**Proiect PBD. Classic Cars Rental & Service**

Bogdan-Mihai Podaru

1. **Descrierea domeniului ales**

Proiectul reprezinta o simulare a unei companii de inchiriere de masini clasice, cu posibilitatea de a incheia un contract de vanzare la cererea clientilor. Compania dispune de un service auto de intretinere a masinilor clasice, prin mentinerea unui stoc propriu de componente de masini. Serviciile de vanzare, inchiriere si intretinere sunt adresate de angajati cu titluri diferite in cadrul companiei. Masinile pot fi livrate nu mai devreme de 14 zile iar masinile inchiriate nu pot fi ridicate mai devreme de 7 zile. Acestea pot fi inchiriate pentru o perioada fixa de 21 zile. Toate acestea vor fi referite sub denumirea bazei de date “ClassicCarsRentalAndService”.

1. **Structura bazei de date**

Baza de date a fost insipirata partial din modelul bazei de date Classic Models din SQL Server. Datele legate de comenzile clientilor si detaliile masinilor clasice au fost preluate si adaptate pentru noua structura din Classic Models.

Proiectarea bazei de date a fost ghidata de cele trei mari servicii propuse: vanzare, inchiriere si intretinere/reparatii. Tipurile de date sunt diversificate: nvarchar, int, float, image, text.

* 1. Pentru o structura optima se verifica conditiile de **normalizare**:

1. Prima forma normala

Fiecare coloana trebuie sa contina valori atomice.

Fiecare coloana continue date de acelasi tip.

1. A doua forma normala

Nu sunt prezente chei primare compuse deci relatiile sunt in forma normala 2.

1. A treia forma normala

Nu exista niciun atribut care ar trebui modificat pentru o modificare a altui atribut.

1. Forma normala Boyce-Codd.

Nu exista atribut non-key care sa determine un atribut key.

* 1. **Diagrama** bazei de date este urmatoarea:



Pe baza tabelei ClassicCars am create din coloanal forModel un fisier csv cu toate numele de modele.

Pentru generarea de imagini am creat un script in python care va parsa fisierul csv si va descarca folosind libraria google\_images download imaginile corespunzatoare modelelor de masini din fisierul csv de la intrare. Se va salva un fisier csv cu modelele de masini in prima coloanal si caile spre imaginile corespunzatoare salvate local.

Script-ul python este urmatorul:

import csv  
from google\_images\_download import google\_images\_download  
response = google\_images\_download.googleimagesdownload()  
  
  
def assign\_images():  
 models = ""  
 with open('models.csv', 'r') as file:  
 reader = csv.reader(file)  
 for model in reader:  
 models = models + "," + str(model).replace('[\'', '').replace('\']', '').replace('\"', '').strip()  
 arguments = {"keywords": models[4:], "limit": 1, "print\_urls": True}  
 paths = response.download(arguments)  
 paths, \_ = paths  
 with open('classic\_cars.csv', 'w', newline='') as file:  
 writer = csv.writer(file)  
 writer.writerow(["model", "picture\_path"])  
 for path in paths.keys():  
 writer.writerow([path, paths.get(path)[0].replace('\'', '').replace('\'', '').replace('\\', '\\\\')])

Pentru inserarea de imagini in campul picture din relatia ClassicCars se foloseste urmatorea abordare folosind cursor:

DECLARE C CURSOR FOR

(SELECT model, picture\_path FROM dbo.classic\_cars)

DECLARE @FilePath sysname

DECLARE @model nvarchar(50)

DECLARE @SQL varchar(max)

-- Loop through each file loading the images one by one

OPEN C

FETCH NEXT FROM C INTO @model, @FilePath

WHILE (@@FETCH\_STATUS <> -1)

BEGIN

SET @SQL =

'UPDATE c set c.picture = (SELECT BulkColumn FROM OPENROWSET(

BULK ''' + @FilePath + ''', SINGLE\_BLOB) AS BLOB)

from ClassicCars c

WHERE c.modelName = ''' + @model + ''''

EXEC (@sql)

FETCH NEXT FROM C INTO @model, @FilePath

END

CLOSE C

DEALLOCATE C

Orders a fost populat pe baza tabelei Orders din ClassicModels in felul urmator:

