Praktiskā darba uzdevums kursā “Datu bāzes I” 2020./2021. m.g. pavasara semestrī

Dalībnieki:

gt20010 - Georgs Toliašvili

Biznesa problēmas formulējums

Mūsdienās katram veikalam vajadzētu būt interneta versijai. Īpaši, aktuāli tas ir šobrīd, valstī izsludinātās COVID-19 ārkārtas situācijas dēļ. Tāpēc es esmu izlēmis izveidot savu interneta veikalu, lai varētu to izmantot savam portfolio. Precīzāk, es vēlos izveidot elektroierīču interneta veikalu. Šo praktisko darbu es apvienošu ar tīmekļa tehnoloģiju kursa pēdējo darbu. Interneta veikalam izveidošu nelielu lietotāja saskarni, kas attēlos preces no datu bāzes, lai lietotājs varētu atrast sev nepieciešamo preci, tiks izveidots meklēšanas lauks, preču kategorijas un papildus filtri preču kategorijās, papildus informācija ir izlasāma biznesa noteikumu sadaļā. Izveidošu arī lietotāju reģistrāciju un autorizāciju, kas atļaus saglabāt informāciju, piemēram, par lietotāju grozu un iepriekšējiem pirkumiem. Lietotāji varēs pievienot vairākas piegādes adreses, kā arī vairākas norēķinu kartes, izveidojot pasūtījumu, lietotājs varēs izvēlēties uz kuru no ievadītājām adresēm tas tiks sūtīts, kā arī kura no ievadītajām norēķinu kartēm tiks izmantota pasūtījuma apmaksai. Pasūtījumus varēs izveidot tikai, ja lietotājs ir autorizējies. Šim darbam plānoju izmantot šādas tehnoloģijas: PHP (Laravel), HTML, CSS, JavaScript, Microsoft SQL Server.

Biznesa prasības:

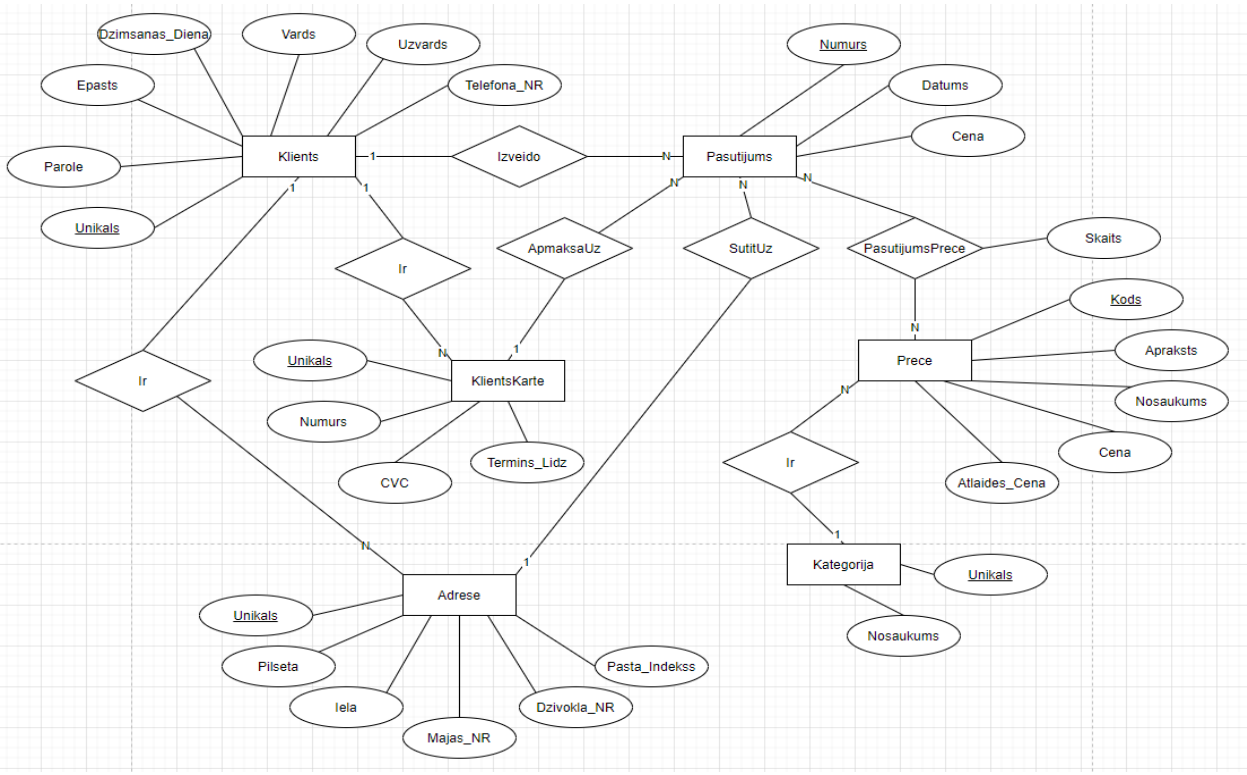
- Lai preces varētu attēlot mājaslapā datubāzē ir nepieciešami dati par precēm, kā arī papildus informācija par tām, piemēram, televizoram, ekrāna izmērs (collās), ekrāna izšķirtspēja u.tml., kad lietotājs uzklikšķina uz preces.
- Datubāzē jāsaglabā informācija par lietotāju, piemēram, e-pasta adrese, parole, vārds un uzvārds.
- Jāsaglabā arī informācija par lietotāja iepriekšējiem pasūtījumiem.

