

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής



Εργασία Μαθήματος «Βάσεις Δεδομένων(4^ο εξ.)»

Όνομα φοιτητή – Αρ. Μητρώου	Μίμογλου Γιώργος – Π17073
	Παπαθεοχάρους Ιωάννης – Π17002
	Ντόστης Χρήστος – Π16189
Ημερομηνία παράδοσης	10/6/2019



Γενική Περιγραφή της λύσης

Στην εργασία αυτή υλοποιήθηκε μια βάση δεδομένων μιας εταιρίας πώλησης εισιτηρίων. Για τις ανάγκες λοιπόν της άσκησης χρησιμοποιήσαμε δεδομένα από την ιστοσελίδα www.mackaroo.com , μικρά προγράμματα python για την επεξεργασία τους καθώς και την πλατφόρμα pgAdmin4. Τέλος, δημιουργήθηκε ένα java project με περιορισμένες λειτουργίες πάνω στην βάση δεδομένων.

Ερώτημα 1 Σχεσιακή Βάση Δεδομένων

Δημιουργήθηκαν οι παρακάτω σχέσης:

- CUSTOMERS (id, name, afm, phone, email, credit card, age).
 - PK=(id).
 - FD1: id→name, afm, phone, email, credit card, age.

Άρα είναι BCNF.

- TICKETS (id, title, category, start_date, end_date, price, available, location, provider).
 - PK=(id).
 - FK={provider}
 - FD1: id→title, category, start_date, end_date, price, available, location, provider.

Άρα από μεταβατική ιδιότητα είναι BCNF.

- PROVIDERS (name, afm).
 - PK=(name).
 - FD1: name→afm.

Άρα είναι BCNF.

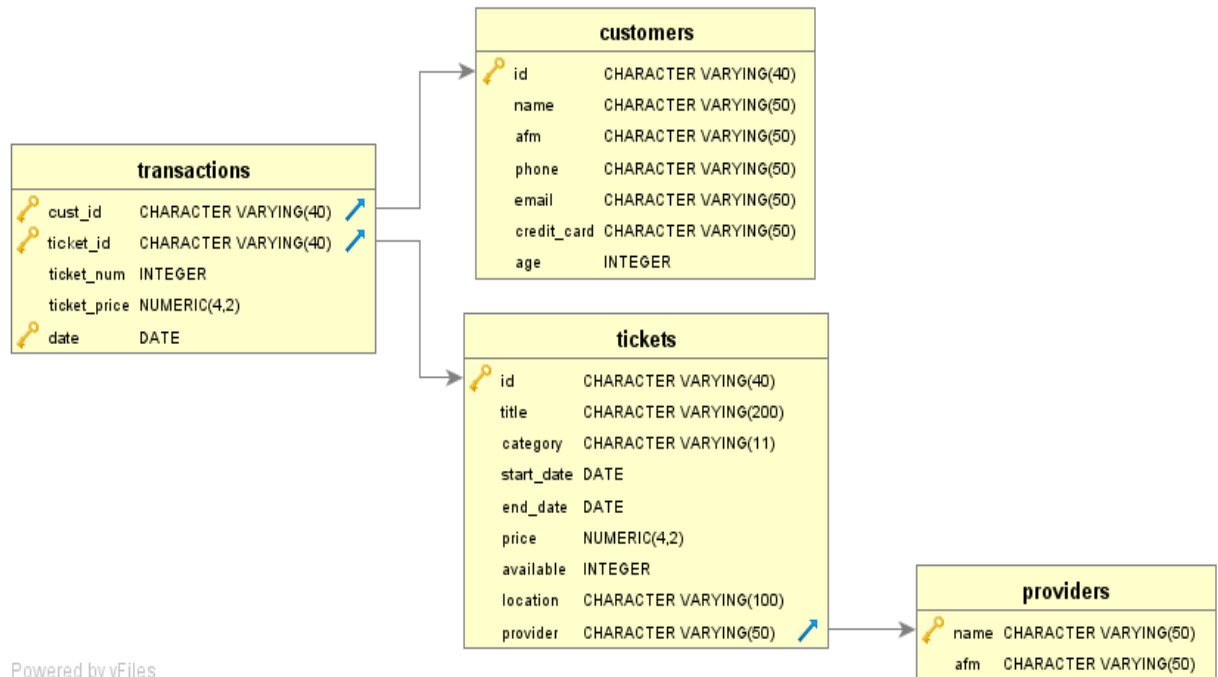
- TRANSACTIONS (cust_id, ticket_id, ticket_num, ticket_price, date).
 - PK=(cust_id, ticket_id, date).
 - FK={cust_id} , {ticket_id}.
 - FD1: {cust_id, ticket_id, date}→ticket_num, ticket_price.

