Data Warehouse & Business Intelligence

Modulul analiză

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 zborurile 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, distanta, 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:

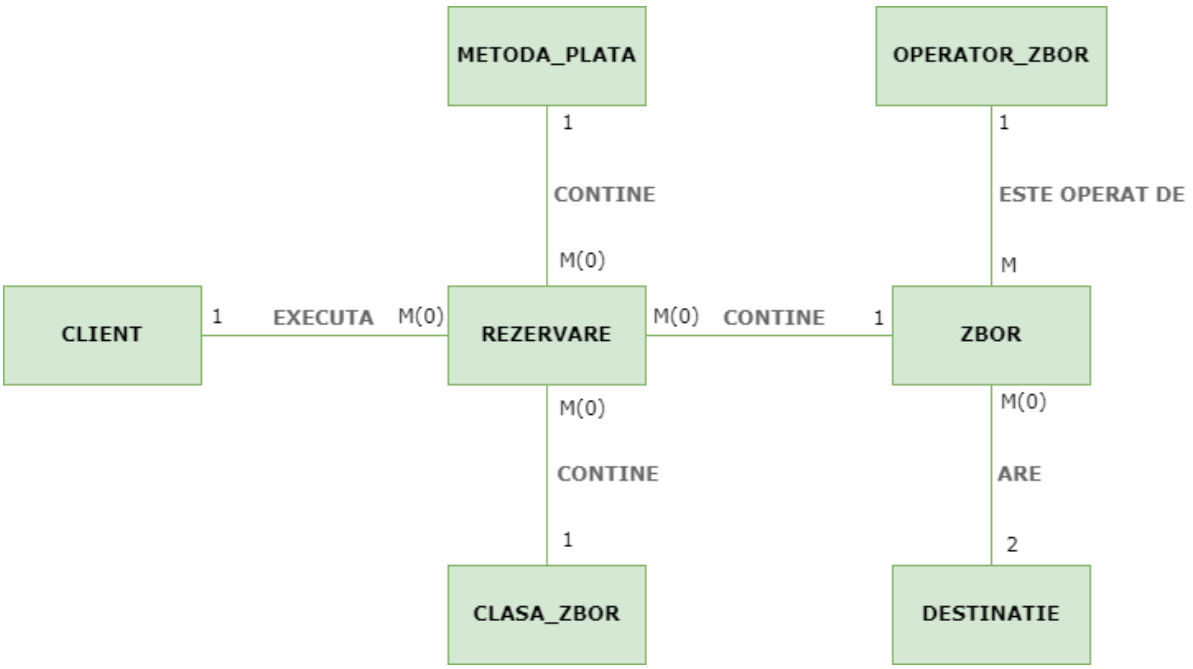
- Adaugă coloana zbor_id, creată incremental, pentru a o folosi în continuare cu ușurință
- Generează coloana total_locuri, cu valori între 50 și 100
- Modifică coloanele data_plecare și data_sosire de tip timestamp, pentru a conține și ora

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.

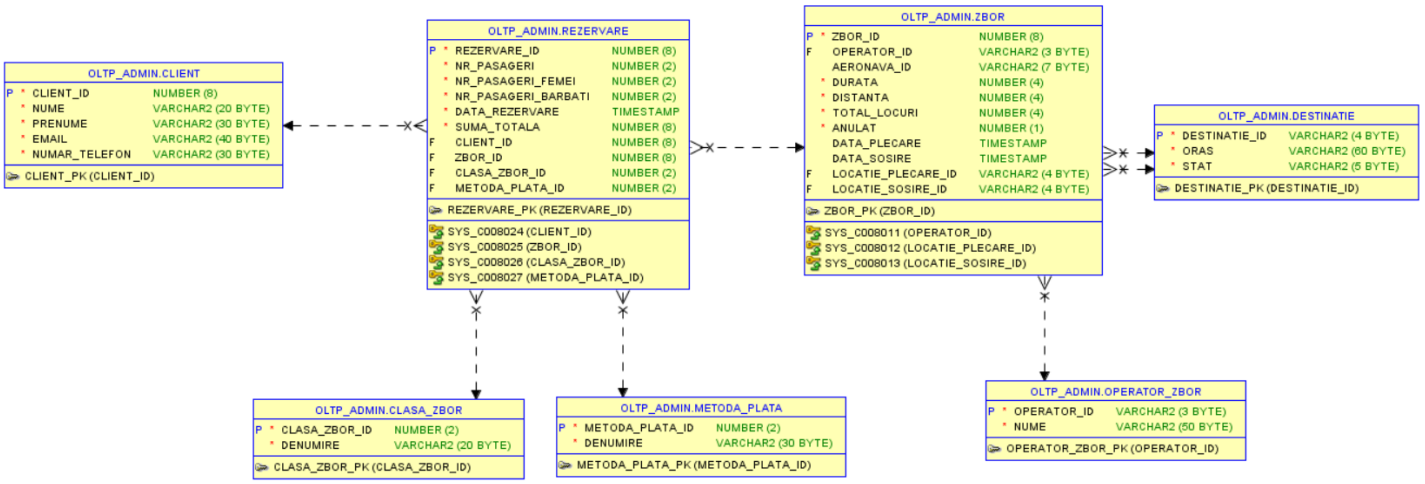
Rezervările clienților au fost generate în totalitate folosind un script de python și conțin: id-ul, numărul total de pasageri conținut de rezervare, numărul de pasageri care sunt femei, respectiv bărbați, data rezervării, suma totală platită de client, id-ul clientului, id-ul zborului, clasa la care 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