Biznesa noteikumi:

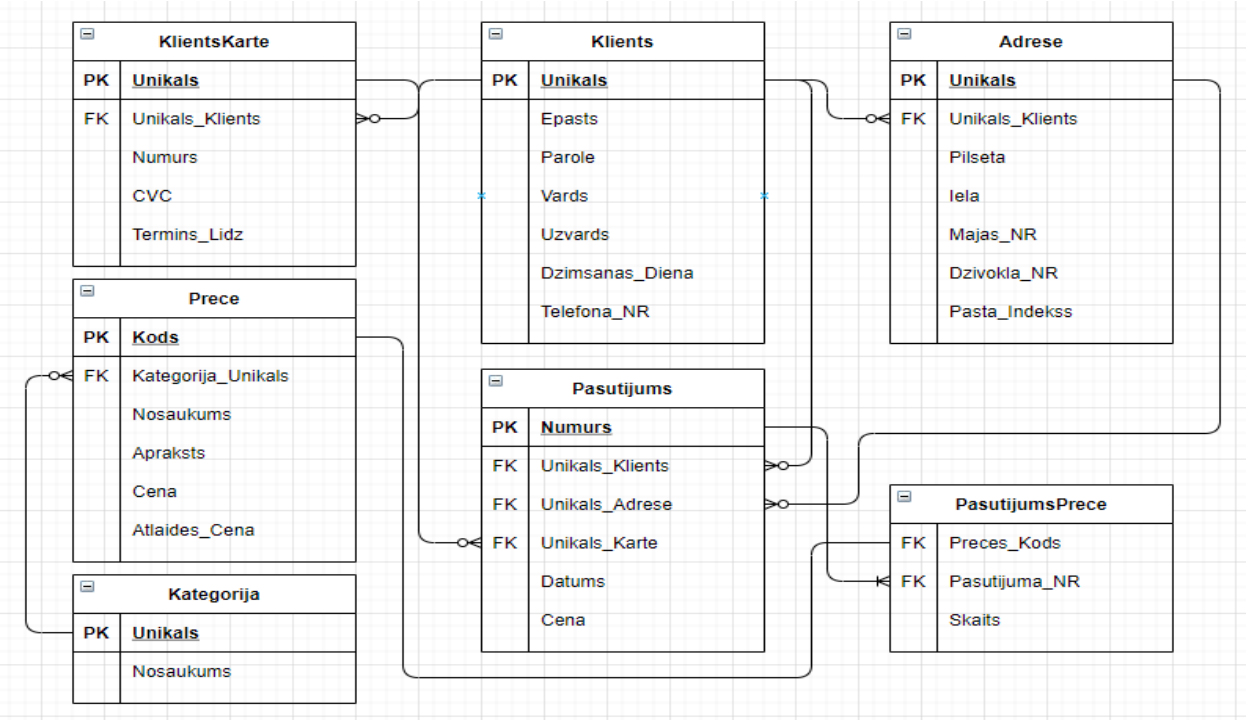
- Lietotājiem jābūt iespējai iegūt papildus informāciju par preci, uz tās uzklikšķinot.
- Varēs izveidot savu personīgo kontu, kurā lietotāji redzēs savus iepriekšējos pasūtījumus.
- Tiks piedāvāta meklēšanas funkcija, lai atrastu sev nepieciešamās preces.
- Būs iespēja pievienot preces iepirkuma grozam, kā arī pārskatīt visas preces, kas atrodas grozā.
- Varēs filtrēt preces ar kategoriju palīdzību, piemēram, datora komponentes, sadzīves tehnika u.tml.
- Lietotāja kontam varēs pievienot vairākas norēķinu kartes, kā arī adreses.
- Lietotājam būs iespēja filtrēt preces noteiktā kategorijā pēc noteiktiem filtriem, piemēram, preces nosaukums no A līdz Z vai otrādi, preces cena no augstākās uz zemāko vai otrādi, kā arī filtrēšana pēc populārākās preces, kas tiek noteikta pēc tā, cik daudz pasūtījumos šī prece ir sastopama.