Άρα είναι BCNF.



Δεχόμαστε ότι (customers)afm, phone, email, credit card, (providers)name, (providers)afm είναι μοναδικά και δεν μπορεί να υπάρξει το ίδιο θέμα πάνω από 1 φορά την ημέρα.

Παρακάτω φαίνεται το σχεσιακό σχήμα της βάσης δεδομένων:



\Παρακάτω παρατίθενται τα SQL queries για την δημιουργία των πινάκων όπως υπάρχουν στον φάκελο init.

```
create table Customers (  
    id VARCHAR(40) PRIMARY KEY,  
    name VARCHAR(50) NOT NULL,  
    AFM VARCHAR(50),  
    phone VARCHAR(50),  
    email VARCHAR(50) NOT NULL,  
    credit_card VARCHAR(50),  
    age INT  
);
```



```
create table Providers (  
    name VARCHAR(50) PRIMARY KEY,  
    AFM VARCHAR(50)  
);
```

```
create table Tickets (  
    id VARCHAR(40) PRIMARY KEY,  
    title VARCHAR(200) NOT NULL,  
    category VARCHAR(11),  
    start_date DATE NOT NULL,  
    end_date DATE,  
    price DECIMAL(4,2),  
    available INT,  
    location VARCHAR(100),  
    provider VARCHAR(50) REFERENCES providers(name)  
);
```

```
create table Transactions (  
    cust_id VARCHAR(40) REFERENCES customers(id),  
    ticket_id VARCHAR(40) REFERENCES tickets(id),  
    ticket_num INT,  
    ticket_price NUMERIC(4,2),  
    date DATE,  
    PRIMARY KEY (cust_id, ticket_id, date)  
);
```

Τα queries που εισάγουν τα δεδομένα στους πίνακες providers και customers βρίσκονται στα ίδια αρχεία με τα CREATE TABLES ενώ τα queries για τους πίνακες tickets και transactions είναι προϊόν προγράμματος python. Τα εν λόγω προγράμματα δημιουργήθηκαν με σκοπό την εγκυρότητα των δεδομένων.



Το `createTickets.py` δέχεται δεδομένα από το αρχείο `ticketsExport` που περιέχει τα δεδομένα για τα εισιτήρια και το αρχείο `providersExport` περιέχει τα δεδομένα για τους `providers`. Με σκοπό την έγκυρη ύπαρξη `provider` σε κάθε εισιτήριο.

Όμοια, το `createTransactions` δέχεται τα δεδομένα από το αρχείο `ticketExport` και `customerExport` που περιέχει τους πελάτες. Με σκοπό δημιουργίας τυχαίων αγορών εισιτηρίων από τους πελάτες.



Ερώτημα 2. Εκτελέστε τις παρακάτω ερωτήσεις (queries) στη ΒΔ (εντολές SELECT).