WITH cte AS

(

SELECT \*,

ROW\_NUMBER() OVER (PARTITION BY orderNumber ORDER BY orderNumber ASC) AS rn

FROM orderdetails2

)

update o set o.carCode = c.productCode

from Orders o, cte c

where o.orderNumber = c.orderNumber and c.productCode in (select distinct carCode from ClassicCars) and rn=1;

select \* from cte where productCode in (select distinct carCode from ClassicCars) and rn =1;

Asignarea de reprezentati de vanzari unor client s-a facut random prin:

create view custom\_view as

with cte as(

select orderNumber, ABS(CHECKSUM(NewId())) % 8 + 1 as IDX from Orders

)

select orderNumber, CHOOSE(IDX, 1188,1216,1286,1370,1401,1621,1625,1702) as salesRepr from cte;

go

update o set o.salesRepr = v.salesRepr

from Orders o, custom\_view v

where o.orderNumber = v.orderNumber

La inserarea datelor s-a tinut cont de constrangerea: timpul de retur = timp de ridicare + 21 zile:

Update r set r.returnDate = DATEADD(day,21,r.pickupDate)

from RentalDetails r

* 1. **Relatiile** identificate in urma proiectarii bazei de date sunt:

1. **Relatii 1:M**



Exemple de astfel de relatii sunt relatiile dintre “RentalDetails” si “Orders” (diagrama de mai sus), respectiv intre “SoldDetails” si “Orders”. Diagrama de mai sus surprinde faptul ca se pot face mai multe inchirieri insa o comanda poate avea un singur grup de detalii ale acesteia.

1. **Relatii M:N**



Relatia “Orders” si “ServiceDetails” sunt exemple de relatii de tip “many-to-many” asa cum se poate observa si in exemplul de mai sus. Relatia “ServiceDetails” surprinde faptul ca in reparatia masinilor o piesa se poate folosi pentru mai multe masini, iar o masina poate avea nevoie de mai multe piese pentru a fi reparata.

1. **Relatii recursive**



Tabela “Employees” prezinta o relatie recursiva prin prisma campului “reportsTo” care este o cheie straina cu referinta spre cheia primara din acelasi tabel. Acest camp desemneaza o persoana cu atributie superioara spre care un alt angajat are responsabilitatea de a-si raporta activitatea. (de exemplu un manager de service va raspunde de mecanicii si electricienii unui anumit service).

1. **Constrangeri**
2. **Unicitate**

Evidenta masinilor se face luand in considerare un model o singura data:

alter table ClassicCars

add constraint UK\_modelName unique(modelName);

Pentru a elimina posibilele duplicate in evidenta de service o pereche componenta-model trebuie sa fie unica, in alte cuvinte pentru a repara o masina e nevoie de atat de informatia de piesa cat si de modelul de masina compatibil cu acea piesa:

alter table Inventory

add constraint UK\_CompoentForModel unique(compName,forModel);

1. **Domeniu**

Sa presupunem ca politia firmei nu permite comisioane mai mari de 10% la inchiriere:

ALTER TABLE RentalDetails

ADD CONSTRAINT DC\_comission CHECK (comission < 10);

Totodata trebuie sa verificam preturile de vanzare si inchiriere sa fie intotdeauna pozitive:

ALTER TABLE SoldDetails

ADD CONSTRAINT DC\_buyPrice CHECK (buyPrice > 0);

ALTER TABLE Orders

ADD CONSTRAINT DC\_total CHECK (total > 0);

1. **Integritate referentiala**

La stergerea unei comenzi, fie de tip inchiriere, fie de tip vanzare, detaliile comenzii trebuiec sterse de asemenea:

ALTER TABLE SoldDetails

ADD FOREIGN KEY (orderNumber) REFERENCES Orders(orderNumber)

ON DELETE CASCADE;

ALTER TABLE RentalDetails

ADD FOREIGN KEY (orderNumber) REFERENCES Orders(orderNumber)

ON DELETE CASCADE ;

1. **Vederi**

In cele ce urmeaza se vor crea vederi cu utilitate in rezolvarea cerintelor ulterioare:

* Masinile care au fost vandute cu un pret mai mic decat cel dat de producator:

create view maybe\_no\_profit as

select modelName,modelDescription,picture, buyPrice, MSRP

from ClassicCars join Orders on ClassicCars.carCode = Orders.carCode join SoldDetails on Orders.orderNumber = SoldDetails.orderNumber

where buyPrice < MSRP;

go

O imagine care conține text, masă

Descriere generată automatO parte din rezultate:

* Datorita faptului ca anumite componente sunt compatibile cu diferite modele vrem sa vedem acele componente pentru care recomandarea de folosire este diferita de utilizarea lor reala:

create view components\_used\_as\_intended as

select compName, forModel as Intended, modelName as Actually, quantity, compUnitPrice

from Inventory join ServiceDetails on Inventory.compId = ServiceDetails.compId join CarService on ServiceDetails.checkId = ServiceDetails.checkId join ClassicCars on ClassicCars.carCode = CarService.carCode

where forModel <> modelName

go

O imagine care conține masă

Descriere generată automatUn esantion din rezultate:

* View pentru a vedea dupa cat timp un client a venit sa ridice masina pe care tocmai a inchirat-o, totalul de plata, modelul masinii, datele de contact ale Clientului si comisionul aplicat:

create view rentail\_details\_for\_customer as

select Customers.contactName, Customers.address, Customers.city, modelName, MSRP,

CAST(comission as nvarchar(2)) +' %' as comission, total, DATEDIFF(day, orderDate, pickupDate) as PickedAfter

from Customers join Orders on Customers.customerNumber = Orders.customerNumber join ClassicCars on ClassicCars.carCode = Orders.carCode join RentalDetails on RentalDetails.orderNumber = Orders.orderNumber;

go

O imagine care conține text, interior, captură de ecran

Descriere generată automatCu urmatorul rezultat:

* Masinile care au necesitat cele mai multe reparatii sau revizii, clientii care le-au folosit, piesele folosite pentru reparatii si tehnicienii care le-au reparat:

create view attrited\_cars as

select

modelName, ClassicCars.stockQuantity as RemainingCars,

Customers.contactName,

compName, quantity, compUnitPrice,

fullName as Technician

from

Customers join Orders on Orders.customerNumber = Customers.customerNumber

join ClassicCars on ClassicCars.carCode = Orders.carCode

join CarService on CarService.carCode = ClassicCars.carCode

join ServiceDetails on CarService.checkId = ServiceDetails.checkId

join Inventory on Inventory.compId = ServiceDetails.compId

join Employees on CarService.Responsible = Employees.employeeNumber

group by

modelName, ClassicCars.stockQuantity,

Customers.contactName,

compName, quantity, compUnitPrice,

fullName

having

count(modelName) >= 2

go

O imagine care conține masă

Descriere generată automatRezultatul se poate observa in imaginea urmatoare:

1. **Functii Transact-SQL**
2. Suma de bani investita intr-o masina pentru reparatii al carui model e dat ca parametru:

create function dbo.getRepairCosts(@model nvarchar(50))

returns int

as

begin

declare @totalCost float;

select

@totalCost = SUM(ServiceDetails.quantity\*Inventory.compUnitPrice)

from

ServiceDetails join Inventory on ServiceDetails.compId = Inventory.compId

join CarService on CarService.checkId = ServiceDetails.checkId

join ClassicCars on ClassicCars.carCode = CarService.carCode

group by

modelName

having

modelName = @model;

return @totalCost;

end

go

select '1911 Ford Town Car' as carModel, dbo.getRepairCosts('1911 Ford Town Car') as TotalCostOnRepairs;

O imagine care conține masă

Descriere generată automatCu rezultatul:

1. Incasarile pentru inchirieri pentru luna data ca parametru (fiind o valoare de tip String):

create function dbo.getIncomesPerMonth(@month nvarchar(10))

returns int

as

begin

declare @incomes float;

declare @monthInt int;

select @monthInt =

case

when @month = 'January' then 1

when @month = 'February' then 2

when @month = 'March' then 3

when @month = 'April' then 4

when @month = 'May' then 5

when @month = 'June' then 6

when @month = 'July' then 7

when @month = 'August' then 8

when @month = 'September' then 9

when @month = 'October' then 10

when @month = 'November' then 11

else 12

end;

select @incomes = SUM(total) from Orders join RentalDetails on Orders.orderNumber = RentalDetails.orderNumber

group by orderDate

having MONTH(orderDate) = @monthInt;

return @incomes;

end

go

select 'March' as Month, dbo.getIncomesPerMonth('March') as MonthIncome

O imagine care conține masă

Descriere generată automatCu urmatorul rezultat:

1. **Proceduri stocate parametrizate**
2. **Cautare:**

Aflarea detaliilor comenzii pentru un client dat ca parametru, care sa includa si specificarea tipului comenzii (inchiriere sau vanzare). O abordare posibila poate fi urmatoare folosind UNION:

create procedure dbo.findOrderDetailsForCustomer(@customerName nvarchar(50))

as

begin

select total, orderDate,modelName, [Sales Agent], [Order type] from (

select contactName, total, orderDate, modelName, fullName as 'Sales Agent', 'Sell' as 'Order type'

from Orders join SoldDetails on Orders.orderNumber = SoldDetails.orderNumber

join Employees on Orders.salesRepr = Employees.employeeNumber

join ClassicCars on Orders.carCode = ClassicCars.carCode

join Customers on Orders.customerNumber = Customers.customerNumber

UNION

select contactName, total, orderDate, modelName, fullName as SalesAgent, 'Rent' as 'Order type'

from Orders join RentalDetails on Orders.orderNumber = RentalDetails.orderNumber

join Employees on Orders.salesRepr = Employees.employeeNumber

join ClassicCars on Orders.carCode = ClassicCars.carCode

join Customers on Orders.customerNumber = Customers.customerNumber

) q

where q.contactName = @customerName;

end

go

exec dbo.findOrderDetailsForCustomer 'Carine Schmitt';

O imagine care conține text

Descriere generată automatCu urmatorul rezultat:

1. **Actualizare:**

O functionalitate de nivel inalt ar fi necesara pentru actualizarea stocului de masini atunci cand se realizeaza o comanda de tip vanzare, astfel am putea pastra relatia “ClassicCars” la un numar relativ mic de intrari:

create procedure dbo.updateStockOnCarBuying (@model nvarchar(50))

as

begin

update ClassicCars set stockQuantity = stockQuantity - 1

where modelName = @model;

end

go

exec dbo.updateStockOnCarBuying '2001 Ferrari Enzo';

Aceasta functionalitatea va putea fi utilizata in cadrul trigerelor (la realizarea unei comenzi sa se apeleze automat) si rapoartelor.

1. **Stergere:**

Managementul personalului presupune si posibilitatea concedierii. Odata conecediat un tehnician responsabilitatile lui legate de o masina se ridica, astfel nu mai sunt necesare informatiile referitoare la munca sa in service. Urmatorul query surprinde acest lucru:

create procedure dbo.fireEmployeeAndUpdateLogs (@employeeName nvarchar(50))

as

begin

declare @empNumber int;

select @empNumber = employeeNumber from Employees where @employeeName = fullName and jobTitle in ('Mechanic','Electrician');

delete from Employees where @empNumber = employeeNumber;

delete from CarService where Responsible = @empNumber;

end

go

1. **Inserare:**

Inserarea unei comenzi noi specificand detalii legate de produs, serviciu (vanzare/inchiriere), clientul produsului si anagajat:

create procedure dbo.insertNewOrder (

@customerName nvarchar(50),

@salesAgentName nvarchar(50),

@modelName nvarchar(50),

@total float,

@type nvarchar(10),

@status nvarchar(10),

-- optional parameters

@comission int = NULL,

@shipDate date = NULL,

@pickupDate date = NULL,

@returnDate date = NULL

)

as

begin

declare @orderNumber int;

declare @salesRep int;

declare @customerNumber int;

declare @carCode nvarchar(10);

select @orderNumber = max(orderNumber)+1 from Orders;

select @customerNumber = customerNumber from Customers where contactName = @customerName;