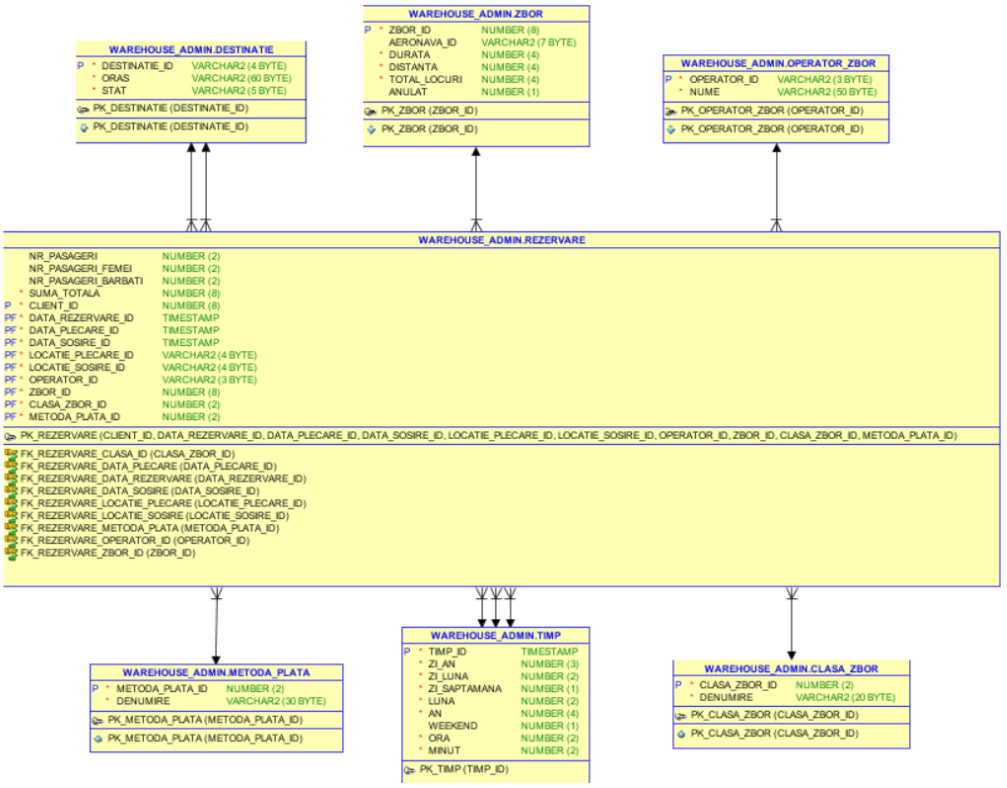
○ Diagrama entitate - relație a bazei de date OLTP



- Diagrama conceptuală a bazei de date OLTP



3. Diagrama stea a bazei de date depozit



4. 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_plecare_id*, *locatie_sosire_id*, *data_plecare_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:*

- *Destinatie* - 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 *minutele,* 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.*

5. 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 norely*. 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 tabelei 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
    - *Rely disable novalidate* va fi o variantă foarte puțin costisitoare, având un consum mic de resurse, fiind potrivită în acest caz
- Constrângere de tip *rely 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_plecare_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 *rely 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:
    - Implicit, 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*

6. Identificarea indecșilor

Din motive de îmbunătățire a 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 tip bitmap întrucât există un număr limitat de valori posibile pentru tipurile de plată.

7. 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.

8. 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_plecare_id,* cu subpartiții de tip *list* pentru *operator_id*

9. 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 clasa, 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

10. Formularea a cel puțin 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).

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.
3. Să se afișeze, pentru fiecare linie aeriană, distanța medie 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.
4. Pentru fiecare operator, să se obțină destinația care a înregistrat vânzări maxime de bilete de avion.
5. Pentru fiecare tip de plată obțineți valoarea procentuală a vânzărilor de bilete de avion realizate raportată la vânzările totale.
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.