Τα queries των ερωτημάτων βρίσκονται στο αρχείο QUERIES.txt

- a) `select sum(ticket_num), category
from tickets , transactions
where tickets.id = transactions.ticket_id
group by category`

Ως πωλήσεις σε αυτό το ερώτημα θεωρούμε τον αριθμό των εισιτηρίων.

```
1 select sum(ticket_num), category
2   from tickets , transactions
3   where tickets.id = transactions.ticket_id
4   group by category
```

	Data Output	Explain	Messages	Notifications
	sum bigint	category character varying (11)		
1	7195	Concert		
2	3892	Presentaion		
3	3710	Party		
4	5543	Show		
5	4982	Exhibition		



b) `select category, sum(price*ticket_num) as tziros
from (select category, price, ticket_num
from tickets inner join transactions on id = ticket_id) as triz
group by category
order by tziros desc
limit 1`

Ως πωλήσεις σε αυτό το ερώτημα θεωρούμε τον τζίρο (αριθμός εισιτηρίων * τιμή εισιτηρίου). Για την επιλογή της κατηγορίας με τον μέγιστο τζίρο, ταξινομούμε ως προς τον τζίρο και επιλέγουμε τον πρώτο.

```
1 select category, sum(price*ticket_num) as tziros  
2 from (  
3     select category, price, ticket_num  
4     from tickets inner join transactions on id = ticket_id) as triz  
5 group by category  
6 order by tziros desc  
7 limit 1
```

Data Output Explain Messages Notifications

	category character varying (11)	tziros numeric
1	Concert	338594.71



c) `select category, round(avg(price*ticket_num),3) as avgSales
from tickets inner join (
 select ticket_id, ticket_num
 from customers inner join transactions on id = cust_id
 where age between 16 and 44
) as rng on ticket_id = id
group by category`

Ως κέρδος σε αυτό το ερώτημα θεωρούμε τον τζίρο (αριθμός εισιτηρίων * τιμή εισιτηρίου). Χρησιμοποιούμε την συνάρτηση round για την στρογγυλοποίηση του τζίρου στα 3 δεκαδικά ψηφία.

```
1 select category, round(avg(price*ticket_num),3) as avgSales
2 from tickets inner join (
3     select ticket_id, ticket_num
4     from customers inner join transactions on id = cust_id
5     where age between 16 and 44
6 ) as rng on ticket_id = id
7 group by category|
```

	Data Output	Explain	Messages	Notifications
	category character varying (11)	avgsales numeric		
1	Concert	4391.441		
2	Presentaion	4815.718		
3	Party	3330.520		
4	Show	5395.075		
5	Exhibition	4761.823		



d) `select title, sum(price*ticket_num) as mxTziros
from tickets inner join transactions on id = ticket_id
group by title
order by mxTziros desc
limit 1`

Ως πωλήσεις σε αυτό το ερώτημα θεωρούμε τον τζίρο (αριθμός εισιτηρίων * τιμή εισιτηρίου).

```
1 select title, sum(price*ticket_num) as mxTziros  
2 from tickets inner join transactions on id = ticket_id  
3 group by title  
4 order by mxTziros desc  
5 limit 1
```

Data Output				Explain	Messages	Notifications
	title		mxtziros			
	character varying (200)		numeric			
1	G.I. Joe: The Movie		36034.56			



e) `select distinct on (category) category, name, max(sm) as numoftickets
from (select name, sum(ticket_num) as sm, category
from tickets, customers, transactions
where tickets.id = ticket_id and customers.id = cust_id
group by name, category
) as top
group by name, category
order by category, numoftickets desc`

Ως αγορές σε αυτό το ερώτημα θεωρούμε τον συνολικό αριθμό των εισιτηρίων ανά κατηγορία.

```
1  select distinct on (category) category, name, max(sm) as numoftickets
2  from (
3      select name, sum(ticket_num) as sm, category
4      from tickets, customers, transactions
5      where tickets.id = ticket_id and customers.id = cust_id
6      group by name, category
7      ) as top
8  group by name, category
9  order by category, numoftickets desc
10  |
```