Datubāzes modeļi

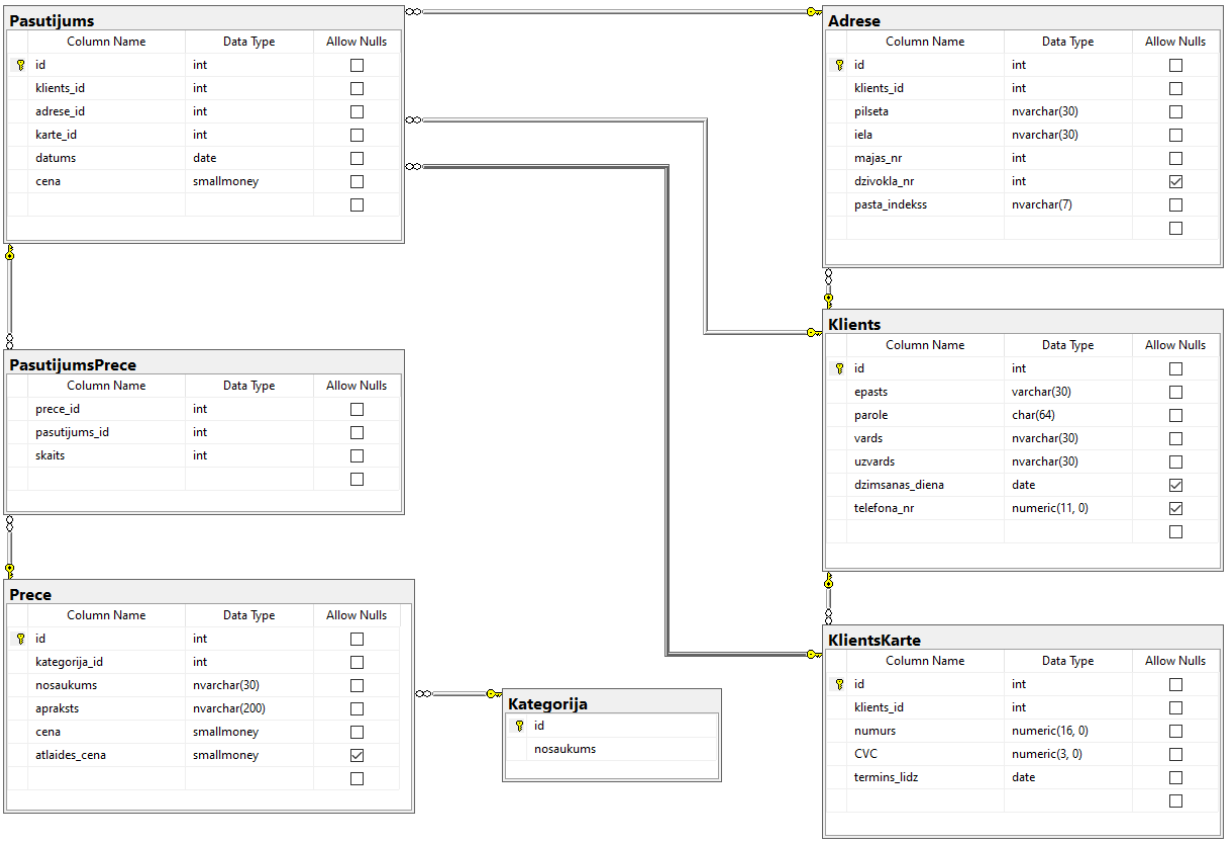
Konceptuālais modelis



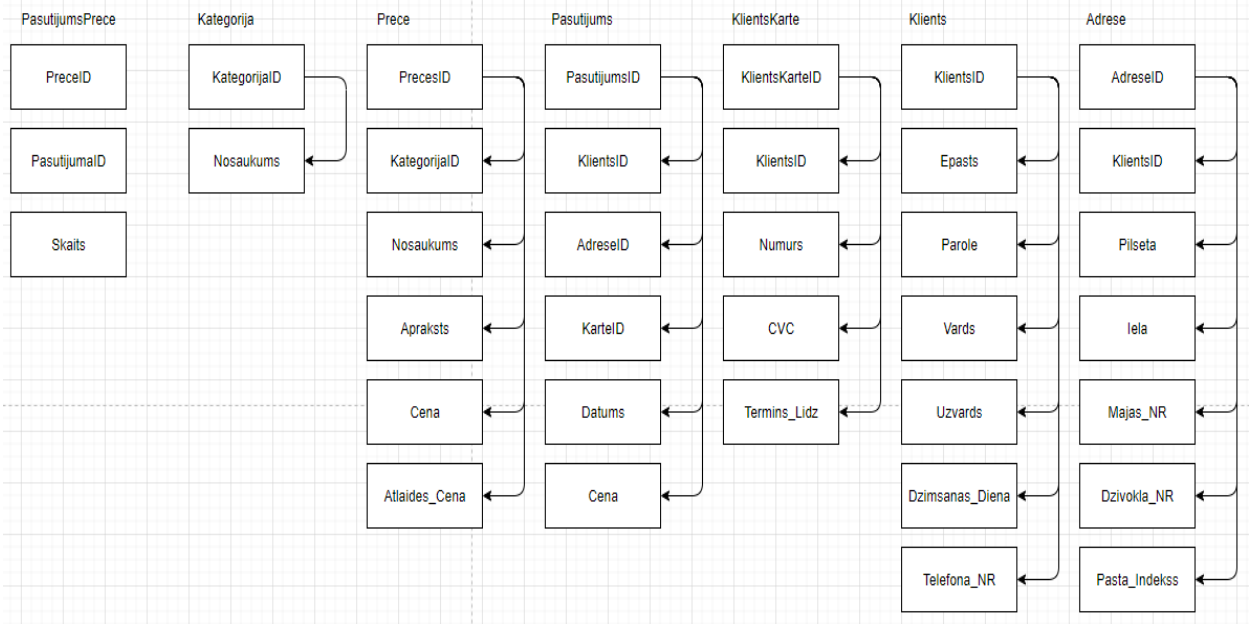
Loģiskais modelis



Fiziskais modelis



Funkcionālo atkarību grafi



Normālformas

Visās tabulas ir 1. normālformā, jo visi lauki ir atomāri.

Visām tabulām eksistē primārās atslēgas (tabulā *PasutijumsPrece* primārās atslēgas attiecīgi ir abu lauku kombinācija, jo katram pasūtījumam var būt tikai viena tāda prece). Tātad visas tabulas ir 2. normālformā.

Visām tabulām funkcionālās atkarības sākas no pilnām primārajām vai potenciālajām atslēgām. Tātad visas tabulas ir BCNF un 3. normālformā.

