0	01-JAN-15	10.05.00.000000000	AM 01-JAN-15	10.55.00.000000000	AM BTV	JFK
395 B6	N559JB	148	1069	80	0	01-JAN-15	10.05.00.000000000	AM 01-JAN-15	12.33.00.000000000	PM JFK	FLL
396 B6	N821JB	59	354	70	0	01-JAN-15	10.05.00.000000000	AM 01-JAN-15	11.04.00.000000000	AM SFO	LGB
397 B6	N809JB	163	1197	100	0	01-JAN-15	10.05.00.000000000	AM 01-JAN-15	12.48.00.000000000	PM FBI	BOS
398 B6	N641JB	129	944	90	0	01-JAN-15	10.05.00.000000000	AM 01-JAN-15	12.14.00.000000000	PM MCO	JFK
399 B6	N503JB	165	1189	90	0	01-JAN-15	10.05.00.000000000	AM 01-JAN-15	12.50.00.000000000	PM SJU	MCO
400 DL	N975DL	130	859	90	0	01-JAN-15	10.05.00.000000000	AM 01-JAN-15	12.15.00.000000000	PM BDL	ATL
401 DL	N3759	57	356	80	0	01-JAN-15	10.05.00.000000000	AM 01-JAN-15	11.02.00.000000000	AM RDU	ATL
402 DL	N336NB	88	599	100	0	01-JAN-15	10.05.00.000000000	AM 01-JAN-15	11.33.00.000000000	AM SFO	SLC
403 EV	N14143	117	936	80	0	01-JAN-15	10.05.00.000000000	AM 01-JAN-15	12.02.00.000000000	PM TUS	IAH
404 EV	N933EV	83	565	100	0	01-JAN-15	10.05.00.000000000	AM 01-JAN-15	11.28.00.000000000	AM PIA	ATL
405 F9	N223FR	150	977	100	0	01-JAN-15	10.05.00.000000000	AM 01-JAN-15	12.35.00.000000000	PM IND	DEN

```
SELECT COUNT(*) FROM ZBOR;
```

Output	Explain Plan	Query Result	Query Result 1
SQL All Rows Fetched: 1 in 0.214 seconds			
COUNT(*)			
1048575			

REZERVARE:

```
SELECT * FROM REZERVARE;
```

```
CREATE SEQUENCE rezervare_seq
START WITH 1040001
INCREMENT BY 1;
```

REZERVA...	NR_PASAGERI	NR_PASAGERI_FEMEI	NR_PASAGERI_BARBATI	DATA_REZERVARE	SUMA_TOTALA	CLIENT_ID	ZBOR_ID	CLASA_ZBOR_ID	METODA_PLATA_ID
490	7	6		1 09-JUN-15 12.00.00.0000000000 AM	197	8395	132053	1	3
491	6	0		6 26-OCT-15 12.00.00.0000000000 AM	7229	5502	424841	2	2
492	5	2		3 05-NOV-15 12.00.00.0000000000 AM	658	4869	95464	1	2
493	3	0		3 13-MAY-15 12.00.00.0000000000 AM	7077	271	189289	3	1
494	5	0		5 26-FEB-15 12.00.00.0000000000 AM	9639	6938	833757	1	3
495	5	0		5 10-DEC-15 12.00.00.0000000000 AM	882	8954	408996	3	3
496	9	7		2 18-APR-15 12.00.00.0000000000 AM	4274	4006	1017593	1	2
497	7	3		4 15-NOV-15 12.00.00.0000000000 AM	4050	2525	975217	2	1
498	4	4		0 23-MAR-15 12.00.00.0000000000 AM	8558	4969	525075	3	2

```
SELECT COUNT(*) FROM REZERVARE;
```

SQL	All Rows Fetched: 1 in 0.428 seconds
COUNT(*)	1040000

OPERATOR_ZBOR:

```
SELECT * FROM OPERATOR_ZBOR;
CREATE SEQUENCE operator_seq
START WITH 15
INCREMENT BY 1;
```

OPERATOR_ID	NUME
UA	United Air Lines Inc.
AA	American Airlines Inc.
US	US Airways Inc.
F9	Frontier Airlines Inc.
B6	JetBlue Airways
OO	Skywest Airlines Inc.
AS	Alaska Airlines Inc.
NK	Spirit Air Lines
WN	Southwest Airlines Co.
DL	Delta Air Lines Inc.
EV	Atlantic Southeast Airlines
HA	Hawaiian Airlines Inc.

```
SELECT COUNT(*) FROM OPERATOR_ZBOR;
```

SQL	All Rows Fetched: 1 in 0.002 seconds
COUNT(*)	14

CLASA_ZBOR:

```
CREATE SEQUENCE clasa_zbor_seq
START WITH 1
INCREMENT BY 1;

INSERT INTO CLASA_ZBOR VALUES(clasa_zbor_seq.NEXTVAL, 'FIRST');
INSERT INTO CLASA_ZBOR VALUES(clasa_zbor_seq.NEXTVAL, 'BUSINESS');
INSERT INTO CLASA_ZBOR VALUES(clasa_zbor_seq.NEXTVAL, 'ECONOMY');

SELECT * FROM CLASA_ZBOR;
```

CLASA_ZBOR_ID	DENUMIRE
1	FIRST
2	BUSINESS
3	ECONOMY

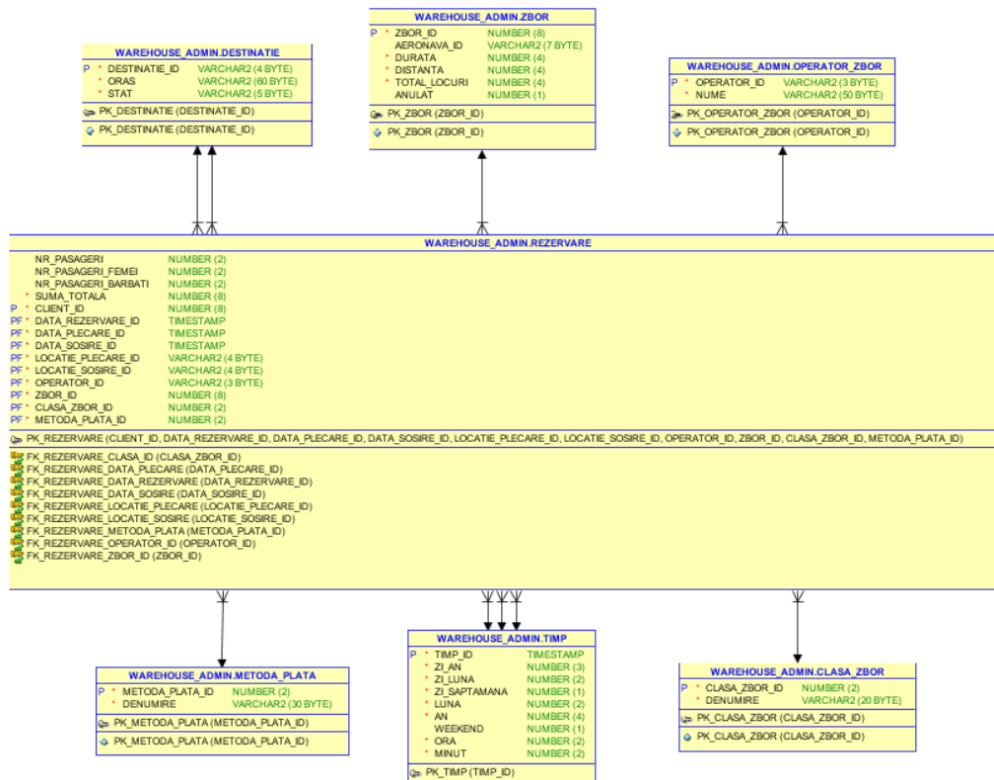
METODA_PLATA:

```
CREATE SEQUENCE metoda_plata_seq
START WITH 1
INCREMENT BY 1;

INSERT INTO METODA_PLATA VALUES(metoda_plata_seq.NEXTVAL, 'CASH');
INSERT INTO METODA_PLATA VALUES(metoda_plata_seq.NEXTVAL, 'CARD');
INSERT INTO METODA_PLATA VALUES(metoda_plata_seq.NEXTVAL, 'TRANSFER BANCAR');
```

METODA_PLATA_ID	DENUMIRE
1	CASH
2	CARD
3	TRANSFER BANCAR

5. Diagrama stea a bazei de date depozit



6. Descrierea câmpurilor necesare pentru fiecare tabel al bazei de date depozit împreună cu modul de populare cu informații

În cadrul bazei de date depozit, tabela REZERVARE devine tabela *fapte*, în timp ce tabelele ZBOR, OPERATOR_ZBOR, DESTINATIE, METODA_PLATA și CLASA_ZBOR devin tabele *dimensiune*. Întrucât cerințele atinse de aplicație nu au ca scop afișarea datelor personale pentru fiecare client, tabela CLIENTI a fost eliminată din modelul warehouse. În plus, a fost adăugată o nouă dimensiune, TIMP, care ne va ajuta în continuare la modelarea diverselor cereri.

Tabela de fapte REZERVARE are o cheie primară compusă, formată din cheile străine către dimensiuni, împreună cu *client_id*: *zbor_id*, *operator_id*, *locatie_plecure_id*, *locatie_sosire_id*, *data_plecure_id*, *data_sosire_id*, *data_rezervare_id*, *clasa_id*, *metoda_plata_id*. Restul coloanelor sunt de tip numeric (*nr_pasageri*, *nr_pasageri_femei*, *nr_pasageri_barbati*, *suma_totală*) și reprezintă metricile ce pot fi utilizate ulterior în cererile SQL.

Tabelele *dimensiune*:

- *Destinație* - păstrează coloanele din *oltp*: id-ul destinației, orasul și statul
- *Zbor* - dintre coloanele din *oltp*, cele care rămân sunt: *id-ul zborului*, *id-ul aeronavei* folosite, *durata*, respectiv *distanța* zborului, *numărul total de locuri* și un *number(1)* *anulat*, care ia valoarea 0 dacă zborul a avut loc, respectiv 1 dacă a fost anulat.
- *Operator zbor* - păstrează coloanele din *oltp*: id-ul operatorului și numele acestuia
- *Timp* - conține: id-ul de tip *timestamp*, *ziua*, raportată la *an*, *lună* și *săptămână*, *anul*, *ora* și *minutul*, iar în final un *number(1)* *weekend*, calculat pe baza coloanei *zi_saptamana*, cu valoarea 0, dacă ziua e în timpul săptămânii și 1 dacă este în weekend.
- *Metoda plata* - păstrează coloanele din *oltp*: id-ul metodei și denumirea acesteia, care are valorile inițiale: *cash*, *card* și *transfer bancar*.
- *Clasa zbor* - păstrează coloanele din *oltp*: id-ul clasei și denumirea - *I*, *business* și *economic*.

În ceea ce privește popularea bazei de date *warehouse* cu date din *OLTP*, am definit o procedură ETL, care va lua în pasul inițial toate datele din baza de date *OLTP* și le va trimite, conform noii structuri, în baza de date *warehouse*. De asemenea, în cazul modificărilor în baza de date sursă, am definit *triggeri* pe *OLTP* care vor propaga schimbările în *warehouse*.

7. Crearea bazei de date depozit și a utilizatorilor

REZERVARE:

```
CREATE TABLE REZERVARE
  (nr_pasageri NUMBER(2) CONSTRAINT ck_rezervare_nr_pasageri
CHECK (nr_pasageri between 1 and 10),
  nr_pasageri_femei NUMBER(2) CONSTRAINT
ck_rezervare_nr_pasageri_femei CHECK (nr_pasageri_femei between 0
and 10),
  nr_pasageri_barbati NUMBER(2) CONSTRAINT
ck_rezervare_nr_pasageri_barbati CHECK (nr_pasageri_barbati
between 0 and 10),
  suma_totala NUMBER(8) CONSTRAINT ck_rezervare_suma_totala
CHECK (suma_totala IS NOT NULL),
  client_id NUMBER(8) CONSTRAINT ck_rezervare_client_id CHECK
(client_id IS NOT NULL),
  data_rezervare_id TIMESTAMP,
  data_plecure_id TIMESTAMP,
  data_sosire_id TIMESTAMP,
```

```
locatie_plecare_id VARCHAR2(4),  
locatie_sosire_id VARCHAR2(4),  
operator_id VARCHAR2(3),  
zbor_id NUMBER(8),  
clasa_zbor_id NUMBER(2),  
metoda_plata_id NUMBER(2));
```

DESTINATIE:

```
CREATE TABLE DESTINATIE  
(destinatie_id VARCHAR2(4),  
oras VARCHAR2(60) CONSTRAINT ck_destinatie_oras CHECK(oras IS  
NOT NULL),  
stat VARCHAR2(5) CONSTRAINT ck_destinatie_stat CHECK(stat IS  
NOT NULL));
```

ZBOR:

```
CREATE TABLE ZBOR  
(zbor_id NUMBER(8),  
aeronava_id VARCHAR2(7),  
durata NUMBER(4) CONSTRAINT ck_zbor_durata CHECK(durata IS  
NOT NULL),  
distanta NUMBER(4) CONSTRAINT ck_zbor_distanta CHECK(distanta  
IS NOT NULL),  
total_locuri NUMBER(4) CONSTRAINT ck_zbor_total_locuri  
CHECK(total_locuri IS NOT NULL),  
anulat NUMBER(1) CONSTRAINT ck_zbor_anulat CHECK(anulat IN  
(0, 1)));
```

OPERATOR_ZBOR:

```
CREATE TABLE OPERATOR_ZBOR  
(operator_id VARCHAR2(3),  
nume VARCHAR2(50) CONSTRAINT ck_operator_nume CHECK(nume IS  
NOT NULL));
```

METODA_PLATA:

```
CREATE TABLE METODA_PLATA  
(metoda_plata_id NUMBER(2),
```

```
denumire VARCHAR2(30) CONSTRAINT ck_metoda_plata_denumire  
CHECK(denumire IS NOT NULL));
```