Data Output Explain Messages Notifications

	category character varying (11)	name character varying (50)	numoftickets bigint	
1	Concert	Deloria Benn	366	
2	Exhibition	Sydel Beadell	246	
3	Party	Charmain Janatka	239	
4	Presentaion	Pam Aleksich	266	
5	Show	Guenevere Hardesty	343	



- f) `select providers.name, sum(ticket_num) as totalTickets
from (transactions inner join tickets on id = ticket_id)
inner join
providers on provider = providers.name
group by providers.name
order by totalTickets desc
limit 1`

```
1 select providers.name, sum(ticket_num) as totalTickets
2   from (transactions inner join tickets on id = ticket_id)
3   inner join
4   providers on provider = providers.name
5   group by providers.name
6   order by totalTickets desc
7   limit 1|
```

Data Output		Explain	Messages	Notifications
	name character varying (50)	totaltickets bigint		
1	Meetz	2296		



- g) `select title, sum(ticket_num) as totalTickets
from tickets inner join transactions on id = ticket_id
where date >= '14/4/2019' and date <= CURRENT_DATE
group by title`

Χρησιμοποιήσαμε μια σταθερή ημερομηνία για το query. Για την υλοποίηση με μεταβλητή ημερομηνία μπορούμε να δημιουργήσουμε κάποια συνάρτηση.

```
1 select title, sum(ticket_num) as totalTickets
2   from tickets inner join transactions on id = ticket_id
3   where date >= '14/4/2019' and date <= CURRENT_DATE
4   group by title
```

	Data Output	Explain	Messages	Notifications
	title character varying (200)	totaltickets bigint		
1	Brady Bunch Movie, The	73		
2	The Beautiful Story	408		
3	Serving Sara	64		
4	Mississippi Masala	116		
5	Titanic	124		
6	Last Angry Man, The	91		
7	High Tech, Low Life	100		
8	Jim Jefferies: BARE	166		
9	Gunbuster (Top wo Narae)	126		
10	Out California Wav	42		



Ερώτημα 3. Υλοποίηση triggers και cursors

a) create or replace function dateChanged()
returns trigger language plpgsql as \$\$

```
begin
if new.end_date < old.end_date then
    update transactions set ticket_price = ticket_price*0.8 where ticket_id
old.id;
end if;
return new;
end;
$$
```

```
create trigger dateChanged
before update on tickets
for each row execute procedure dateChanged()
```

Το trigger που υλοποιήσαμε ενεργοποιείται κάθε φορά που γίνεται ανανέωση στον πίνακα tickets. Αν η νέα ημερομηνία είναι πριν την παλιά, η τιμή των αγορασμένων εισιτηρίων (οι εγγραφές του πίνακα transactions) μειώνεται κατά 20%.

```
1 create or replace function dateChanged()
2     returns trigger language plpgsql as $$
3
4 begin
5 if new.end_date < old.end_date then
6     update transactions set ticket_price = ticket_price*0.8 where ticket_id = old.id;
7 end if;
8 return new;
9 end;
10 $$
11
12 create trigger dateChanged
13 before update on tickets
14 for each row execute procedure dateChanged()
15
16 --old date was '2018-08-09'
17 update tickets set end_date = '2018-08-07' where id = '614546f2-409e-4f53-82c3-08187125c076';
```

Data Output Explain Messages Notifications

UPDATE 1

Query returned successfully in 81 msec.



```
b) create or replace function getTicketsUntilToday(dt date)
    returns table (title text, totalTickets integer) as $$
declare
    recTick record;
    curTick cursor (dt date)
        for select tickets.title as tTitle, sum(ticket_num) as tTick
        from tickets inner join transactions on id = ticket_id
        where transactions.date >= dt
            and transactions.date <= CURRENT_DATE
        group by tickets.title;

begin
    open curTick(dt);
    loop
        fetch curTick into recTick;
        exit when not found;
        title := recTick.tTitle;
        totalTickets := recTick.tTick;
        return next;
    end loop;
    close curTick;
end; $$
language plpgsql;

select * from getTicketsUntilToday('14-04-2018');
```