select @salesRep = employeeNumber from Employees where fullName = @salesAgentName and jobTitle in ('Sales Rep','Sales Manager(APAC)','Sales Manager(EMEA)','Sales Manager(NA)');

select @carCode = carCode from ClassicCars where modelName = @modelName;

insert into Orders(orderNumber, customerNumber, salesRepr, carCode, total, orderDate, status)

values(@orderNumber, @customerNumber, @salesRep, @carCode, @total, GETDATE(), @status);

if @type = 'sell'

insert into SoldDetails(orderNumber, shipDate, buyPrice)

values(@orderNumber, @shipDate, @total);

else -- 'rent'

insert into RentalDetails(orderNumber, pickupDate, returnDate, comission)

values(@orderNumber, @pickupDate, @returnDate, @comission);

end

go

exec dbo.insertNewOrder 'Mary Saveley','Gerard Hernandez','1904 Buick Runabout', 11000.500, 'sell', '2020-12-26';

1. **Triggere**
2. **Inserare**

Dupa cum spuneam mai sus, la inserarea unei comenzi noi de tip vanzare e nevoie sa se actualizeze stock-ul de masini, astfel:

create trigger trgUpdateStockOnNewSellOrder

on Orders

after insert

as

declare @carCode nvarchar(10);

declare @model nvarchar(50);

declare @existingQuantity int;

begin

select @carCode = carCode from inserted;

select @existingQuantity = stockQuantity, @model = modelName from ClassicCars where carCode = @carCode;

if (@existingQuantity = 0)

begin

raiserror ('Model %s is out of stock!',16,1, @model);

ROLLBACK TRANSACTION;

end

exec dbo.updateStockOnCarBuying @model;

return

end

go

1. **Actualizare**

Pentru ca datele sa fie consistente nu ar trebui sa se poate modifica datele angajatilor care sunt implicati in comenzi nefinalizate:

create trigger trgUpdateEmployeesWithTransitiveOrders

on Employees

after update

as

declare @empId int;

begin

select @empId = employeeNumber from deleted; -- we need values before update

if @empId in (select salesRepr from Orders where status <> 'Sold')

begin

raiserror('Employee details could not be changed',16,1);

ROLLBACK TRANSACTION;

end

return

end

go

1. **Stergere**

Un exemplu de utilizare ar fi sa nu se permita stergerea comenzilor in curs de procesare:

create trigger trgTransitiveOrders

on Orders

after delete

as

declare @status nchar(10);

begin

select @status = status from deleted;

if @status <> 'Sold'

begin

raiserror('Transaction is still disputed',16,1);

ROLLBACK TRANSACTION;

end

return

end

go

1. **“Instead Of” Trigger:**

O utilitate posibila ar fi combinatia intre un view si un trigger. Odata examinata o masina in service trebuiesc actualizate toate tabelele cu privire la prezenta sa in service (de ex: pisele folosite la reparatie, descrierea problemei, responsabilul de service, etc.)

Pentru aceasta putem crea un view care sa tina evidenta masini-piese folosite, astfel:

create view CarComponentsHistory as

select

compName,forModel, quantity, serviceAddress, estimatedPeriod, modelName, problemDescription, fullName as Responsible

from ServiceDetails join Inventory on ServiceDetails.compId = Inventory.compId

join CarService on CarService.checkId = ServiceDetails.checkId

join ClassicCars on CarService.carCode = ClassicCars.carCode

join Employees on Employees.employeeNumber = CarService.Responsible

go

Doua inserari in relatiile “CarService” si “ServiceDetails” se pot comprima printr-o combinatie de inserare pe noul view create + un trigger de tip “instead of”, astfel:

create trigger trgInsertOnCarServiceAndServiceDetails

on dbo.CarComponentsHistory

INSTEAD OF INSERT

as

begin

declare @compId int;

declare @checkId int;

declare @employeeNumber int;

declare @compName nvarchar(30);

declare @forModel nvarchar(30);

declare @quantity int;

declare @serviceAddress nvarchar(50);

declare @estimatedPeriod int;

declare @modelName nvarchar(30);

declare @responsible nvarchar(50);

declare @responsibleId int;

declare @stockQuantity int;