CLASA_ZBOR:

```
CREATE TABLE CLASA_ZBOR  
(clasa_zbor_id NUMBER(2),  
denumire VARCHAR2(20) CONSTRAINT ck_clasa_denumire  
CHECK(denumire IS NOT NULL));
```

TIMP:

```
CREATE TABLE TIMP(  
    timp_id TIMESTAMP,  
    zi_an NUMBER(3) CONSTRAINT ck_timp_zi_an CHECK(zi_an IS NOT  
NULL),  
    zi_luna NUMBER(2) CONSTRAINT ck_timp_zi_luna CHECK(zi_luna IS  
NOT NULL),  
    zi_saptamana NUMBER(1) CONSTRAINT ck_timp_zi_saptamana  
CHECK(zi_saptamana IS NOT NULL),  
    luna NUMBER(2) CONSTRAINT ck_timp_luna CHECK(luna IS NOT  
NULL),  
    an NUMBER(4) CONSTRAINT ck_timp_an CHECK(an IS NOT NULL),  
    weekend NUMBER(1) CONSTRAINT ck_timp_weekend CHECK(weekend IN  
(0, 1)),  
    ora NUMBER(2) CONSTRAINT ck_timp_ora CHECK(ora IS NOT NULL),  
    minut NUMBER(2) CONSTRAINT ck_timp_minut CHECK(minut IS NOT  
NULL));
```

Crearea utilizatorilor *warehouse* și acordarea drepturilor:

```
CREATE USER warehouse_admin IDENTIFIED BY pass_warehouse;  
GRANT CREATE SESSION TO warehouse_admin;  
GRANT CREATE TABLE TO warehouse_admin;  
ALTER USER warehouse_admin QUOTA UNLIMITED ON USERS;
```

8. Popularea cu informații a bazei de date depozit folosind ca sursă datele din baza de date OLTP

Crearea procedurii de transmitere a datelor implică acordarea următoarelor privilegii din *sys* pentru *admin_oltp*:


```

GRANT CREATE PROCEDURE TO oltp_admin;
GRANT SELECT, INSERT, DELETE, UPDATE ON warehouse_admin.rezervare
TO oltp_admin;
GRANT SELECT, INSERT, DELETE, UPDATE ON warehouse_admin.DESTINATIE
TO oltp_admin;
GRANT SELECT, INSERT, DELETE, UPDATE ON
warehouse_admin.OPERATOR_ZBOR TO oltp_admin;
GRANT SELECT, INSERT, DELETE, UPDATE ON
warehouse_admin.METODA_PLATA TO oltp_admin;
GRANT SELECT, INSERT, DELETE, UPDATE ON warehouse_admin.CLASA_ZBOR
TO oltp_admin;
GRANT SELECT, INSERT, DELETE, UPDATE ON warehouse_admin.ZBOR TO
oltp_admin;
GRANT SELECT, INSERT, DELETE, UPDATE ON warehouse_admin.TIMP TO
oltp_admin;

```

Pentru transferul inițial al datelor din baza de date *OLTP* în baza de date warehouse, s-a creat următoarea procedură:

```

CREATE OR REPLACE PROCEDURE creeaza_tabela_timp AS
    l_current_date TIMESTAMP;
    l_end_date TIMESTAMP;
BEGIN
    l_current_date := to_timestamp('2015-01-01 00:00:00',
'YYYY-MM-DD hh24:mi:ss');
    l_end_date := to_timestamp('2015-12-31 23:59:00', 'YYYY-MM-DD
hh24:mi:ss');
    WHILE l_current_date <= l_end_date LOOP
        INSERT INTO warehouse_admin.timp(timp_id, zi_an, zi_luna,
zi_saptamana, luna, an, weekend, ora, minut)
VALUES
        (l_current_date,
        TO_CHAR(l_current_date, 'DDD'),
        TO_CHAR(l_current_date, 'DD'),
        TO_CHAR(l_current_date, 'D'),
        TO_CHAR(l_current_date, 'MM'),
        TO_CHAR(l_current_date, 'YYYY'),
        CASE WHEN(TO_CHAR(l_current_date, 'D') BETWEEN 2 AND 6) THEN
0 ELSE 1 END,
        TO_CHAR(l_current_date, 'hh24'),
        TO_CHAR(l_current_date, 'mi'));
    END LOOP;
END;

```

```

        l_current_date := l_current_date + interval '1' minute;
    END LOOP;
END;
/

CREATE OR REPLACE PROCEDURE etl AS
BEGIN
    INSERT INTO warehouse_admin.ZBOR SELECT zbor_id, aeronava_id,
durata, distanta, total_locuri, anulat FROM zbor;
    INSERT INTO warehouse_admin.REZERVARE
        (SELECT nr_pasageri, nr_pasageri_femei,
nr_pasageri_barbati, suma_totala, client_id, data_rezervare,
        data_plecare, data_sosire, locatie_plecare_id,
locatie_sosire_id, operator_id, a.zbor_id, clasa_zbor_id,
metoda_plata_id
        FROM rezervare a
        JOIN zbor b ON (a.zbor_id = b.zbor_id));
    INSERT INTO warehouse_admin.OPERATOR_ZBOR SELECT * FROM
oltp_admin.OPERATOR_ZBOR;
    INSERT INTO warehouse_admin.METODA_PLATA SELECT * FROM
METODA_PLATA;
    INSERT INTO warehouse_admin.CLASA_ZBOR SELECT * FROM
CLASA_ZBOR;
    INSERT INTO warehouse_admin.DESTINATIE SELECT * FROM
DESTINATIE;
    creeaza_tabela_timp;
    COMMIT;
END;
/

```

De asemenea, au fost creați triggeri pentru toate tabelele care detectează insert-urile:

```

CREATE OR REPLACE TRIGGER insert_plata_warehouse AFTER INSERT ON
metoda_plata FOR EACH ROW
BEGIN
    INSERT INTO warehouse_admin.metoda_plata VALUES
(:NEW.metoda_plata_id, :NEW.denumire);
END;
/

CREATE OR REPLACE TRIGGER insert_destinatie_warehouse AFTER INSERT
ON destinatie FOR EACH ROW

```

```

BEGIN
    INSERT INTO warehouse_admin.destinatie VALUES
(:NEW.destinatie_id, :NEW.oras, :NEW.stat);
END;
/

CREATE OR REPLACE TRIGGER insert_clasa_warehouse AFTER INSERT ON
clasa_zbor FOR EACH ROW
BEGIN
    INSERT INTO warehouse_admin.clasa_zbor VALUES
(:NEW.clasa_zbor_id, :NEW.denumire);
END;
/

CREATE OR REPLACE TRIGGER insert_operator_warehouse AFTER INSERT
ON operator_zbor FOR EACH ROW
BEGIN
    INSERT INTO warehouse_admin.operator_zbor VALUES
(:NEW.operator_id, :NEW.num);
END;
/

-- Procedura care insereaza in tabela timp in cazul in care
inregistrarea nu exista deja
CREATE OR REPLACE PROCEDURE proc_add_into_timp(p_data DATE) AS
    v_zi_an NUMBER;
    v_zi_luna NUMBER;
    v_zi_saptamana NUMBER;
    v_luna NUMBER;
    v_an NUMBER;
    v_weekend NUMBER;
    v_ora NUMBER;
    v_minut NUMBER;
    v_count NUMBER;
BEGIN
    -- Extrage fiecare camp din data
    SELECT TO_CHAR(p_data, 'DDD') INTO v_zi_an
    FROM DUAL;

    SELECT TO_CHAR(p_data, 'DD') INTO v_zi_luna
    FROM DUAL;

```

```

SELECT TO_CHAR(p_data, 'D') INTO v_zi_saptamana
FROM DUAL;

SELECT EXTRACT(MONTH FROM p_data) INTO v_luna
FROM DUAL;

SELECT EXTRACT(YEAR FROM p_data) INTO v_an
FROM DUAL;

IF v_zi_saptamana BETWEEN 6 AND 7 THEN
    v_weekend := 1;
ELSE
    v_weekend := 0;
END IF;

SELECT TO_CHAR(p_data, 'hh24') INTO v_oracle
FROM DUAL;

SELECT TO_CHAR(p_data, 'mi') INTO v_minut
FROM DUAL;

-- Se verifica daca aceasta data exista deja, iar in caz contrar
este inserata in tabela Timp
SELECT COUNT(*) INTO v_count FROM warehouse_admin.timp t
WHERE t.an = v_an
AND t.luna = v_luna
AND t.zi_luna = v_zi_luna
AND t.ora = v_oracle
AND t.minut = v_minut;

IF v_count = 0 THEN
    INSERT INTO warehouse_admin.timp VALUES(p_data, v_zi_an,
v_zi_luna, v_zi_saptamana, v_luna, v_an, v_weekend, v_oracle,
v_minut);
END IF;

END;
/

CREATE OR REPLACE TRIGGER insert_rezervare_warehouse AFTER INSERT
ON rezervare FOR EACH ROW

```

```

DECLARE
    data_plecare_id
warehouse_admin.rezervare.data_plecare_id%type;
    data_sosire_id warehouse_admin.rezervare.data_sosire_id%type;
    locatie_plecare_id
warehouse_admin.rezervare.locatie_plecare_id%type;
    locatie_sosire_id
warehouse_admin.rezervare.locatie_sosire_id%type;
    operator_id warehouse_admin.rezervare.operator_id%type;
BEGIN
    select data_plecare
into data_plecare_id
from zbor z
where z.zbor_id = :NEW.zbor_id;

    select data_sosire
into data_sosire_id
from zbor z
where z.zbor_id = :NEW.zbor_id;

    select locatie_plecare_id
into locatie_plecare_id
from zbor z
where z.zbor_id = :NEW.zbor_id;

    select locatie_sosire_id
into locatie_sosire_id
from zbor z
where z.zbor_id = :NEW.zbor_id;

    select operator_id
into operator_id
from zbor z
where z.zbor_id = :NEW.zbor_id;

-- se adauga datele in tabela Timp daca acestea nu exista deja
proc_add_into_timp(:NEW.data_rezervare);
proc_add_into_timp(data_sosire_id);
proc_add_into_timp(data_plecare_id);

    INSERT INTO warehouse_admin.rezervare VALUES (:NEW.nr_pasageri,
:NEW.nr_pasageri_femei, :NEW.nr_pasageri_barbati,

```

```

:NEW.suma_totala, :NEW.client_id, :NEW.data_rezervare,
data_plecare_id, data_sosire_id, locatie_plecare_id,
locatie_sosire_id,
operator_id, :NEW.zbor_id, :NEW.clasa_zbor_id,
:NEW.metoda_plata_id);
END;
/

CREATE OR REPLACE TRIGGER insert_zbor_warehouse AFTER INSERT ON
zbor FOR EACH ROW
BEGIN
INSERT INTO warehouse_admin.zbor VALUES (:NEW.zbor_id,
:NEW.aeronava_id, :NEW.durata, :NEW.distanta, :NEW.total_locuri,
:NEW.anulat);
END;
/

```

Pentru a testa triggerii, am făcut o inserare în tabelul rezervări din OLTP și am verificat apoi dacă a fost adăugată în tabelul corespunzător din warehouse.

```

insert into rezervare values (1900003, 5, 2, 3,
to_timestamp('2015-01-01 13:00:00', 'YYYY-MM-DD hh24:mi:ss'),
6000, 1, 2, 1, 1);

```

```

1 row inserted.

Commit complete.

```

```

SELECT * FROM rezervare where client_id = 1
AND zbor_id = 2
AND clasa_zbor_id = 1
AND suma_totala = 6000
AND metoda_plata_id = 1;

```

NR_PASAGERI	NR_PASAGERI_FEMEI	NR_PASAGERI_BARBATI	SUMA_TOTALA	CLIENT_ID	DATA_REZERVARE_ID	DATA_PLECARE_ID	DATA_SOSIRE_ID
1	5	2	3	6000	1	01-JAN-15 13.00.00.0000000000	01-JAN-15 00.10.00.0000000000
						01-JAN-15 04.33.00.000000	

9. Identificarea constrângerilor

În momentul creării tabelelor, am adăugat constrângeri de tip check. Alături de acestea, am definit următoarele constrângeri specifice bazelor de date warehouse:

- Constrângere de tip *disable validate* pentru cheia primară din tabela fapte REZERVARE:
 - Cheia primară din tabela REZERVARE trebuie să fie formată din cheile externe spre tabelele dimensiuni
 - În cazul în care am fi creat cheia primară fără nicio specificație, aceasta ar fi fost de tipul *enable validate noverify*. Astfel, toate datele ar fi fost validate și s-ar fi creat un index unic.
 - În continuare vrem să păstrăm opțiunea *validate*, deoarece verifică dacă toate datele existente ale tabelului verifică constrângerea - astfel se va asigura consistența datelor
 - Însă tabela *fapte* fiind foarte mare, având peste un milion de rows, nu vrem să creăm și un index, deoarece este costisitor și va fi rar utilizat pentru procesarea datelor - astfel, vom da opțiunea *disable validate*
 - Însă această opțiune este read-only, astfel că trebuie să modificăm constrângerea în *enable* de fiecare dată când vrem să inserăm date în baza de date depozit
- Constrângere de tip *rely disable novalidate* pentru cheia primară din tabela TIMP:
 - În modul în care am definit procesul ETL, putem să ne asigurăm că tabela timp nu va avea niciodată chei primare duplicate (cheia primară fiind un atribut de tip *timestamp*). Aplicația își propune să populeze tabela timp o dată pe an, la jumătatea acestuia, cu toate datele din anul viitor, pentru fiecare minut. Astfel, ca administratori ai bazei de date, vom avea încredere că acestea au fost introduse corect
 - Astfel, nu vom crea un index pentru cheie primară (opțiunea *disable*), nu vrem să validăm datele (*novalidate*), deoarece știm că vor fi construite corespunzător și îi comunicăm serverului prin *rely* că acea constrângere este satisfăcută
 - *Rele disable novalidate* va fi variantă foarte puțin costisitoare, având un consum mic de resurse, fiind potrivită în acest caz
- Constrângere de tip *rele disable novalidate* pentru cheile externe din tabela REZERVARI spre tabela TIMP:
 - Ca și în cazul cheii primare de pe tabela timp, se va cunoaște faptul că procesul ETL populează corespunzător tabelele. Coloanele *data_plecure_id*,

data_sosire_id și *data_rezervare_id* nu vor putea avea valori mai mari decât cele ce există deja în tabela TIMP, deoarece zborurile sunt programate cu doar jumătate de an înainte de a avea loc, iar rezervările nu pot avea o dată din viitor.