Το cursor που δημιουργήσαμε υλοποιεί το υποερώτημα (g).

```
1 create or replace function getTicketsUntilToday(dt date)
2   returns table (title text, totalTickets integer) as $$
3 declare
4   recTick record;
5   curTick cursor (dt date) for select tickets.title as tTitle, sum(ticket_num) as tTick
6   from tickets inner join transactions on id = ticket_id
7   where transactions.date >= dt and transactions.date <= CURRENT_DATE
8   group by tickets.title;
9 begin
10  open curTick(dt);
11  loop
12    fetch curTick into recTick;
13    exit when not found;
14    title := recTick.tTitle;
15    totalTickets := recTick.tTick;
16    return next;
17  end loop;
18  close curTick;
19 end; $$
20 language plpgsql;
21
22 select * from getTicketsUntilToday('14-04-2018')
23
```



Ερώτημα 4 (20%). Σύνδεση ΒΔ με JDBC client

Δημιουργήσαμε ένα πρόγραμμα κονσόλας που υλοποιεί το υποερώτημα c.

```
public static void main(String[] args) {
    Connection c = null;
    PreparedStatement pstmt= null;
    ResultSet rs = null;

    try {
        String user = "postgres";
        String password = "Geomimo99";
        String url = "jdbc:postgresql://localhost:5432/TicketService";

        c = DriverManager.getConnection(url, user, password);
        System.out.println("Opened Database Successfully!");

        Scanner inp = new Scanner(System.in);
        boolean f = true;
        do {
            System.out.println("Choose action:\n"
                + "A: Average Sales per Category.\n"
                + "B: Top Show (not functional).\n"
                + "C: Tickets Sold Until Today (not functional)");
            switch(inp.next()) {
                case("A"):
                    f = false;
                    break;
                case("B"):
                    break;
                case("C"):
                    break;
            }
        }
    }
}
```

Γίνεται σύνδεση στη βάση και εμφανίζεται ένα μενού επιλογών. Όπως φαίνεται και στο μενού, μόνο η επιλογή A είναι λειτουργική.



```
String sql = "select category, round(avg(price*ticket_num),3) as avgSales\r\n" +
    "from tickets inner join (\r\n" +
    "    select ticket_id, ticket_num\r\n" +
    "    from customers inner join transactions on id = cust_id\r\n" +
    "    where age between ? and ?\r\n" +
    ") as rng on ticket_id = id\r\n" +
    "group by category";

pstmt = c.prepareStatement(sql);
pstmt.setInt(1, 16);
pstmt.setInt(2, 44);

rs = pstmt.executeQuery();

String spaces = " ".repeat(15);
System.out.println("Category" + spaces + "AvgSales");
System.out.println("-----");
while(rs.next()) {
    String category = rs.getString("category");
    float avgSales = rs.getFloat("avgSales");

    spaces = " ".repeat(15 + ("Category".length() - category.length()));
    String output = category + spaces + avgSales;
    System.out.println(output);
}
```

Χρησιμοποιήσαμε PreparedStatement για την δημιουργία του sql ερτήματος, το οποίο εκτελείται και στη συνέχεια μέσω ενός ResultSet ανακτούμε και προβάλουμε τα δεδομένα.

```
Console
<terminated> Main [Java Application] C:\Program Files\Java\jdk-11.0.1\bin\javaw
Opened Database Successfully!
Choose action:
A: Average Sales per Category.
B: Top Show (not functional).
C: Tickets Sold Until Today (not functional)
B
Choose action:
A: Average Sales per Category.
B: Top Show (not functional).
C: Tickets Sold Until Today (not functional)
A
Category          AvgSales
-----
Concert            4391.441
Presentaion        4815.718
Party              3330.52
Show               5395.075
Exhibition         4761.823
```