declare @problemDescription nvarchar(100);

declare @carCode nvarchar(10);

select @compName = compName,

@forModel = forModel,

@quantity = quantity,

@serviceAddress = serviceAddress,

@estimatedPeriod = estimatedPeriod,

@modelName = modelName,

@problemDescription = CAST(problemDescription as nvarchar(100)),

@responsible = Responsible

from inserted;

select @carCode = carCode from ClassicCars where modelName = @modelName;

select @responsibleId = Employees.employeeNumber from Employees where jobTitle in ('Mechanic','Electrician') and fullName = @responsible;

select @compId = compId, @stockQuantity = stockQuantity from Inventory where compName = @compName and @forModel = forModel;

-- data validation

if @compId is NULL or @quantity > @stockQuantity or @carCode is NULL or @responsibleId is NULL

begin

raiserror('Invalid data on insert',16,1);

ROLLBACK TRANSACTION;

end;

insert into ServiceDetails(compId,checkId,quantity)

values(@compId,@checkId,@quantity);

insert into CarService(serviceAddress,estimatedPeriod,carCode,problemDescription,

Responsible)

values(@serviceAddress, @estimatedPeriod, @carCode,

@problemDescription, @responsible);

return

end

go

1. **Rapoarte**
2. **Window Functions**

Afisarea numarului de comenzi pentru fiecare tip de masina in parte, cu detalii despre masina si comanda (inclusiv tipul comenzii). Se vor afisa subtotaluri pentru comenzi de tip vanzare si pentru comenzi de tip inchiriere.

Raportul are la baza query-ul:

select top 50 total, orderDate, modelName, picture, [Sales Agent],

COUNT(orderNumber) OVER(PARTITION BY q.carCode) as 'Total orders',

[Order type] from (

select Orders.orderNumber, contactName, total, orderDate, ClassicCars.carCode, fullName as 'Sales Agent', 'Sell' as 'Order type'

from Orders join SoldDetails on Orders.orderNumber = SoldDetails.orderNumber

join Employees on Orders.salesRepr = Employees.employeeNumber

join ClassicCars on Orders.carCode = ClassicCars.carCode

join Customers on Orders.customerNumber = Customers.customerNumber

UNION

select Orders.orderNumber, contactName, total, orderDate, ClassicCars.carCode, fullName as SalesAgent, 'Rent' as 'Order type'

from Orders join RentalDetails on Orders.orderNumber = RentalDetails.orderNumber

join Employees on Orders.salesRepr = Employees.employeeNumber

join ClassicCars on Orders.carCode = ClassicCars.carCode

join Customers on Orders.customerNumber = Customers.customerNumber

) q join ClassicCars on q.carCode = ClassicCars.carCode order by [Total orders] ASC;

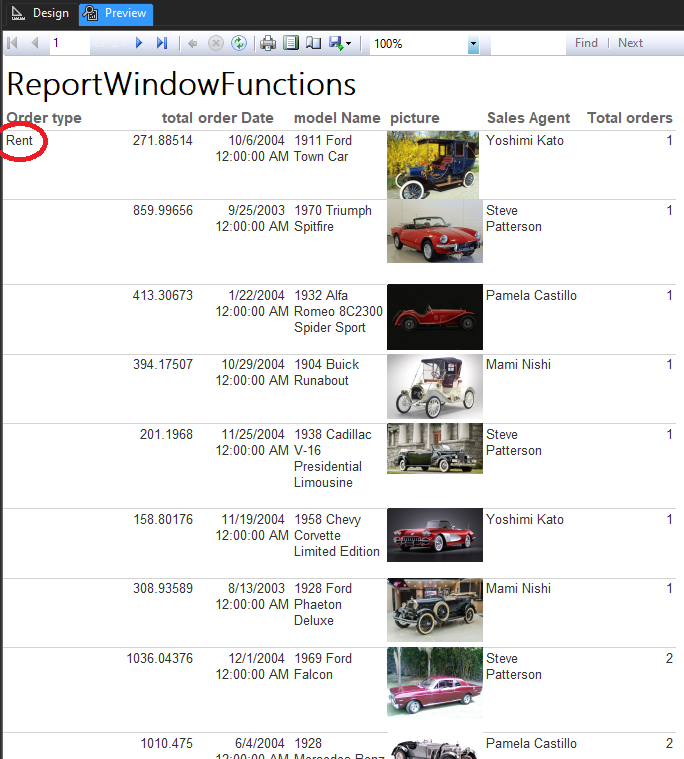
Cu design-ul raportului:

O imagine care conține masă

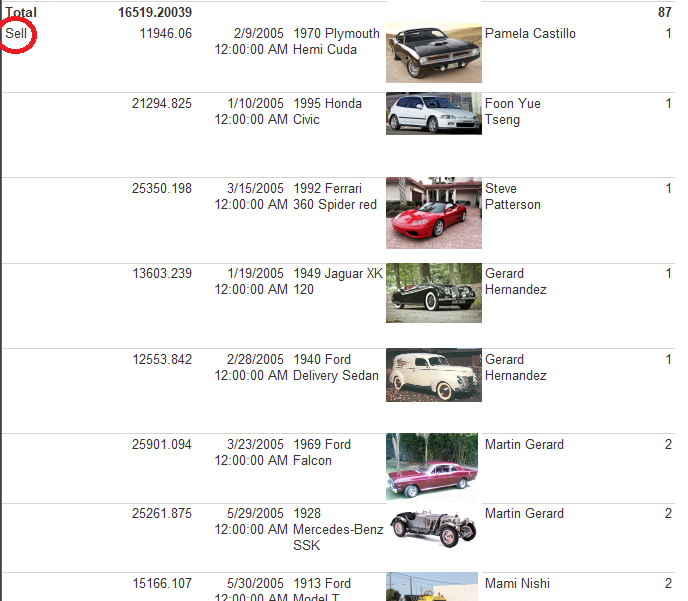
Descriere generată automat

Cu urmatoarele rezultate

* Pentru rent:



* Pentru sell:



1. **Use of images**

Pentru a tine evidenta stocului e nevoie uneori sa aflam care sunt piesele cele mai utilizate. Pentru aceasta, descriem un raport parametrizat care ofera informatii despre masinile din service care au avut nevoie de o anumita piesa de schimb (parametru raportului). Raportul va include detalii precum: pretul total folosit pentru o anumita masina (cantitate \* pret unitar), modelul masinii, imaginea sa, perioada estimata de reparatie, responsabilul de service si problema pentru care masina e in service:

Raportul va folosi urmatorul query:

select quantity, compUnitPrice,

estimatedPeriod, modelName, problemDescription, picture,

fullName as Responsible

from ServiceDetails join Inventory on ServiceDetails.compId = Inventory.compId

join CarService on CarService.checkId = ServiceDetails.checkId

join Employees on Employees.employeeNumber = CarService.Responsible

join ClassicCars on ClassicCars.carCode = CarService.carCode

where compName = @compName;

Design-ul continue un parametru “compName”, o expresie (unitPrice \* quantity ) si un field de tip Image:

O imagine care conține text

Descriere generată automat

Rezultatul poate fi observat in imaginea urmatoare:

O imagine care conține text

Descriere generată automat

1. **Graphics**

Raport priviind componentele folosite in reparatiile masinilor. Fiecare intrare va contine o histograma care va descrie numarul de componente/revizii folosite pentru un anumit tip de masina pe toata durata in care masina a fost detinuta de companie:

select modelName, compName, sum( compUnitPrice \* quantity ) as Total

from Inventory join ServiceDetails on Inventory.compId = ServiceDetails.compId

join CarService on CarService.checkId = ServiceDetails.checkId

join ClassicCars on ClassicCars.carCode = CarService.carCode

group by compName, modelName;