- Astfel, cea mai potrivită variantă de cheie externă este *relly disable novalidate*, pentru a avea un consum mic de resurse și a nu crea un index
- Constrângeri de tip *enable novalidate* pentru restul cheilor externe din tabela REZERVARE:
 - Împlicit, o constrângere de cheie străină e construită cu opțiunile *enable validate*
 - Însă în baza de date depozit, este posibil ca datele să nu vină sincronizate, astfel că înainte de finalizarea tranzacției, se poate întâmpla ca cheile străine să nu fie satisfăcute
 - În acest sens, vom păstra opțiunea *enable*, deoarece vrem să vedem că este satisfăcută în final, însă vom dezactiva validarea (*enable novalidate*)
- Constrângeri de tip *enable validate* pentru restul tabelelor dimensiune:
 - Deși este recomandat ca pentru cheile primare să se folosească opțiunea *disable validate*, pentru a evita crearea indexului, în acest caz, deoarece vrem ca FK din tabela fapte să fie de tipul *enable novalidate*, nu se poate crea PK de tipul *disable validate* (este necesară o cheie primară *enabled*)
 - Deoarece tabelele dimensiuni sunt mai mici decât tabela fapte și vrem și să validăm că nu sunt introduse date eronate, vom păstra opțiunea implicită *enable validate*

Alături de constrângerile *check* definite la crearea tabelelor, am adăugat următoarele constângeri:

Cheia primară în tabela de fapte - compusă, de tip DISABLE VALIDATE:

```
ALTER TABLE rezervare
ADD CONSTRAINT pk_rezervare
PRIMARY KEY(client_id, data_rezervare_id, data_plecare_id,
data_sosire_id, locatie_plecare_id, locatie_sosire_id,
operator_id, zbor_id, clasa_zbor_id, metoda_plata_id)
DISABLE VALIDATE;
```

Cheia primară în tabela destinație:

```
ALTER TABLE destinatie
ADD CONSTRAINT pk_destinatie
```



```
PRIMARY KEY(destinatie_id)
ENABLE VALIDATE;
```

Cheile străine către DESTINATIE din tabela de fapte - tip ENABLE NOVALIDATE:

```
ALTER TABLE rezervare
ADD CONSTRAINT fk_rezervare_locatie_plecure
FOREIGN KEY(locatie_plecure_id)
REFERENCES DESTINATIE(destinatie_id)
ENABLE NOVALIDATE;
```

```
ALTER TABLE rezervare
ADD CONSTRAINT fk_rezervare_locatie_sosire
FOREIGN KEY(locatie_sosire_id)
REFERENCES DESTINATIE(destinatie_id)
ENABLE NOVALIDATE;
```

Cheie primară pentru TIMP:

```
ALTER TABLE timp
ADD CONSTRAINT pk_timp
PRIMARY KEY(timp_id)
RELY DISABLE NOVALIDATE;
```

Chei străine spre tabela TIMP:

```
ALTER TABLE rezervare
ADD CONSTRAINT fk_rezervare_data_plecure
FOREIGN KEY(data_plecure_id)
REFERENCES TIMP(timp_id)
RELY DISABLE NOVALIDATE;
```

```
ALTER TABLE rezervare
ADD CONSTRAINT fk_rezervare_data_sosire
FOREIGN KEY(data_sosire_id)
REFERENCES TIMP(timp_id)
RELY DISABLE NOVALIDATE;
```

```
-- FK pentru data_rezervare
ALTER TABLE rezervare
```

```
ADD CONSTRAINT fk_rezervare_data_rezervare
FOREIGN KEY(data_rezervare_id)
REFERENCES TIMP(timp_id)
RELY DISABLE NOVALIDATE;
```

Cheie primară în OPERATOR_ZBOR și cheia străină spre acesta din tabela fapte:

```
ALTER TABLE operator_zbor
ADD CONSTRAINT pk_operator_zbor
PRIMARY KEY(operator_id);

ALTER TABLE rezervare
ADD CONSTRAINT fk_rezervare_operator_id
FOREIGN KEY(operator_id)
REFERENCES OPERATOR_ZBOR(operator_id)
ENABLE NOVALIDATE;
```

Cheie primară în ZBOR și cheia străină spre acesta din tabela fapte:

```
ALTER TABLE zbor
ADD CONSTRAINT pk_zbor
PRIMARY KEY(zbor_id);

ALTER TABLE rezervare
ADD CONSTRAINT fk_rezervare_zbor_id
FOREIGN KEY(zbor_id)
REFERENCES ZBOR(zbor_id)
ENABLE NOVALIDATE;
```

Cheie primară în CLASA_ZBOR și cheia străină spre acesta din tabela fapte:

```
ALTER TABLE clasa_zbor
ADD CONSTRAINT pk_clasa_zbor
PRIMARY KEY(clasa_zbor_id);

ALTER TABLE rezervare
ADD CONSTRAINT fk_rezervare_clasa_id
FOREIGN KEY(clasa_zbor_id)
REFERENCES CLASA_ZBOR(clasa_zbor_id)
ENABLE NOVALIDATE;
```

Cheie primară în METODA_PLATA și cheia străină spre acesta din tabela fapte:

```
ALTER TABLE metoda_plata
ADD CONSTRAINT pk_metoda_plata
PRIMARY KEY(metoda_plata_id);

ALTER TABLE rezervare
ADD CONSTRAINT fk_rezervare_metoda_plata
FOREIGN KEY(metoda_plata_id)
REFERENCES METODA_PLATA(metoda_plata_id)
ENABLE NOVALIDATE;
```

Vizualizarea constrângerilor:

```
SELECT a.table_name, a.column_name, b.constraint_name, generated,
b.constraint_type, search_condition,
delete_rule, r_constraint_name, status, validated, rely
FROM user_cons_columns a, user_constraints b
WHERE a.constraint_name = b.constraint_name
AND a.table_name IN ('REZERVARE', 'DESTINATIE', 'CLASA_ZBOR',
'OPERATOR_ZBOR', 'METODA_PLATA', 'ZBOR', 'TIMP')
AND b.constraint_name NOT LIKE 'CK%';
```

TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	GENERATED	CONSTRAINT_TYPE	SEARCH_CONDITION	DELETE_RULE	R_CONSTRAINT_NAME	STATUS	VALIDATED	RELY
REZERVARE	CLIENT_ID	PK_REZERVARE	USER NAME	P	(null)	(null)	(null)	DISABLED	VALIDATED	(null)
REZERVARE	DATA_REZERVARE_ID	FK_REZERVARE	USER NAME	P	(null)	(null)	(null)	DISABLED	VALIDATED	(null)
REZERVARE	DATA_PLECARE_ID	FK_REZERVARE	USER NAME	P	(null)	(null)	(null)	DISABLED	VALIDATED	(null)
REZERVARE	DATA_SOSIRE_ID	FK_REZERVARE	USER NAME	P	(null)	(null)	(null)	DISABLED	VALIDATED	(null)
REZERVARE	LOCATIE_PLECARE_ID	FK_REZERVARE	USER NAME	P	(null)	(null)	(null)	DISABLED	VALIDATED	(null)
REZERVARE	LOCATIE_SOSIRE_ID	FK_REZERVARE	USER NAME	P	(null)	(null)	(null)	DISABLED	VALIDATED	(null)
REZERVARE	OPERATOR_ID	FK_REZERVARE	USER NAME	P	(null)	(null)	(null)	DISABLED	VALIDATED	(null)
REZERVARE	ZBOR_ID	FK_REZERVARE	USER NAME	P	(null)	(null)	(null)	DISABLED	VALIDATED	(null)
REZERVARE	CLASA_ZBOR_ID	FK_REZERVARE	USER NAME	P	(null)	(null)	(null)	DISABLED	VALIDATED	(null)
REZERVARE	METODA_PLATA_ID	FK_REZERVARE	USER NAME	P	(null)	(null)	(null)	DISABLED	VALIDATED	(null)
REZERVARE	LOCATIE_PLECARE_ID	FK_REZERVARE_LOCATIE_PLECARE	USER NAME	R	(null)	NO ACTION	FK_DESTINATIE	ENABLED	NOT VALIDATED	(null)
DESTINATIE	DESTINATIE_ID	PK_DESTINATIE	USER NAME	P	(null)	(null)	(null)	ENABLED	VALIDATED	(null)
REZERVARE	LOCATIE_SOSIRE_ID	FK_REZERVARE_LOCATIE_SOSIRE	USER NAME	R	(null)	NO ACTION	FK_DESTINATIE	ENABLED	NOT VALIDATED	(null)
TIMP	TIMP_ID	PK_TIMP	USER NAME	P	(null)	(null)	(null)	DISABLED	NOT VALIDATED	RELY
REZERVARE	DATA_PLECARE_ID	FK_REZERVARE_DATA_PLECARE	USER NAME	R	(null)	NO ACTION	PK_TIMP	DISABLED	NOT VALIDATED	RELY
REZERVARE	DATA_SOSIRE_ID	FK_REZERVARE_DATA_SOSIRE	USER NAME	R	(null)	NO ACTION	PK_TIMP	DISABLED	NOT VALIDATED	RELY
REZERVARE	DATA_REZERVARE_ID	FK_REZERVARE_DATA_REZERVARE	USER NAME	R	(null)	NO ACTION	PK_TIMP	DISABLED	NOT VALIDATED	RELY
OPERATOR_ZBOR	OPERATOR_ID	FK_OPERATOR_ZBOR	USER NAME	P	(null)	(null)	(null)	ENABLED	VALIDATED	(null)
REZERVARE	OPERATOR_ID	FK_REZERVARE_OPERATOR_ID	USER NAME	R	(null)	NO ACTION	FK_OPERATOR_ZBOR	ENABLED	NOT VALIDATED	(null)
ZBOR	ZBOR_ID	PK_ZBOR	USER NAME	P	(null)	(null)	(null)	ENABLED	VALIDATED	(null)
REZERVARE	ZBOR_ID	FK_REZERVARE_ZBOR_ID	USER NAME	R	(null)	NO ACTION	PK_ZBOR	ENABLED	NOT VALIDATED	(null)
CLASA_ZBOR	CLASA_ZBOR_ID	FK_CLASA_ZBOR	USER NAME	P	(null)	(null)	(null)	ENABLED	VALIDATED	(null)
REZERVARE	CLASA_ZBOR_ID	FK_REZERVARE_CLASA_ID	USER NAME	R	(null)	NO ACTION	PK_CLASA_ZBOR	ENABLED	NOT VALIDATED	(null)
METODA_PLATA	METODA_PLATA_ID	PK_METODA_PLATA	USER NAME	P	(null)	(null)	(null)	ENABLED	VALIDATED	(null)
REZERVARE	METODA_PLATA_ID	FK_REZERVARE_METODA_PLATA	USER NAME	R	(null)	NO ACTION	PK_METODA_PLATA	ENABLED	NOT VALIDATED	(null)

10. Identificarea indecșilor

Din motive de îmbunătățirea relațiilor cu clienții, în cadrul aplicației se dorește obținerea unor statistici cu privire la numărul de rezervări anulate care au fost plătite cu o anumită metodă de plată. Cum tabelele *Rezervare* și *Zbor* au un număr mare de înregistrări în tabele, se va defini un index de tip *bitmap join* între cele 2 pe coloana *anulat*, respectiv unul

de tip *bitmap* pe tabela *Rezervare* pe coloana *metoda_plata_id*. Index-ul va fi tot de timp bitmap întrucât există un număr limitat de valori posibile pentru tipurile de plată.

Pentru început, se definește cererea SQL, executată fără cei 2 indecși:

```
SELECT COUNT(*)
FROM rezervare r
JOIN zbor z ON r.zbor_id = z.zbor_id
WHERE z.anulat = 1 AND r.metoda_plata_id = 2;
```

Rezultatul ei:

	COUNT(*)
1	13374

De asemenea, se verifică și costul cererii, respectiv unul de **3966**.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
SORT				3966
HASH JOIN		AGGREGATE		1
Access Predicates				45165
R.ZBOR_ID=Z.ZBOR_ID				3966
TABLE ACCESS	ZBOR	FULL	40527	1186
Filter Predicates				
Z.ANULAT=1				
TABLE ACCESS	REZERVARE	FULL	346527	2777
Filter Predicates				
R.METODA_PLATA_ID=2				

Se vor defini cei 2 indecși menționați anterior după cum urmează:

```
CREATE BITMAP INDEX idx_rezervare_metoda_plata
ON rezervare(metoda_plata_id);
```

```
CREATE BITMAP INDEX idx_join_zbor_rezervare ON rezervare(z.anulat)
FROM rezervare r, zbor z
WHERE r.zbor_id = z.zbor_id;
```

În continuare, se activează generarea de statistici pentru indecși:

```
ANALYZE INDEX idx_join_zbor_rezervare COMPUTE STATISTICS;
ANALYZE INDEX idx_rezervare_metoda_plata COMPUTE STATISTICS;
```

Se rulează din nou cererea SQL, pentru care se verifică costul prin plan:

```
EXPLAIN PLAN
SET STATEMENT_ID = 's6' FOR
SELECT COUNT(*) /*+INDEX(r idx_join_zbor_rezervare) +INDEX(r
idx_rezervare_metoda_plata)*/
FROM rezervare r
```

```

JOIN zbor z ON r.zbor_id = z.zbor_id
WHERE z.anulat = 1 AND r.metoda_plata_id = 2;

SELECT plan_table_output
FROM table(dbms_xplan.display('plan_table', 's6','serial'));

```

PLAN_TABLE_OUTPUT							
1	Plan hash value: 1125178480						
2							
3	-----						
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5	-----						
6	0	SELECT STATEMENT		1	9	4 (0)	00:00:01
7	1	SORT AGGREGATE		1	9		
8	2	BITMAP CONVERSION COUNT		346K	3045K	4 (0)	00:00:01
9	3	BITMAP AND					
10	* 4	BITMAP INDEX SINGLE VALUE	IDX_JOIN_ZBOR_REZERVARE				
11	* 5	BITMAP INDEX SINGLE VALUE	IDX_REZERVARE_METODA_PLATA				
12	-----						

Se observă o scădere a costului, de la o valoare de ~**4000** la una de **4**.