Skripti tabulu veidošanai

```
create table Klients (
	id int identity(1,1) primary key,
	epasts char(30) not null,
	parole char(64) not null,
	vards nvarchar(30) not null,
	uzvards nvarchar(30) not null,
	dzimsanas_diena date,
	telefona_nr numeric(11,0)
)

create table KlientsKarte (
	id int identity(1,1) primary key,
	klients_id int foreign key references Klients not null,
	numurs numeric(16,0) not null,
	CVC numeric(3,0) not null,
	termins_lidz date
)

create table Adrese (
	id int identity(1,1) primary key,
	klients_id int foreign key references Klients not null,
	pilseta nvarchar(30) not null,
	iela nvarchar(30) not null,
	majas_nr int not null,
	dzivokla_nr int,
	pasta_indekss nvarchar(6) not null
)

create table Pasutijums (
	id int identity(1,1) primary key,
	klients_id int foreign key references Klients not null,
	adrese_id int foreign key references Adrese not null,
	karte_id int foreign key references KlientsKarte not null,
	datums date not null,
	cena smallmoney not null
)
```

```
create table Prece (
	id int identity(1,1) primary key,
	nosaukums nvarchar(30) not null,
	apraksts nvarchar(200) not null,
	cena smallmoney not null,
	atlaides_cena smallmoney
)


create table PasutijumsPrece (
	prece_id int foreign key references Prece not null,
	pasutijums_id int foreign key references Pasutijums not null,
	skaits int not null
)
```

Skripti datu ievietošanai tabulās

Klients

INSERT INTO Klients VALUES ('janis@gmail.com','2bb80d537b1da3e38bd30361aa855686bde0eacd7162fef6a25fe97bf527a25b', 'Jānis', 'Ozols', NULL, 20565555)

INSERT INTO Klients VALUES ('davis@gmail.com','2bb80d537b1da3e38bd30361aa855686bde0eacd7162fef6a25fe97bf527a25b', 'Dāvis', 'Ozols', NULL, 22565555)

INSERT INTO Klients VALUES ('markuss@gmail.com','2bb80d537b1da3e38bd30361aa855686bde0eacd7162fef6a25fe97bf527a25b', 'Markuss', 'Ozols', NULL, 23565555)

Adrese

INSERT INTO Adrese VALUES (100, 'Rīga', 'Krišjāņa', 12, 31, 'LV-1092')

INSERT INTO Adrese VALUES (99, 'Rīga', 'Krišjāņa', 12, 112, 'LV-1092')

INSERT INTO Adrese VALUES (98, 'Rīga', 'Krišjāņa', 12, 1, 'LV-1092')

Klients Karte

INSERT INTO KlientsKarte VALUES (1, 1111995215526666, 431, '20220131')

INSERT INTO KlientsKarte VALUES (2, 2222995215526666, 431, '20230131')

INSERT INTO KlientsKarte VALUES (3, 3333995215526666, 431, '20240131')

Kategorija

INSERT INTO Kategorija VALUES ('Mobilie telefoni')

INSERT INTO Kategorija VALUES ('Datori')

INSERT INTO Kategorija VALUES ('Izklaižu tehnika')

Prece

INSERT INTO Prece VALUES ('Telefons A', 'Nodrošina cilvēku komunkāciju', 100.00, 50.00)

INSERT INTO Prece VALUES ('Telefons B', 'Nodrošina cilvēku komunkāciju', 200.00, 150.00)

INSERT INTO Prece VALUES ('Telefons C', 'Nodrošina cilvēku komunkāciju', 300.00, 250.00)

Pasutijums

```
INSERT INTO Pasutijums VALUES (50, 51, 50, '20210131', 500.00)

INSERT INTO Pasutijums VALUES (49, 52, 49, '20210313', 1500.00)

INSERT INTO Pasutijums VALUES (48, 53, 48, '20210407', 200.00)
```

Pasutijums Prece

```
INSERT INTO PasutijumsPrece VALUES (1, 1, 10)

INSERT INTO PasutijumsPrece VALUES (40, 2, 1)

INSERT INTO PasutijumsPrece VALUES (45, 2, 1)
```