Pentru a putea obtine un subgrafic care sa contina cantitatea din fiecare piesa folosita e nevoie de un subraport parametrizat:

select compName, sum(quantity) as [Component Count]

from Inventory join ServiceDetails on Inventory.compId = ServiceDetails.compId

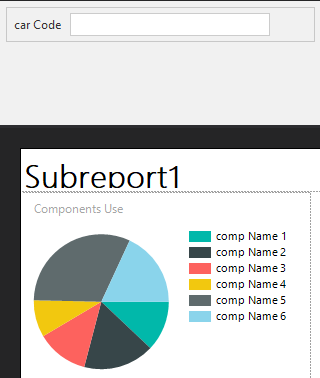
join CarService on CarService.checkId = ServiceDetails.checkId

join ClassicCars on ClassicCars.carCode = CarService.carCode

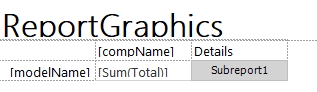
group by compName, ClassicCars.carCode

having ClassicCars.carCode = @carCode;

Design-ul subraportului este urmatorul:



Design-ul raportului este urmatorul:



O imagine care conține masă

Descriere generată automatPutem vizualiza rezultatele:

1. **Subreports**

Printre cele mai utile rapoarte in cadrul companiei sunt rapoarte legate de comenzi, astfel e nevoie de o privire de ansamblu in cadrul comenzilor dar si de posibilitatea detalierii comenzilor. Astfel se va creea urmatorul raport cu 2 subrapoarte, fiecare va prezenta unde este cazul detalii despre inchiriere sau vanzare:

Query-ul raportului parinte:

select Orders.orderNumber, contactName as customer, fullName as [Sale Representative],

modelName as [Car model],total as Total, orderDate as [Order Date], status as Status

from Orders join Customers on Orders.customerNumber = Customers.customerNumber

join Employees on Employees.employeeNumber = Orders.salesRepr

join ClassicCars on ClassicCars.carCode = Orders.carCode;

Subrapoartele se leaga de raportul parinte prin campul orderNumber:

* Subraport pentru inchirieri:

select pickupDate as [Pickup Date], returnDate as [Return Date], comission as Comission

from RentalDetails where orderNumber = @orderNumber;

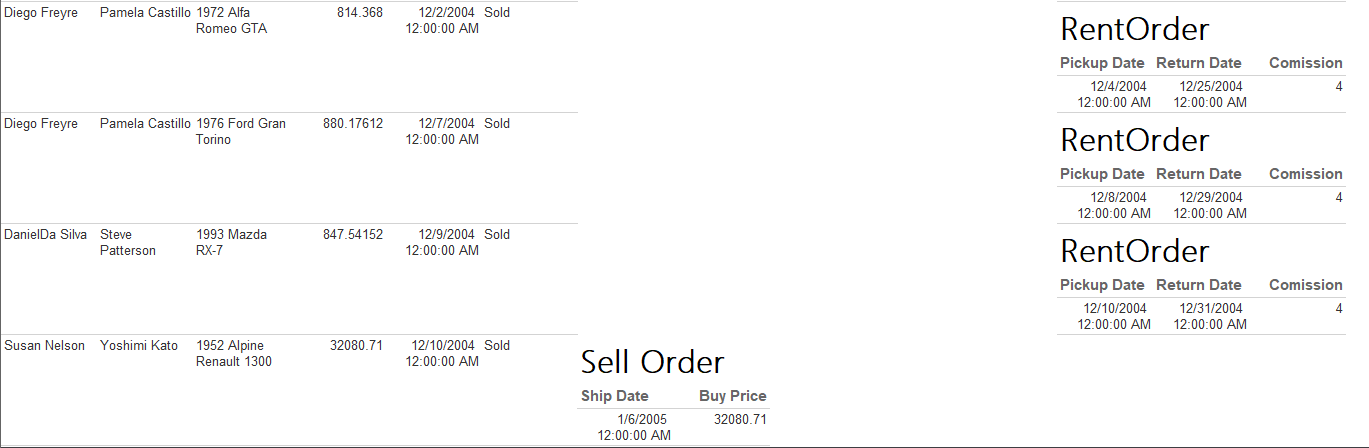
* Subraport pentru vanzari:

select shipDate as [Ship Date], buyPrice as [Buy Price] from SoldDetails where orderNumber = @orderNumber;

Design-ul raportului final arata astfel:

O imagine care conține masă

Descriere generată automat

Cu urmatorul rezultat (se observa campul Rent Orders este completat doar atunci cand este vorba de o inchiriere, similar pentru Sell Orders):