Se verifică planul care confirmă același lucru:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
SORT		AGGREGATE		1
BITMAP CONVERSION		COUNT		346527
BITMAP AND				4
BITMAP INDEX	IDX_JOIN_ZBOR_REZERVARE	SINGLE VALUE		
Access Predicates				
R.SYS_NC00015\$=1				
BITMAP INDEX	IDX_REZERVARE_METODA_PLATA	SINGLE VALUE		
Access Predicates				
R.METODA_PLATA_ID=2				

În plus, indecșii pot să nu fie dați ca hint întrucât optimizatorul îi va folosi în continuare:

```

SELECT COUNT(*)
FROM rezervare r
JOIN zbor z ON r.zbor_id = z.zbor_id
WHERE z.anulat = 1 AND r.metoda_plata_id = 2;

```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
SORT		AGGREGATE		1
BITMAP CONVERSION		COUNT		346527
BITMAP AND				4
BITMAP INDEX	IDX_JOIN_ZBOR_REZERVARE	SINGLE VALUE		
Access Predicates				
R.SYS_NC00015\$=1				
BITMAP INDEX	IDX_REZERVARE_METODA_PLATA	SINGLE VALUE		
Access Predicates				
R.METODA_PLATA_ID=2				

11. Identificarea obiectelor de tip dimensiune

Am identificat că tabelul TIMP are date dimensionale, cu ajutorul cărora se poate crea un obiect de tip dimensiune. Astfel, am observat că se pot defini ierarhii la nivel de dată și

oră. Ziua, identificată prin timp_id are ca părinte luna, identificată prin timp.luna, care la rândul său are ca părinte anul, identificat prin timp.an. Am identificat, de asemenea, dependențe unidirecționale între atribute. Mai precis, unei valori a atributului zi îi corespunde o singură valoare a atributelor zi_saptamana, zi_luna, zi_an.

Am creat prima dată dimensiunea timp_dim.

```
create dimension timp_dim
level zi is (timp.timp_id)
level luna is (timp.luna)
level an is (timp.an)
hierarchy h
  (zi child of
   luna child of
   an)
ATTRIBUTE zi DETERMINES
      (TIMP.zi_saptamana,
       TIMP.zi_luna,
       TIMP.zi_an);
```

Pachetul demo_dim nu este definit implicit. Pentru a defini pachetul demo_dim am rulat fișierul smdim.sql, dat la laborator. Am vizualizat obiectul dimension:

```
set serveroutput on;
EXECUTE DEMO_DIM.PRINT_DIM ('timp_dim');
EXECUTE DBMS_OUTPUT.ENABLE(10000);
EXECUTE DEMO_DIM.PRINT_ALLDIMS;
```

```
DIMENSION WAREHOUSE_ADMIN.TIMP_DIM
  LEVEL AN IS WAREHOUSE_ADMIN.TIMP.AN
  LEVEL LUNA IS WAREHOUSE_ADMIN.TIMP.LUNA
  LEVEL ZI IS WAREHOUSE_ADMIN.TIMP.TIMP_ID

  HIERARCHY H (
    ZI
    CHILD OF LUNA
    CHILD OF AN
  )

  ATTRIBUTE ZI DETERMINES WAREHOUSE_ADMIN.TIMP.ZI_AN
  ATTRIBUTE ZI DETERMINES WAREHOUSE_ADMIN.TIMP.ZI_LUNA
  ATTRIBUTE ZI DETERMINES WAREHOUSE_ADMIN.TIMP.ZI_SAPTAMANA
```

PL/SQL procedure successfully completed.

Am consultat și vizualizarea user_dimensions din dicționarul datelor pentru a vedea obiectul dimension creat.

```
select dimension_name, invalid, compile_state
from   user_dimensions;
```

	DIMENSION_NAME	INVALID	COMPILE_STATE
1	TIMP_DIM	N	VALID

Am utilizat procedura *VALIDATE_DIMENSION* din pachetul *DBMS_DIMENSION* pentru a valida relațiile specificate în obiectul *dimension*.

```
EXECUTE
DBMS_DIMENSION.VALIDATE_DIMENSION(UPPER('timp_dim'),FALSE,TRUE,'st
_id2');
```

Am verificat dacă procedura *VALIDATE_DIMENSION* executată anterior a generat erori.

```
SELECT count(*)
FROM   timp
WHERE  ROWID IN (SELECT BAD_ROWID
                  FROM DIMENSION_EXCEPTIONS
                  WHERE STATEMENT_ID = 'st_id2');
```

	COUNT(*)
1	0

12. Identificarea tabelelor ce vor fi partiționate

Se vor analiza 3 tipuri de partiționare asupra tabelului de fapte REZERVARI pentru a vedea costul asociat cererilor de forma „Aflați numărul de rezervări făcute în luna ianuarie 2015 pentru zborurile operate de compania aeriană Hawaiian Airlines Inc.”. Variantele posibile sunt:

- Partiționarea *range* (prin ordonare) după coloana *data_plecare_id*
- Partiționarea *list* după coloana *operator_id*

- Partiționarea *compusă*, de tip *range* după *data_plecure_id*, cu subpartiții de tip *list* pentru *operator_id*

Vom implementa 3 tipuri de partiționare asupra tabelului REZERVARE, pentru a compara costurile și a alege metoda prin care optimizorul va utiliza cel mai eficient partițiile.

Pentru început aflăm rezultatul cererii SQL neoptimizate și costul asociat:

```
SELECT COUNT(data_rezervare_id) FROM rezervare
WHERE operator_id = 'HA'
AND data_plecure_id BETWEEN TO_DATE('2015-01-01 00:00:00',
'YYYY-MM-DD hh24:mi:ss')
AND TO_DATE('2015-01-31 00:00:00', 'YYYY-MM-DD
hh24:mi:ss');
```

	COUNT(DATA_REZERVARE_ID)
1	6028

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				2773
SORT		AGGREGATE	1	
TABLE ACCESS	REZERVARE	FULL	32437	2773
Filter Predicates				
AND				
OPERATOR_ID='HA'				
DATA_PLECURE_ID<=TIMESTAMP' 2015-01-31 00:00:00'				
DATA_PLECURE_ID>=TIMESTAMP' 2015-01-01 00:00:00'				

Costul pentru varianta nepartiționată este 2773.

1. Partiționarea *range* (prin ordonare) după coloana *data_plecure_id*

Creăm tabelul:

```
CREATE TABLE rezervare_ord_data_plecure
PARTITION BY RANGE(data_plecure_id)
( PARTITION rezervari_jan2015
VALUES LESS THAN(TO_TIMESTAMP('2015-02-01 00:00:00', 'YYYY-MM-DD
hh24:mi:ss')),
PARTITION rezervari_feb2015
VALUES LESS THAN(TO_DATE('2015-03-01 00:00:00', 'YYYY-MM-DD
hh24:mi:ss')),
PARTITION rezervari_mar2015
VALUES LESS THAN (MAXVALUE))
AS
SELECT *
```



```
FROM rezervare;
```

```
ANALYZE TABLE rezervare_ord_data_plecure COMPUTE STATISTICS;
```

Pentru început, calculăm costul fără a pune condiția operatorului de zbor:

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_rezervare_data_1'
FOR
SELECT COUNT(*)
FROM rezervare_ord_data_plecure
WHERE data_plecure_id BETWEEN TO_DATE('2015-01-01 00:00:00',
'YYYY-MM-DD hh24:mi:ss')
AND TO_DATE('2015-01-31 00:00:00', 'YYYY-MM-DD
hh24:mi:ss');

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_rezervare_data_1','seria
1'));
```

Costul asociat acestei cereri este 1208, însă fără condiția operatorului.

PLAN_TABLE_OUTPUT									
Plan hash value: 1751623316									
Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	
0	SELECT STATEMENT		1	14	1208 (2)	00:00:01			
1	SORT AGGREGATE		1	14					
2	PARTITION RANGE SINGLE		446K	6109K	1208 (2)	00:00:01	1	1	
* 3	TABLE ACCESS FULL	REZERVARE_ORD_DATA_PLECARE	446K	6109K	1208 (2)	00:00:01	1	1	

Adăugăm condiția pentru operator:

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_rezervare_data_2'
FOR
SELECT COUNT(*)
FROM rezervare_ord_data_plecure
WHERE data_plecure_id BETWEEN TO_DATE('2015-01-01 00:00:00',
'YYYY-MM-DD hh24:mi:ss')
AND TO_DATE('2015-01-31 00:00:00', 'YYYY-MM-DD hh24:mi:ss')
AND OPERATOR_ID = 'HA';
```

```
SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_rezervare_data_2','seria
1'));
```

PLAN_TABLE_OUTPUT									
Plan hash value: 1751623316									

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	

0	SELECT STATEMENT		1	9	1209 (2)	00:00:01			
1	SORT AGGREGATE		1	9					
2	PARTITION RANGE SINGLE		31916	280K	1209 (2)	00:00:01	1	1	
* 3	TABLE ACCESS FULL	REZERVARE_ORD_DATA_PLECARE	31916	280K	1209 (2)	00:00:01	1	1	

Costul final, pentru o partiționare de tip range după data_plecare_id pentru cererea SQL analizată este 1209 și se poate observa că singura partiție utilizată este 1.

2. Partiționarea *list* după coloana *operator_id*

Creăm tabelul:

```
CREATE TABLE rezervare_lista_operator
PARTITION BY LIST(operator_id)(
PARTITION rezervari_ua VALUES('UA'),
PARTITION rezervari_aa VALUES('AA'),
PARTITION rezervari_us VALUES('US'),
PARTITION rezervari_f9 VALUES('F9'),
PARTITION rezervari_b6 VALUES('B6'),
PARTITION rezervari_oo VALUES('OO'),
PARTITION rezervari_as VALUES('AS'),
PARTITION rezervari_nk VALUES('NK'),
PARTITION rezervari_wn VALUES('WN'),
PARTITION rezervari_dl VALUES('DL'),
PARTITION rezervari_ev VALUES('EV'),
PARTITION rezervari_ha VALUES('HA'),
PARTITION rezervari_mq VALUES('MQ'),
PARTITION rezervari_vx VALUES('VX'),
PARTITION rezervari_rest VALUES(DEFAULT))
AS
SELECT *
FROM rezervare;

ANALYZE TABLE rezervare_lista_operator COMPUTE STATISTICS;
```

Analizăm cererea ce conține numai condiția operatorului, nu și a datei.

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_rezervare_lista_1'
FOR
SELECT COUNT(*)
FROM rezervare_lista_operator
WHERE operator_id = 'HA';

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table', 'st_rezervare_lista_1', 'serial'));
```

PLAN_TABLE_OUTPUT									
Plan hash value: 206592571									
Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	
0	SELECT STATEMENT		1	2	43 (3)	00:00:01			
1	SORT AGGREGATE		1	2					
2	PARTITION LIST SINGLE		14082	28164	43 (3)	00:00:01	KEY	KEY	
3	TABLE ACCESS FULL	REZERVARE_LISTA_OPERATOR	14082	28164	43 (3)	00:00:01	12	12	

În acest caz, costul este 43. Adăugăm și condiția pentru dată:

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_rezervare_lista_2'
FOR
SELECT COUNT(*)
FROM rezervare_lista_operator
WHERE operator_id = 'HA'
AND data_plecare_id BETWEEN TO_DATE('2015-01-01 00:00:00',
'YYYY-MM-DD hh24:mi:ss')
AND TO_DATE('2015-01-31 00:00:00', 'YYYY-MM-DD
hh24:mi:ss');

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table', 'st_rezervare_lista_2', 'serial'));
```

PLAN_TABLE_OUTPUT									
Plan hash value: 206592571									

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	

0	SELECT STATEMENT		1	9	43 (3)	00:00:01			
1	SORT AGGREGATE		1	9					
2	PARTITION LIST SINGLE		6140	55260	43 (3)	00:00:01	KEY	KEY	
* 3	TABLE ACCESS FULL	REZERVARE_LISTIA_OPERATOR	6140	55260	43 (3)	00:00:01	12	12	

Costul final al variantei de partiționare de tip list este până acum cel mai optim, având valoarea 43.

3. Partiționarea *compusă*, de tip *range* după *data_plecare_id*, cu subpartiții de tip *list* pentru *operator_id*

În ultimul caz verificăm dacă o partiționare compusă ar aduce beneficii față de o partiționare simplă sau va aduce prea multă complexitate, astfel ridicând costurile.