Atbilstošo atskaišu/statistiku SQL vaicājumi

1) Atlasam klientus, kuri nav pievienojuši savu maksājumu karti mājaslapā un atlasam to e-pastus, lai varētu tiem nosūtīt ziņu, ja tie izlem pievienot savu karti šīs nedēļas laikā, tie saņem atlaidi savam nākamajam pirkumam.

```
SELECT epasts FROM Klients

WHERE

    NOT EXISTS(SELECT * FROM KlientsKarte WHERE Klients.id = KlientsKarte.klients_id)

    AND

    EXISTS(SELECT * FROM Adrese WHERE Klients.id = Adrese.klients_id)
```

2) Atlasam visas preces kategorijā 'Datori' un pārbaudām vai to cenas ir lielākas par 500.00, kā arī kārtojam šīs preces pēc nosaukuma alfabētiskā secībā, šādi simulējot filtrus, ko kāds lietotājs vārētu izvēlēties mājaslapā.

```
SELECT *

FROM Prece

WHERE Prece.cena > 500 AND Prece.kategorija_id = 2

ORDER BY Prece.nosaukums
```

3) Apvienojam trīs tabulas un no katras tabulas izvēlamies noteiktas kolonas, papildus pārbaudām vai lietotājs ir pievienojis telefona numuru, iegūto informāciju varētu attēlot lietotāja profilā.

```
SELECT Klients.epasts, Klients.vards, Klients.uzvards, Klients.telefona_nr, Adrese.pilseta,
KlientsKarte.numurs FROM Klients

JOIN Adrese ON Klients.id = Adrese.klients_id

JOIN KlientsKarte ON Klients.id = KlientsKarte.klients_id

WHERE Klients.telefona_nr IS NOT NULL
```

4) No adrešu tabulas atlasam pilsētu un pasta indeksu kombinācijas un aprēķinam lietotāju skaitu šajā lokācijā.

```
SELECT pilseta, pasta_indekss, COUNT (id) lietotaju_skaits

FROM Adrese

GROUP BY pilseta, pasta_indekss
```

5) Apvienojam trīs tabulas un no katras tabulas izvēlamies noteiktas kolonas, kārtojam tās pēc pasūtījuma datuma, šo atlasi varētu izmantot, lai pārskatītu pasūtītās preces pasūtījumos.

```
SELECT Pasutijums.id, PasutijumsPrece.skaits, Prece.nosaukums, Prece.cena, Prece.atlaides_cena,
Pasutijums.datums, Pasutijums.cena AS pasutijuma_kopsumma

FROM PasutijumsPrece

JOIN Prece ON PasutijumsPrece.prece_id = Prece.id

JOIN Pasutijums ON PasutijumsPrece.pasutijums_id = Pasutijums.id

ORDER BY Pasutijums.datums
```

6) Atlasam preces pēc to kategorijas un cenas, šos datus varētu izmantot, lai atlasītu dārgākās preces noteiktā kategorijā komanda ‘RANK’ tiek izmantots, lai redzētu preces vietu atkarībā no citām precēm tajā pašā kategorijā.

```
SELECT id, kategorija_id, nosaukums, cena, RANK () OVER ( PARTITION BY kategorija_id ORDER
BY cena DESC) cenas_vieta

FROM Prece
```

7) Atlasam pasūtījumus pēc to veiktā datuma un aprēķinam ieguvumus tajā dienā no visiem veiktajiem pasūtījumiem.

```
SELECT DISTINCT datums, SUM (cena) OVER (PARTITION BY datums) ieguvumi

FROM Pasutijums

ORDER BY ieguvumi DESC
```

8) Apvienojam klientu un pasūtījumu tabulas, iegūstam lietotāja veikto pasūtījumu skaitu, kā arī šo pasūtījumu kopējo cenu.

```
SELECT DISTINCT Klients.id, Klients.epasts, Klients.vards, Klients.uzvards,

        COUNT(*) OVER (PARTITION BY Pasutijums.klients_id) veiktie_pasutijumi,

        SUM(Pasutijums.cena) OVER (PARTITION BY Pasutijums.klients_id) pasutijumu_ieguvumi

FROM Klients, Pasutijums

WHERE Pasutijums.klients_id = Klients.id
```