Crearea tabelului:

```
CREATE TABLE rezervari_lunar_operator
PARTITION BY RANGE (data_plecare_id)
SUBPARTITION BY LIST (operator_id)
(PARTITION rezervari_jan2015 VALUES LESS THAN
(TO_TIMESTAMP('2015-02-01 00:00:00', 'YYYY-MM-DD hh24:mi:ss'))
(SUBPARTITION rezervari_jan2015_ua VALUES('UA'),
SUBPARTITION rezervari_jan2015_aa VALUES('AA'),
SUBPARTITION rezervari_jan2015_us VALUES('US'),
SUBPARTITION rezervari_jan2015_f9 VALUES('F9'),
SUBPARTITION rezervari_jan2015_b6 VALUES('B6'),
SUBPARTITION rezervari_jan2015_oo VALUES('OO'),
SUBPARTITION rezervari_jan2015_as VALUES('AS'),
SUBPARTITION rezervari_jan2015_nk VALUES('NK'),
SUBPARTITION rezervari_jan2015_wn VALUES('WN'),
SUBPARTITION rezervari_jan2015_dl VALUES('DL'),
SUBPARTITION rezervari_jan2015_ev VALUES('EV'),
SUBPARTITION rezervari_jan2015_ha VALUES('HA'),
SUBPARTITION rezervari_jan2015_mq VALUES('MQ'),
SUBPARTITION rezervari_jan2015_vx VALUES('VX'),
SUBPARTITION rezervari_jan2015_rest VALUES(DEFAULT)),
PARTITION rezervari_feb2015 VALUES LESS THAN
(TO_TIMESTAMP('2015-03-01 00:00:00', 'YYYY-MM-DD hh24:mi:ss'))
(SUBPARTITION rezervari_feb2015_ua VALUES('UA'),
SUBPARTITION rezervari_feb2015_aa VALUES('AA'),
SUBPARTITION rezervari_feb2015_us VALUES('US'));
```

```

SUBPARTITION rezervari_feb2015_f9 VALUES('F9'),
SUBPARTITION rezervari_feb2015_b6 VALUES('B6'),
SUBPARTITION rezervari_feb2015_oo VALUES('OO'),
SUBPARTITION rezervari_feb2015_as VALUES('AS'),
SUBPARTITION rezervari_feb2015_nk VALUES('NK'),
SUBPARTITION rezervari_feb2015_wn VALUES('WN'),
SUBPARTITION rezervari_feb2015_dl VALUES('DL'),
SUBPARTITION rezervari_feb2015_ev VALUES('EV'),
SUBPARTITION rezervari_feb2015_ha VALUES('HA'),
SUBPARTITION rezervari_feb2015_mq VALUES('MQ'),
SUBPARTITION rezervari_feb2015_vx VALUES('VX'),
SUBPARTITION rezervari_feb2015_rest VALUES(DEFAULT)),
PARTITION rezervari_mar2015 VALUES LESS THAN (MAXVALUE)
(SUBPARTITION rezervari_mar2015_ua VALUES('UA'),
SUBPARTITION rezervari_mar2015_aa VALUES('AA'),
SUBPARTITION rezervari_mar2015_us VALUES('US'),
SUBPARTITION rezervari_mar2015_f9 VALUES('F9'),
SUBPARTITION rezervari_mar2015_b6 VALUES('B6'),
SUBPARTITION rezervari_mar2015_oo VALUES('OO'),
SUBPARTITION rezervari_mar2015_as VALUES('AS'),
SUBPARTITION rezervari_mar2015_nk VALUES('NK'),
SUBPARTITION rezervari_mar2015_wn VALUES('WN'),
SUBPARTITION rezervari_mar2015_dl VALUES('DL'),
SUBPARTITION rezervari_mar2015_ev VALUES('EV'),
SUBPARTITION rezervari_mar2015_ha VALUES('HA'),
SUBPARTITION rezervari_mar2015_mq VALUES('MQ'),
SUBPARTITION rezervari_mar2015_vx VALUES('VX'),
SUBPARTITION rezervari_mar2015_rest VALUES(DEFAULT)))
AS SELECT * FROM rezervare;

```

```
ANALYZE TABLE rezervari_lunar_operator COMPUTE STATISTICS;
```

Analiza costului:

```

EXPLAIN PLAN
SET STATEMENT_ID = 'st_rezervare_lunar_operator_1'
FOR
SELECT COUNT(*)
FROM   rezervari_lunar_operator
WHERE  operator_id = 'HA'
AND    data_plecare_id BETWEEN TO_DATE('2015-01-01 00:00:00',
'YYYY-MM-DD hh24:mi:ss')

```

```
AND TO_DATE('2015-01-31 00:00:00', 'YYYY-MM-DD
hh24:mi:ss');

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_rezervare_lunar_operator
_1','serial'));
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT		1	9	23 (0)	00:00:01		
1	SORT AGGREGATE		1	9				
2	PARTITION RANGE SINGLE		6023	54207	23 (0)	00:00:01	1	1
3	PARTITION LIST SINGLE		6023	54207	23 (0)	00:00:01	KEY	KEY
4	TABLE ACCESS FULL	REZERVARI_LUNAR_OPERATOR	6023	54207	23 (0)	00:00:01	12	12

Calcularea costului în cazul în care accesăm direct subpartitia:

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_rezervare_lunar_operator_2'
FOR
SELECT COUNT(*)
FROM rezervari_lunar_operator SUBPARTITION
(rezervari_jan2015_ha);
-- cost

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_rezervare_lunar_operator_2','serial'));
```

Id	Operation	Name	Rows	Cost (%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT		1	23 (0)	00:00:01		
1	SORT AGGREGATE		1				
2	PARTITION COMBINED ITERATOR		6222	23 (0)	00:00:01	KEY	KEY
3	TABLE ACCESS FULL	REZERVARI_LUNAR_OPERATOR	6222	23 (0)	00:00:01	12	12

În ambele cazuri, pentru partiționarea compusă se obține cel mai mic cost, 23. Această variantă are costul cel redus, de 100 de ori mai mic față de cererea inițială neoptimizată. Însă pe de altă parte această variantă presupune calcularea a unui număr mult mai mare de subpartiții față de varianta a doua, cu partiționarea prin listă, care avea costul de doar 43.

13. Formularea unei cereri SQL complexe

Cerere SQL:

Să se afișeze suma totală care trebuie restituită în luna februarie de către fiecare companie aeriană, pentru zborurile la care au fost achiziționate bilete, dar care au fost anulate. De asemenea, sa se afișeze și numărul pasagerilor afectați, grupând după operator și clasă, numai operator și după totalul general.

Metode de optimizare:

- Se va folosi un tabel partiționat după luni
- Se va folosi un index pe coloana *anulat*
- Se vor utiliza *grouping sets* pentru grupare

Pentru început, vom scrie cererea fără optimizări:

```
SELECT operator_id, c.denumire, SUM(suma_totala) "Suma de
restituit", SUM(nr_pasageri) "Nr pasageri afectati"
FROM rezervare r
JOIN zbor z ON (r.zbor_id = z.zbor_id)
JOIN clasa_zbor c ON (c.clasa_zbor_id = r.clasa_zbor_id)
WHERE data_plecare_id BETWEEN TO_DATE('2015-02-01 00:00:00',
'YYYY-MM-DD hh24:mi:ss')
AND TO_DATE('2015-02-28 00:00:00', 'YYYY-MM-DD hh24:mi:ss')
AND z.anulat = 1
GROUP BY GROUPING SETS ((operator_id,
c.denumire),(operator_id),())
ORDER BY operator_id, c.denumire;
```

45	US	FIRST	2628874	3008
46	US	(null)	7651178	8400
47	VX	BUSINESS	232944	248
48	VX	ECONOMY	387146	403
49	VX	FIRST	293019	319
50	VX	(null)	913109	970
51	WN	BUSINESS	5205042	5730
52	WN	ECONOMY	5660051	6187
53	WN	FIRST	5859781	6178
54	WN	(null)	16724874	18095
55	(null)	(null)	94041800	102175

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				3968
SORT		GROUP BY ROLLUP		3968
HASH JOIN			46180	3965
Access Predicates				
C.CLASA_ZBOR_ID=R.CLASA_ZBOR_ID				
TABLE ACCESS	CLASA_ZBOR	FULL	3	3
HASH JOIN			46180	3962
Access Predicates				
R.ZBOR_ID=Z.ZBOR_ID				
TABLE ACCESS	ZBOR	FULL	40527	1186

Costul inițial fără optimizări este de 3968.

Introducem tabelul partiționat creat la subpunctul anterior:

```

EXPLAIN PLAN
SET STATEMENT_ID = 'st_cerere_complexa_1'
FOR
SELECT operator_id, c.denumire, SUM(suma_totala) "Suma de
restituit", SUM(nr_pasageri) "Nr pasageri afectati"
FROM rezervare_ord_data_plecare r
JOIN zbor z ON (r.zbor_id = z.zbor_id)
JOIN clasa_zbor c ON (c.clasa_zbor_id = r.clasa_zbor_id)
WHERE data_plecare_id BETWEEN TO_DATE('2015-02-01 00:00:00',
'YYYY-MM-DD hh24:mi:ss')
AND TO_DATE('2015-02-28 00:00:00', 'YYYY-MM-DD hh24:mi:ss')
AND z.anulat = 1
GROUP BY GROUPING SETS ((operator_id,
c.denumire),(operator_id),())
ORDER BY operator_id, c.denumire;

SELECT plan_table_output
FROM

```



```
table(dbms_xplan.display('plan_table','st_cerere_complexa_1','serial'));
```

PLAN_TABLE_OUTPUT									
Plan hash value: 2370609071									

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	

0	SELECT STATEMENT		30	1200	2312 (3)	00:00:01			
1	SORT GROUP BY ROLLUP		30	1200	2312 (3)	00:00:01			
* 2	HASH JOIN		61904	2418K	2308 (3)	00:00:01			
3	TABLE ACCESS FULL	CLASA_ZBOR	3	33	3 (0)	00:00:01			
* 4	HASH JOIN		61904	1753K	2304 (3)	00:00:01			
* 5	TABLE ACCESS FULL	ZBOR	40527	356K	1186 (3)	00:00:01			
6	PARTITION RANGE SINGLE		410K	8009K	1115 (2)	00:00:01	2	2	
* 7	TABLE ACCESS FULL	REZERVARE_ORD_DATA_PLECARE	410K	8009K	1115 (2)	00:00:01	2	2	

În acest caz, costul total scade, fiind de 2312.

Introducem indexul pe coloana anulat:

```
CREATE BITMAP INDEX idx_zbor_anulat ON zbor(anulat);
ANALYZE INDEX idx_zbor_anulat compute statistics;

EXPLAIN PLAN
SET STATEMENT_ID = 'st_cerere_complexa_2'
FOR
SELECT operator_id, c.denumire, SUM(suma_totala) "Suma de
restituit", SUM(nr_pasageri) "Nr pasageri afectati"
FROM rezervare_ord_data_plecare r
JOIN zbor z ON (r.zbor_id = z.zbor_id)
JOIN clasa_zbor c ON (c.clasa_zbor_id = r.clasa_zbor_id)
WHERE data_plecare_id BETWEEN TO_DATE('2015-02-01 00:00:00',
'YYYY-MM-DD hh24:mi:ss')
AND TO_DATE('2015-02-28 00:00:00', 'YYYY-MM-DD hh24:mi:ss')
AND z.anulat = 1
GROUP BY GROUPING SETS ((operator_id,
c.denumire),(operator_id),())
ORDER BY operator_id, c.denumire;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_cerere_complexa_2','seri
```

```
al')));
```

PLAN_TABLE_OUTPUT									
Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	
0	SELECT STATEMENT		30	1200	2214 (2)	00:00:01			
1	SORT GROUP BY ROLLUP		30	1200	2214 (2)	00:00:01			
* 2	HASH JOIN		61904	2418K	2210 (2)	00:00:01			
3	TABLE ACCESS FULL	CLASA_ZBOR	3	33	3 (0)	00:00:01			
* 4	HASH JOIN		61904	1753K	2206 (2)	00:00:01			
5	TABLE ACCESS BY INDEX ROWID BATCHED	ZBOR	40527	356K	1088 (1)	00:00:01			
6	BITMAP CONVERSION TO ROWIDS								
* 7	BITMAP INDEX SINGLE VALUE	IDX_ZBOR_ANULAT							
8	PARTITION RANGE SINGLE		410K	8009K	1115 (2)	00:00:01	2	2	
* 9	TABLE ACCESS FULL	REZERVARE_ORD_DATA_PLECARE	410K	8009K	1115 (2)	00:00:01	2	2	

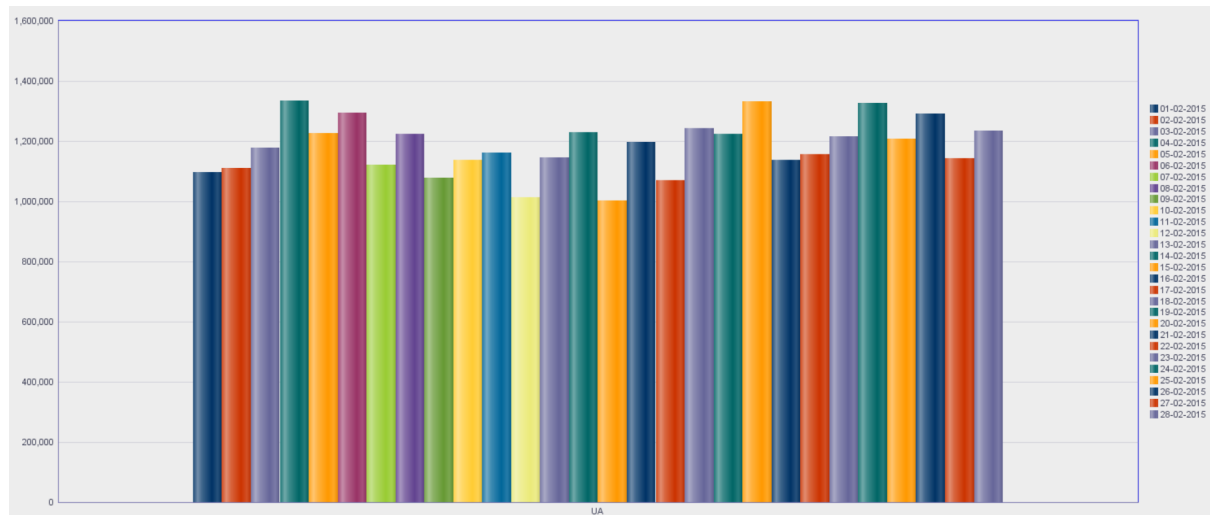
Noul cost este de 2214, scăzând la jumătate față de costul inițial.

14. Formularea a 5 cereri care vor fi concretizate în rapoarte

1. Să se calculeze, pentru operatorul de zbor *United Air Lines Inc.*, evoluția mediei centrată pe săptămâni, împreună cu suma totală pe săptămâni, data afișată reprezentând a 4-a zi din săptămâna analizată (ziua din mijloc)

```
SELECT operator_id, TO_CHAR(data_rezervare_id, 'DD-MM-YYYY')
data_rezervare,
SUM(suma_totala) incasari,
ROUND(AVG(SUM(suma_totala))
      OVER (PARTITION BY operator_id ORDER BY data_rezervare_id
            RANGE BETWEEN INTERVAL '3' DAY PRECEDING
            AND INTERVAL '3' DAY FOLLOWING), 2) AS
medie_centrata_7_zile
FROM rezervare r, timp t
WHERE r.data_rezervare_id = t.timp_id
AND an = 2015
AND operator_id = 'UA'
GROUP BY operator_id, data_rezervare_id;
```

	OPERATOR_ID	DATA_REZERVARE	INCASARI	MEDIE_CENTRATA_7_ZILE
1	UA	01-01-2015	1163047	1216533.75
2	UA	02-01-2015	1290808	1256072.8
3	UA	03-01-2015	1285130	1251800.83
4	UA	04-01-2015	1127150	1235323.71
5	UA	05-01-2015	1414229	1226932.29
6	UA	06-01-2015	1230441	1232658.43
7	UA	07-01-2015	1136461	1246135
8	UA	08-01-2015	1104307	1238629.29
9	UA	09-01-2015	1330891	1216617.43
10	UA	10-01-2015	1379466	1207775.29
11	UA	11-01-2015	1074610	1194962.86
12	UA	12-01-2015	1260146	1209184.86
13	UA	13-01-2015	1168546	1178708
14	UA	14-01-2015	1046774	1147969.57
15	UA	15-01-2015	1203861	1157785.43
16	UA	16-01-2015	1117553	1156978.86
17	UA	17-01-2015	1164297	1149161.14



2. Să se afișeze evoluția diferenței dintre numărul total de pasageri, de la o zi la alta, pentru toate zborurile din prima jumătate a lunii ianuarie, grupate după zi.

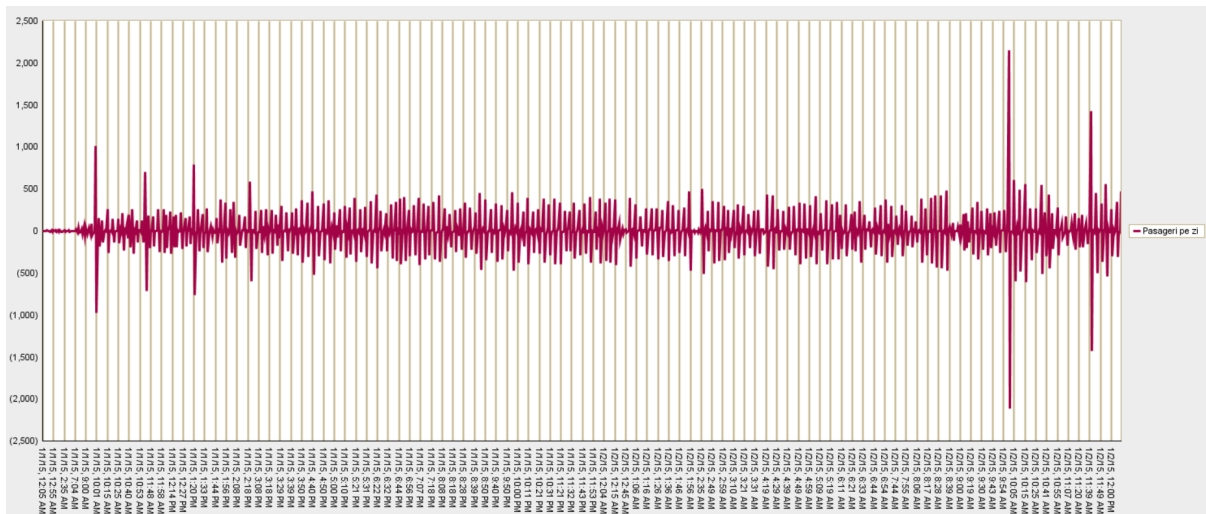
```
SELECT data_plecare_id, nr_pasageri_total,
COALESCE(nr_pasageri_total - LAG(nr_pasageri_total, 1) OVER (ORDER
BY data_plecare_id), 0) AS diferenta_pasageri_total
```

```

FROM (SELECT data_plecare_id,
            SUM(nr_pasageri) AS nr_pasageri_total
      FROM rezervare r, timp t
     WHERE r.data_plecare_id=t.timp_id
    AND zi_luna <= 15
    AND luna= 1
    AND an = 2015
    GROUP BY data_plecare_id);

```

DATA_PLECAR_ID	NR_PASAGERI_T...	DIFERENTA_PASAGERI_TOTAL
01-JAN-15 12.05.00.000000000 AM	14	0
01-JAN-15 12.10.00.000000000 AM	10	-4
01-JAN-15 12.20.00.000000000 AM	10	0
01-JAN-15 12.25.00.000000000 AM	8	-2
01-JAN-15 12.30.00.000000000 AM	16	8
01-JAN-15 12.35.00.000000000 AM	15	-1
01-JAN-15 12.40.00.000000000 AM	14	-1
01-JAN-15 12.45.00.000000000 AM	4	-10
01-JAN-15 12.48.00.000000000 AM	12	8
01-JAN-15 12.50.00.000000000 AM	5	-7
01-JAN-15 12.55.00.000000000 AM	12	7
01-JAN-15 01.40.00.000000000 AM	13	1
01-JAN-15 01.43.00.000000000 AM	19	6
01-JAN-15 01.45.00.000000000 AM	7	-12
01-JAN-15 01.55.00.000000000 AM	17	10
01-JAN-15 02.05.00.000000000 AM	12	-5
01-JAN-15 02.15.00.000000000 AM	27	15

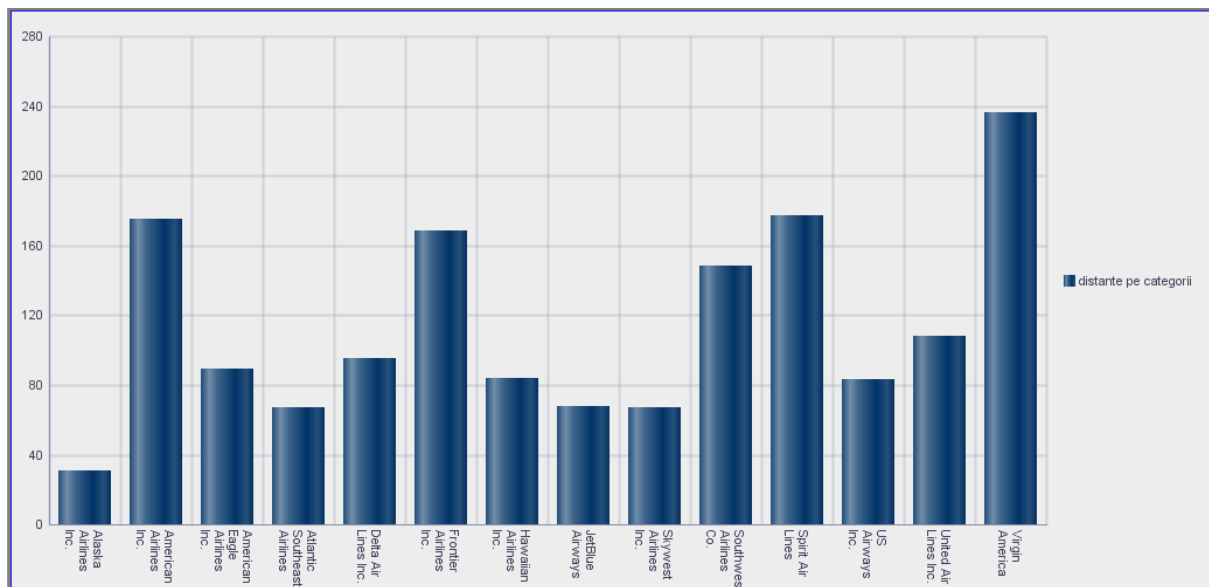


- Să se afișeze, pentru fiecare linie aeriană, distanța minimă și maximă a zborurilor acesteia, precum și prima și ultima dată la care s-au făcut rezervări pentru zborurile cu cea mai scurtă / cea mai lungă distanță.

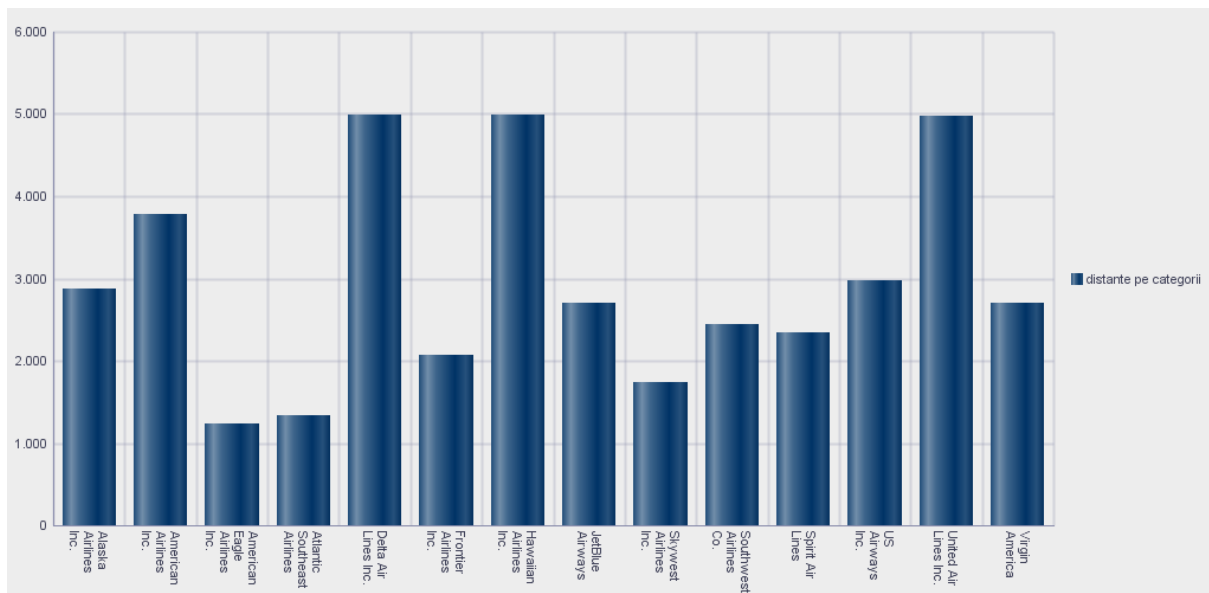
```
SELECT o.ume,
MIN(r.data_rezervare_id) KEEP (DENSE_RANK FIRST ORDER BY
z.distanța) "Data minimă cea mai scurtă distanța",
MIN(r.data_rezervare_id) KEEP (DENSE_RANK LAST ORDER BY
z.distanța) "Data minimă cea mai lungă distanța",
MIN(z.distanța) "DISTANȚA MINIMĂ",
MAX(r.data_rezervare_id) KEEP (DENSE_RANK FIRST ORDER BY
z.distanța) "Data maximă cea mai scurtă distanța",
MAX(r.data_rezervare_id) KEEP (DENSE_RANK LAST ORDER BY
z.distanța) "Data maximă cea mai lungă distanța",
MAX(z.distanța) "DISTANȚA MAXIMĂ"
FROM zbor z
JOIN rezervare r on z.zbor_id = r.zbor_id
JOIN operator_zbor o on o.operator_id = r.operator_id
GROUP BY o.ume;
```

NUME	Data minimă cea mai scurtă distanța	Data minimă cea mai lungă distanța	DISTANȚA MINIMĂ	Data maximă cea mai scurtă distanța	Data maximă cea mai lungă distanța	DISTANȚA MAXIMĂ
1 Alaska Airlines Inc.	03-01-2015 00:00:00,000000000	01-01-2015 00:00:00,000000000	31	25-12-2015 00:00:00,000000000	30-12-2015 00:00:00,000000000	2874
2 American Airlines Inc.	01-01-2015 00:00:00,000000000	01-01-2015 00:00:00,000000000	175	30-12-2015 00:00:00,000000000	30-12-2015 00:00:00,000000000	3784
3 American Eagle Airlines Inc.	01-01-2015 00:00:00,000000000	03-01-2015 00:00:00,000000000	89	29-12-2015 00:00:00,000000000	26-12-2015 00:00:00,000000000	1236
4 Atlantic Southeast Airlines	13-02-2015 00:00:00,000000000	04-01-2015 00:00:00,000000000	67	13-12-2015 00:00:00,000000000	30-12-2015 00:00:00,000000000	1330
5 Delta Air Lines Inc.	12-01-2015 00:00:00,000000000	05-01-2015 00:00:00,000000000	95	10-12-2015 00:00:00,000000000	23-11-2015 00:00:00,000000000	4983
6 Frontier Airlines Inc.	22-03-2015 00:00:00,000000000	04-01-2015 00:00:00,000000000	168	24-09-2015 00:00:00,000000000	23-12-2015 00:00:00,000000000	2065
7 Hawaiian Airlines Inc.	01-01-2015 00:00:00,000000000	02-01-2015 00:00:00,000000000	84	29-12-2015 00:00:00,000000000	30-12-2015 00:00:00,000000000	4983
8 JetBlue Airways	01-01-2015 00:00:00,000000000	01-01-2015 00:00:00,000000000	68	30-12-2015 00:00:00,000000000	30-12-2015 00:00:00,000000000	2704
9 Skywest Airlines Inc.	01-01-2015 00:00:00,000000000	04-01-2015 00:00:00,000000000	67	30-12-2015 00:00:00,000000000	30-12-2015 00:00:00,000000000	1735
10 Southwest Airlines Co.	01-01-2015 00:00:00,000000000	02-01-2015 00:00:00,000000000	148	30-12-2015 00:00:00,000000000	29-12-2015 00:00:00,000000000	2447
11 Spirit Air Lines	02-01-2015 00:00:00,000000000	03-01-2015 00:00:00,000000000	177	30-12-2015 00:00:00,000000000	20-12-2015 00:00:00,000000000	2342
12 US Airways Inc.	06-06-2015 00:00:00,000000000	01-01-2015 00:00:00,000000000	83	05-11-2015 00:00:00,000000000	26-12-2015 00:00:00,000000000	2979
13 United Air Lines Inc.	02-01-2015 00:00:00,000000000	02-01-2015 00:00:00,000000000	108	12-11-2015 00:00:00,000000000	23-12-2015 00:00:00,000000000	4962
14 Virgin America	01-01-2015 00:00:00,000000000	03-01-2015 00:00:00,000000000	236	30-12-2015 00:00:00,000000000	30-12-2015 00:00:00,000000000	2704

Generăm un raport în care este prezentată cea mai scurtă distanță pentru fiecare linia aeriană:



Respectiv un alt raport în care este afișată distanța maximă:

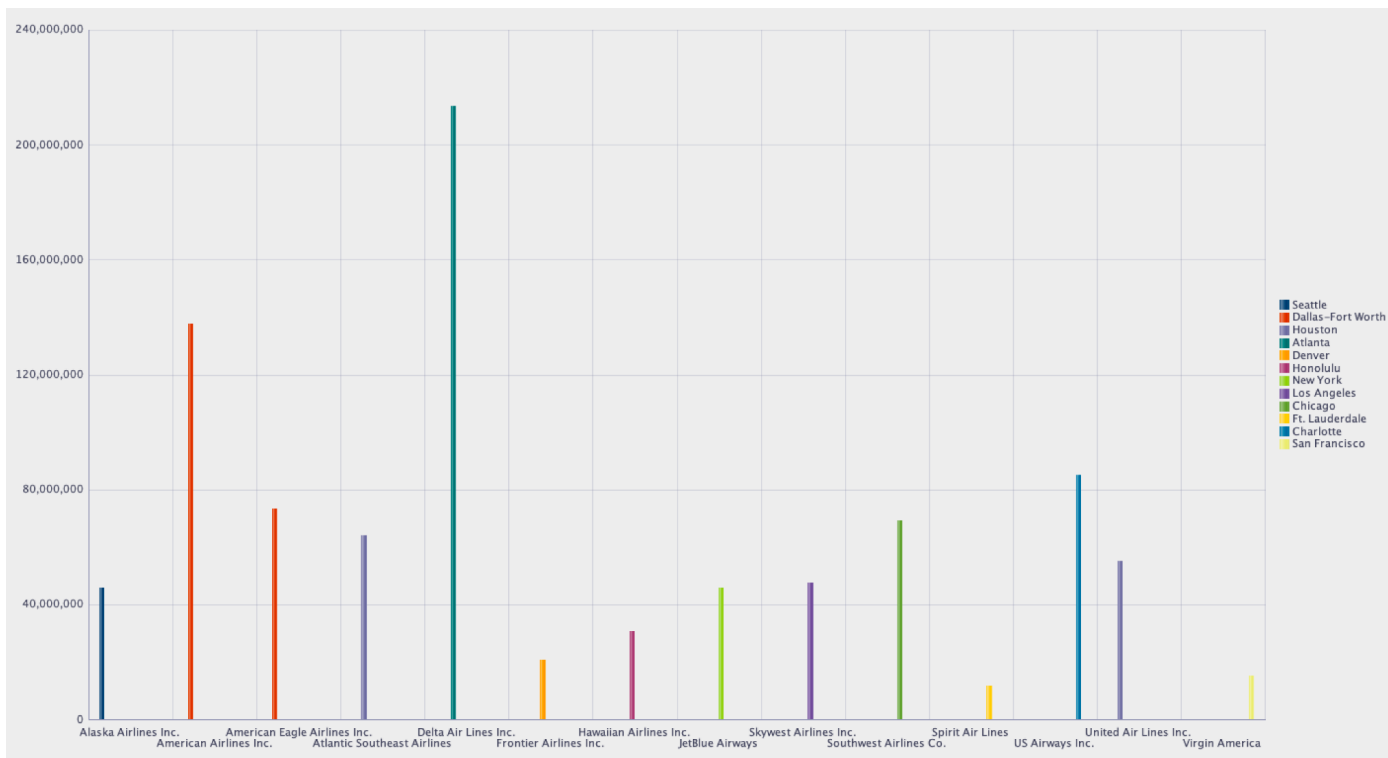


4. Pentru fiecare operator, să se obțină destinația care a înregistrat vânzări maxime de bilete de avion.

```
SELECT nume AS operator, oras, valoare
FROM (SELECT nume, oras,
SUM(suma_totala) AS valoare,
MAX(SUM(suma_totala))
OVER (PARTITION BY nume) AS max_val
FROM rezervare r, operator_zbor o, destinatie d
WHERE r.locatie_sosire_id=d.destinatie_id
AND r.operator_id=o.operator_id)
```

```
GROUP BY nume, oras)
WHERE valoare= max_val;
```

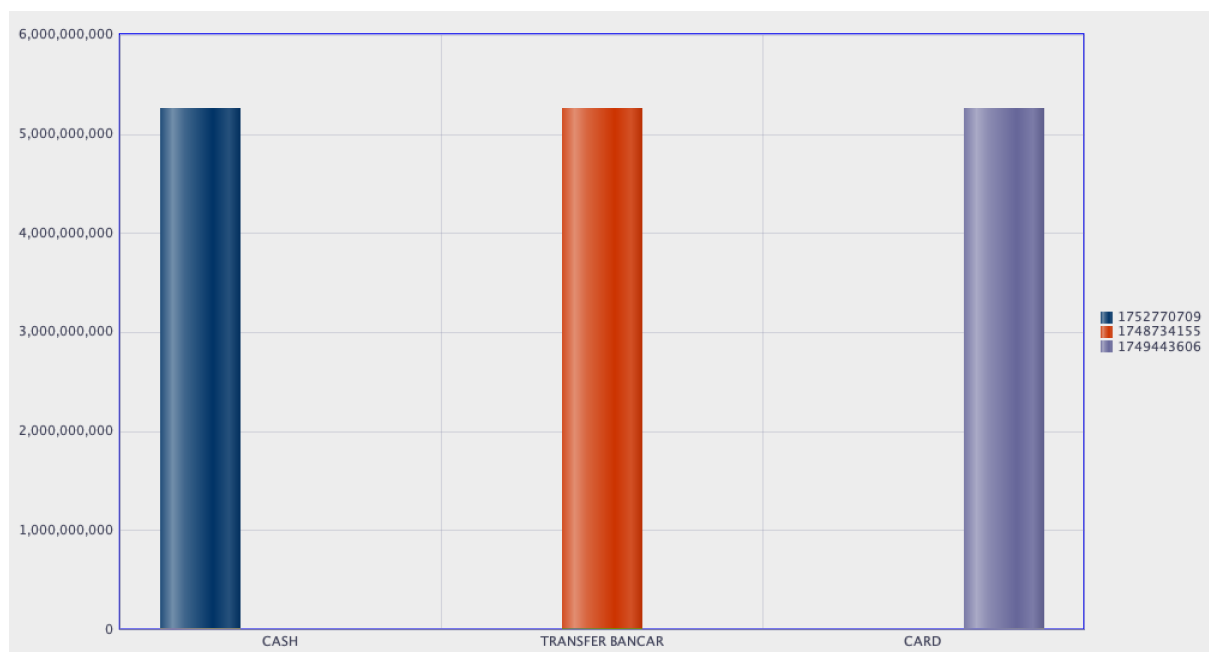
	✈ OPERATOR	✈ ORAS	✈ VALOARE
1	Alaska Airlines Inc.	Seattle	45710681
2	American Airlines Inc.	Dallas-Fort Worth	137586867
3	American Eagle Airlines Inc.	Dallas-Fort Worth	73281007
4	Atlantic Southeast Airlines	Houston	64018971
5	Delta Air Lines Inc.	Atlanta	213137810
6	Frontier Airlines Inc.	Denver	20499320
7	Hawaiian Airlines Inc.	Honolulu	30587646
8	JetBlue Airways	New York	45707578
9	Skywest Airlines Inc.	Los Angeles	47466987
10	Southwest Airlines Co.	Chicago	69251196
11	Spirit Air Lines	Ft. Lauderdale	11710798
12	US Airways Inc.	Charlotte	85076266
13	United Air Lines Inc.	Houston	55073189
14	Virgin America	San Francisco	15126187



5. Pentru fiecare tip de plată obțineți valoarea procentuală a vânzărilor de bilete de avion realizate raportată la vânzările totale.

```
SELECT denumire,
       SUM(suma_totala)
       VANZARI,
       SUM(SUM(suma_totala))
       OVER ()
       AS TOTAL_VANZ,
       round(RATIO_TO_REPORT(
       SUM(suma_totala))
       OVER (), 6)
       AS RATIO_REP
FROM metoda_plata m, rezervare r
WHERE r.metoda_plata_id=m.metoda_plata_id
GROUP BY denumire;
```

	DENUMIRE	VANZARI	TOTAL_VANZ	RATIO_REP
1	CASH	1752770709	5250948470	0.333801
2	TRANSFER BANCAR	1748734155	5250948470	0.333032
3	CARD	1749443606	5250948470	0.333167



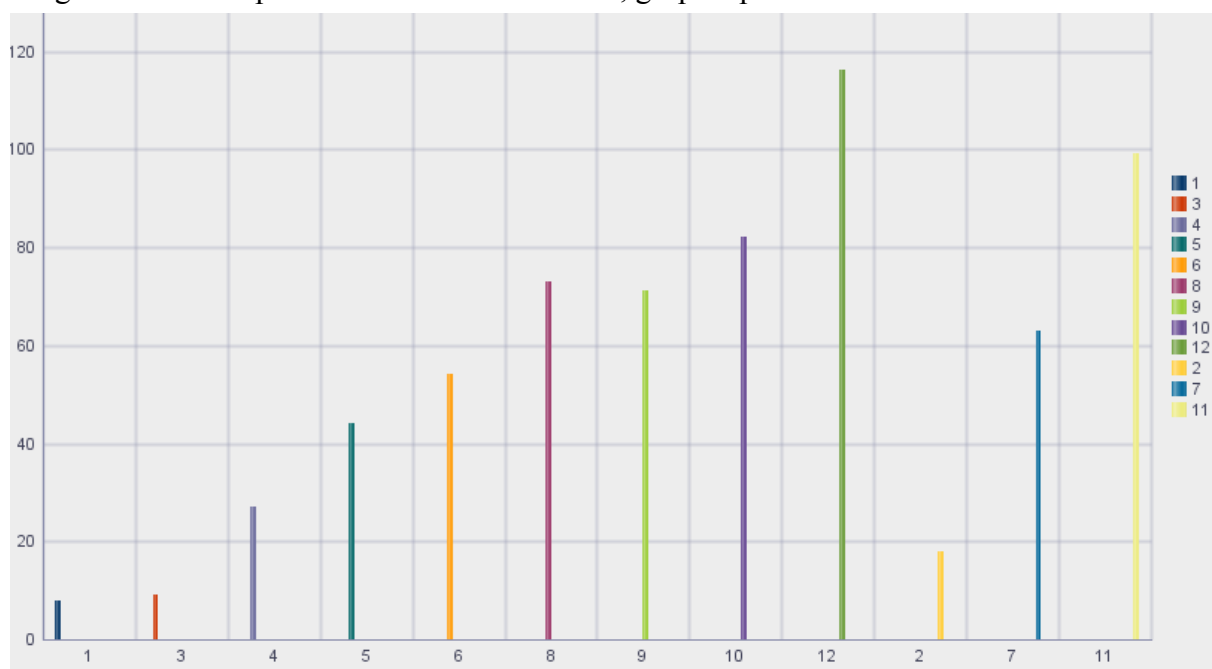
6. Pentru fiecare client dintr-o mulțime dată, să se afișeze suma cheltuită pe rezervările sale, grupate pe luni, respectiv pentru câți pasageri a rezervat bilete până în

momentul respectiv. Aceste rezervări trebuie să conțină minim 8 pasageri, dintre care cel puțin 6 femei.

```
SELECT client_id, luna, SUM(r.suma_totala) SUMA,
SUM(r.nr_pasageri) "Nr pasageri in luna X",
SUM(SUM(r.nr_pasageri)) OVER(PARTITION BY client_id ORDER BY
client_id, luna ROWS UNBOUNDED PRECEDING) "Nr pasageri pana in
luna X"
FROM rezervare r, timp t
WHERE r.data_rezervare_id = t.timp_id AND nr_pasageri > 7 AND
nr_pasageri_femei > 5 AND client_id IN (1,2,3)
GROUP BY client_id, luna;
```

	CLIENT_ID	LUNA	SUMA	Nr pasageri in luna X	Nr pasageri pana in luna X
1	1	1	4270	9	9
2	1	3	5968	8	17
3	1	4	5947	9	26
4	1	5	5688	10	36
5	1	6	17046	18	54
6	1	8	6613	9	63
7	1	9	199	8	71
8	1	10	11307	17	88
9	1	12	21734	28	116
10	2	3	1989	9	9

Se generează un raport cu numărul de rezervări, grupate pe lună

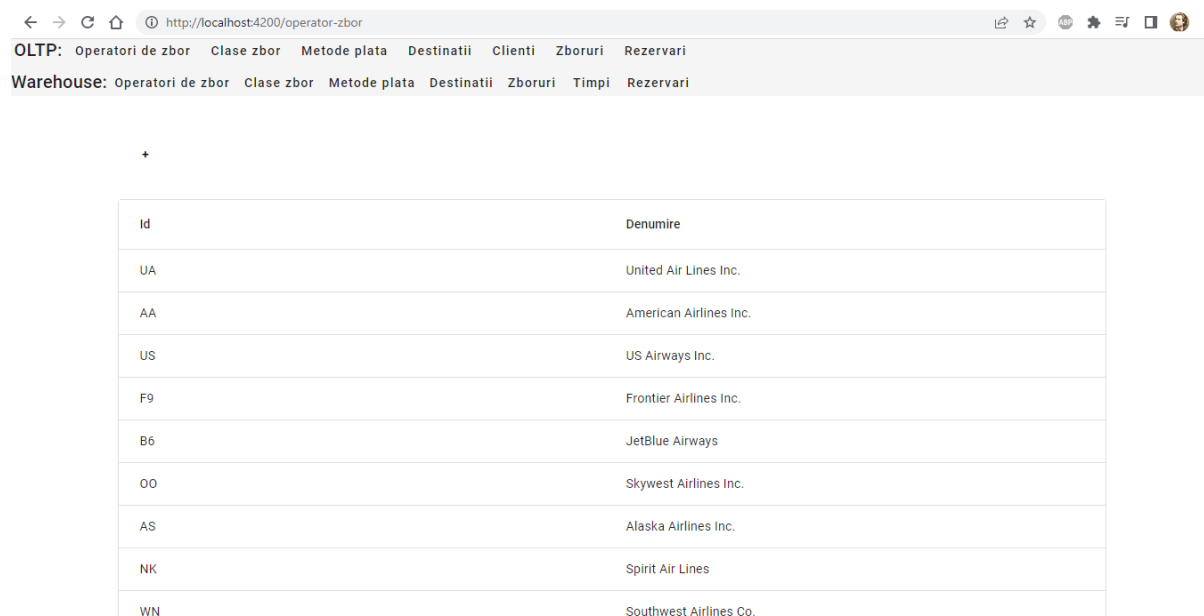


15. Modulul aplicație

- Introducerea și gestionarea informațiilor la nivelul bazei de date OLTP

Pentru modulul “Aplicație”, a fost realizată o aplicație Web folosind Typescript/Angular pentru partea de frontend, respectiv Java/Spring Boot pentru backend. Prin intermediul aplicației, utilizatorul are dreptul să vizualizeze datele din ambele baze de date: OLTP și Warehouse, respectiv să insereze date în baza de date OLTP, date care vor fi adăugate automat în baza de date tip depozit.

Pagina principală a aplicației conține un widget de tip *navbar* prin care utilizatorul poate accesa datele în funcție de baza de date aleasă:



The screenshot shows a web browser window with the URL `http://localhost:4200/operator-zbor`. The page has a navbar with two sections: "OLTP" and "Warehouse". The "OLTP" section is active, showing a list of menu items: "Operatori de zbor", "Clase zbor", "Metode plata", "Destinatii", "Clienti", "Zboruri", and "Rezervari". The "Warehouse" section shows a similar list but with "Timpi" instead of "Clienti". Below the navbar, there is a table with two columns: "Id" and "Denumire". The table contains 10 rows of data representing different airlines.

Id	Denumire
UA	United Air Lines Inc.
AA	American Airlines Inc.
US	US Airways Inc.
F9	Frontier Airlines Inc.
B6	JetBlue Airways
OO	Skywest Airlines Inc.
AS	Alaska Airlines Inc.
NK	Spirit Air Lines
WN	Southwest Airlines Co.

Acesta poate naviga și vizualiza date despre oricare din categorii: *Operatori de zbor*, *Clase de zbor*, *Metode de plată*, *Destinații*, *Clienți* (doar pentru baza de date OLTP), *Zboruri*, *Rezervări* sau *Timp* (doar pentru baza de date depozit).

Exemplificăm o parte din aceste date:

1) Operatori de Zbor

Id	Denumire
UA	United Air Lines Inc.
AA	American Airlines Inc.
US	US Airways Inc.
F9	Frontier Airlines Inc.
B6	JetBlue Airways
OO	Skywest Airlines Inc.
AS	Alaska Airlines Inc.
NK	Spirit Air Lines
WN	Southwest Airlines Co.
DL	Delta Air Lines Inc.

2) Clase de Zbor

Id	Denumire
1	FIRST
2	BUSINESS
3	ECONOMY

3) Metode de plată

Id	Denumire
1	CASH
2	CARD
3	TRANSFER BANCAR

4) Destinații

Id	Oras	Stat
ABE	Allentown	PA
ABI	Abilene	TX
ABQ	Albuquerque	NM
ABR	Aberdeen	SD
ABY	Albany	GA
ACK	Nantucket	MA
ACT	Waco	TX
ACV	Arcata/Eureka	CA
ACY	Atlantic City	NJ

5) Clienți

Id	Nume	Prenume	Email	Numar telefon
1	Mcguire	Sara	tsharp@example.net	(971)643-6089x9160
2	Hebert	Alisha	vincentgarrett@example.net	+1-114-355-1841x78347
3	Sheppard	Gwendolyn	mercadojonathan@example.coi	9017807728
4	Mccann	Kristine	lindsay55@example.com	+1-607-333-9911x59088
5	Pittman	Bobby	blevinsmorgan@example.com	3739847538
6	Ramsey	Calvin	loretta85@example.com	001-314-829-5014x1792
7	Allison	Collin	yvaughn@example.net	(314)591-7413
8	Branch	Nicholas	greerjimmy@example.net	-7199
9	Robinson	Emma	charleshiggins@example.org	166-234-6882x7457
10	Cordova	Pedro	leslie08@example.com	(389)824-3204x8287
11	Aguilar	Jean	raymond24@example.org	(285)029-1604x5466

6) Zboruri

Id	Operator Id	Aeronava Id	Durata	Distanta	Total locuri	Anulat	Data plecare	Data sosire	Locatie plecare	Locatie sosire
1048576	MQ	MQ	65 (minute)	200	100	<input type="checkbox"/>	11-28-1998 15:50	11-28-1998 16:55	ABE	ABI
1048575	MQ	N539MQ	0 (minute)	196	90	<input checked="" type="checkbox"/>	03-10-2015 16:53	03-10-2015 16:53	CID	ORD
1048574	EV	N14158	127 (minute)	837	90	<input type="checkbox"/>	03-10-2015 16:53	03-10-2015 19:00	MSY	ORD
1048573	UA	N76508	220 (minute)	1723	80	<input type="checkbox"/>	03-10-2015 16:53	03-10-2015 20:33	SAN	ORD
1048572	UA	N79279	219 (minute)	1416	50	<input type="checkbox"/>	03-10-2015 16:53	03-10-2015 20:32	LGA	IAH
1048571	EV	N11191	64 (minute)	416	100	<input type="checkbox"/>	03-10-2015 16:53	03-10-2015 17:57	RDU	EWR
1048570	DL	N555NW	183 (minute)	1587	90	<input type="checkbox"/>	03-10-2015 16:53	03-10-2015 19:56	PHX	ATL
1048569	B6	N633JB	199 (minute)	1693	100	<input type="checkbox"/>	03-10-2015 16:53	03-10-2015 20:12	BOS	STT
1048568	AS	N558AS	37	204	100	<input type="checkbox"/>	03-10-2015	03-10-2015	SCC	BBW

7) Rezervări

Id	Pasageri	Data rezervare	Suma	Client id	Zbor id	Clasa zbor	Metoda plata id
1040031	5: 3F / 2B	02-01-2023 1:28	125	2	1048570	2	2
1040030	3: 1F / 2B	02-01-2023 1:25	121	1	1048573	1	1
1040027	2: 1F / 1B	02-01-2023 0:52	121	1	1048572	1	1
1040026	2: 1F / 1B	02-01-2023 0:46	23	1	1048573	1	1
1040025	3: 1F / 2B	02-01-2023 0:44	30	1	1048574	1	1
1040001	22: 2F / 2B	01-31-2023 10:11	100	1	1048575	1	1
1040000	9: 6F / 3B	01-20-2015 0:00	3269	9544	988291	1	3
1039999	1: 0F / 1B	12-18-2015 0:00	6495	1237	276588	1	3
1039998	10: 2F / 8B	10-25-2015 0:00	8740	474	629344	3	1
1039997	1: 1F / 0B	08-13-2015 0:00	890	6271	597392	1	3
1039996	8: 6F / 2B	07-14-2015 0:00	2404	6454	1032730	2	3

- Vizualizarea datelor introduse. Propagarea operațiilor în baza de date depozit

Pentru fiecare entitate, se pot insera date prin apăsarea butonului + din colțul din partea de sus. Astfel, în funcție de câmpurile fiecărei entități, vor apărea diferite formulare cu datele corespunzătoare.

Exemplificăm pentru câteva entități:

1) Clase zbor

După confirmare, se observă că a fost introdusă o nouă înregistrare în baza de date OLTP:

Id	Denumire
1	FIRST
2	BUSINESS
3	ECONOMY
7	BASIC

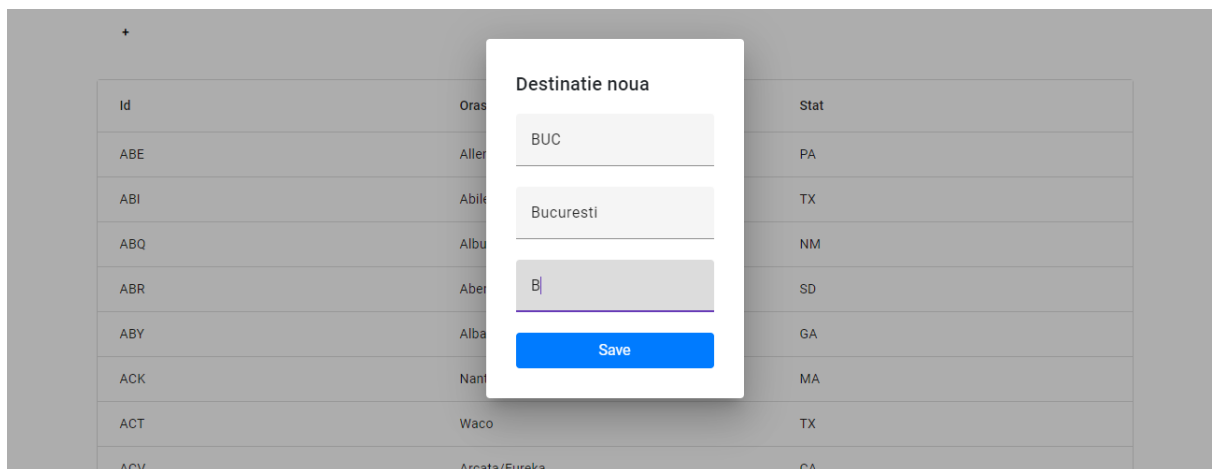
De asemenea, se verifică și datele din baza de date de tip depozit, prin accesarea tabului corespunzător din bara de navigare:

Warehouse: Operatori de zbor Clase zbor Metode plata Destinatii Zboruri Timpi Rezervari

Se poate observa că datele au fost propagate:

Id	Denumire
1	FIRST
2	BUSINESS
3	ECONOMY
7	BASIC

2) Destinații



După confirmare, se observă că a fost introdusă o nouă înregistrare:

BUC	Bucuresti	B
-----	-----------	---

De asemenea, se verifică și datele din baza de date de tip depozit:

BUC	Bucuresti	B
-----	-----------	---

3) Zboruri

+

Zbor nou

MQ

680

Data plecare*
05.02.2023 15:00

N76508

120

05.02.2023 19:20

BUC

TOL

Save

Id	Operator Id	Aeronava Id	Durata	Distanța	Total locuri	Anulat	Data plecare	Data sosire	Locatie plecare	Locatie sosire
1048576	MQ	N11191	260 (minute)	416	100	<input type="checkbox"/>	2015-03-10 16:53	2015-03-10 17:57	ABE	ABI
1048575	MQ	N555NW	183 (minute)	1587	90	<input type="checkbox"/>	2015-03-10 16:53	2015-03-10 19:56	CID	ORD
1048574	EV								MSY	ORD
1048573	UA								SAN	ORD
1048572	UA								LGA	IAH
1048571	EV								RDU	EWR
1048570	DL								PHX	ATL

După confirmare, se observă că a fost introdusă o nouă înregistrare:

Id	Operator Id	Aeronava Id	Durata	Distanța	Total locuri	Anulat	Data plecare	Data sosire	Locatie plecare	Locatie sosire
1048596	MQ	N76508	260 (minute)	680	120	<input type="checkbox"/>	02-05-2023 15:00	02-05-2023 19:20	BUC	TOL

Se verifică din nou că datele au fost introduse în warehouse:

Id	Aeronava Id	Durata	Distanța	Total locuri	Anulat
1048596	N76508	260 (minute)	680	120	<input type="checkbox"/>

După cum se vede și în imagine, cele două tabele au câmpuri diferite, în funcție de baza de date din care provin: OLTP sau Warehouse.

4) Rezervări

+

Rezervare noua

3

2

4

1048596

1580

7

3

Save

Id	Pasageri	Clasa zbor	Metoda plata id
1040031	5: 3F / 2B	2	2
1040030	3: 1F / 2B	1	1
1040027	2: 1F / 1B	1	1
1040026	2: 1F / 1B	1	1
1040025	3: 1F / 2B	1	1
1040001	22: 2F / 2B	1	1
1040000	9: 6F / 3B	1	3
1039999	1: 0F / 1B	1	3

După confirmare, se observă că a fost introdusă o nouă înregistrare:

Id	Pasageri	Data rezervare	Suma	Client id	Zbor id	Clasa zbor	Metoda plata id
1040032	7: 3F / 4B	02-01-2023 2:46	1580	2	1048596	7	3

Se verifică propagarea în baza de date depozit pentru aceeași tabelă:

Pasageri	Client id	Data rezervare	Data plecare	Data sosire	Destinatii	Operator	Zbor id	Clasa zbor id	Metoda plata id
7: 3F / 4B	2	02-01-2023 2:46	02-05-2023 15:00	02-05-2023 19:20	BUC -> TOL	MQ	1048596	7	3

De asemenea, tabela *Timp* specifică doar bazei de date depozit, a fost și ea actualizată cu noile valori pentru datele calendaristice introduse:

Id	Data	Zi an/saptamana an	Ora	Weekend
2023-02-05T17:20:00.000+00:00	2023/2/5	36/7	19:20	<input type="checkbox"/>
2023-02-05T13:00:00.000+00:00	2023/2/5	36/7	15:0	<input type="checkbox"/>