

**Base de dados**

**Relatório do Trabalho Prático**

Base de dados Companhia de Táxis

3ºsemestre 2016/2017

Docente: Prof. Pedro Salgueiro

Alunos: João Dias nº 35476

Eduardo Romão nº 35477

**Introdução**

No âmbito da unidade curricular de Base de dados, incorporada no programa de Licenciatura em Engenharia Informática, segundo ano, semestre ímpar, foi-nos pedido que elaborássemos um trabalho prático como instrumento de avaliação. Este trabalho consiste na criação de uma base de dados que armazene toda a informação de uma companhia de táxis.

Para criarmos a respetiva base de dados iremos utilizar oito relações que irão ser preenchidas com as devidas informações. Por ultimo iremos responder a umas querys utilizando a linguagem de álgebra relacional e SQL.

**Exercício 1**

Indique as chaves primárias, candidatas e estrangeiras de cada relação.

|  |  |  |  |
| --- | --- | --- | --- |
| Relação | Chave Candidata | Chave Primária | Chave Estrangeira |
| Motorista | Nbi | Nbi | Não tem |
| Telefone | Telefone e Nbi | Telefone e Nbi | Nbi |
| Modelo | Modelo | Modelo | Não tem |
| Taxi | Matricula | Matricula | Modelo |
| Serviço | Matricula e DataInicio | Matricula e DataInicio | Matricula |
| Turno | Nbi e DataInicio | Nbi e DataInicio | Matricula e Nbi |
| Cliente | Nif | Nif | Não tem |
| Pedido | Nif e DataInicio | Nif e DataInicio | Matricula e Nif |

**Exercício 2**

Comandos SQL para a criação das tabelas.

drop schema public cascade;

create schema public;

create table motorista(

Nome varchar(50),

NCartaCond varchar(50),

DataNasc date,

Nbi integer check(Nbi > 0),

primary key(Nbi)

);

create table telefone(

Nbi integer check(Nbi > 0),

Telefone integer,

primary key(Telefone, Nbi),

foreign key (Nbi) references motorista on delete restrict

);

create table modelo(

Marca varchar(50),

Modelo varchar(50),

Nlugares integer check(Nlugares > 0),

Consumo decimal check(Consumo > 0),

primary key(Modelo)

);

create table taxi(

Modelo varchar(50),

Ano integer check(ano > 1900),

Kms integer check(Kms > 0),

Matricula char(8),

primary key(Matricula),

foreign key (Modelo) references modelo on delete restrict

);

create table servico(

DataInicio timestamp,

DataFim timestamp,

Kms integer check(Kms > 0),

Valor decimal check(Valor > 0),

Matricula char(8),

CoordGPSInic decimal,

CoordGFPFim decimal,

foreign key (Matricula) references taxi on delete restrict,

primary key(Matricula, DataInicio)

);

create table turno(

DataInicio timestamp,

DataFim timestamp,

KmsInicio integer check(KmsInicio > 0),

KmsFim integer check(KmsFim > KmsInicio),

Matricula char(8),

Nbi integer check(Nbi > 0),

foreign key (Matricula) references taxi on delete restrict,

foreign key (Nbi) references motorista on delete restrict,

primary key(Nbi, DataInicio)

);

create table cliente(

Nome varchar(50),

Morada varchar(50),

CodigoPostal varchar(50),

Nif varchar(50),

primary key(Nif)

);

create table pedido(

Nif varchar(50),

MoradaInicio varchar(50),

CodigoPostalInicio varchar(50),

DataPedido timestamp,

Matricula char(8),

DataInicio timestamp,

foreign key (Matricula) references taxi on delete restrict,

foreign key (Nif) references cliente on delete restrict,

primary key(Nif, DataInicio)

);

**Exercício 3 e 4**

Expressões em SQL para inserir informação na base de dados.

/\*frota de taxis\*/

insert into modelo values ('Renault','Escape','7','7');

insert into taxi values('Escape','2015','123098','22-AA-22');

insert into modelo values ('Mercedes','CLK','7','9');

insert into taxi values('CLK','2014','234554','21-AA-22');

insert into modelo values ('Honda','Civic','5','5');

insert into taxi values('Civic','2012','89764','20-AA-22');

insert into modelo values ('Mercedes','Mercedes-Benz Classe S','5','6.5');

insert into taxi values('Mercedes-Benz Classe S','2015','79744','19-AA-22');

insert into modelo values('Audi','TT','5','6');

insert into taxi values('TT','2015','50000','25-XX-15');

insert into modelo values('Renault','Clio Break','5','7');

insert into taxi values('Clio Break','2015', '98756','13-QE-89');

/\*taxistas\*/

insert into motorista values('Manuel Duarte','L-123','1976-01-14','1234');

insert into telefone values('1234','266262626');

insert into telefone values('1234','939393939');

insert into motorista values('Fernando Nobre','L-124','1977-01-14','1235');

insert into telefone values('1235','266262627');

insert into telefone values('1235','939393940');

insert into motorista values('Anibal Silva','L-125','1978-01-14','1236');

insert into telefone values('1236','266262628');

insert into telefone values('1236','939393941');

insert into motorista values('Francisco Lopes','L-126','1979-01-14','1237');

insert into telefone values('1237','266262629');

insert into motorista values('Samuel Melo','L-520','1997-11-14','2220');

insert into telefone values('2220','962020202');

insert into motorista values('João Dias','L-535','1997-04-14','2550');

insert into telefone values('2550','982200364');

/\*clientes\*/

insert into cliente values('José Silva','Rua António Silva 23','7100-434 Évora',

'600700800900');

insert into cliente values('Francisco Passos','Rua Manuel Passos 12','7000-131 Évora',

'600700800901');

insert into cliente values('Pedro Sousa','Rua Joaquim Sousa 21','7500-313 Évora',

'600700800902');

insert into cliente values('Eduardo Romão','Rua Maria Leal 12','7000-120 Évora',

'600700800903');

/\*pedidos\*/

insert into pedido values('600700800900','Rua Silva Pais 33','7120-212 Évora','2016-01-02

09:00:00','19-AA-22','2016-01-02 09:43:00');

insert into pedido values('600700800900','Rua Silva Pais 33','7120-212 Évora','2016-01-02

15:00:00','13-QE-89','2016-01-02 15:43:00');

insert into pedido values('600700800903','Rua Paraiso 25','1222-232 Évora','2016-01-03

09:00:00','13-QE-89','2016-01-02 09:36:00');

/\*turnos\*/

insert into turno values('2016-01-02 08:00:00','2016-01-02 17:00:00','79744','79944',

'19-AA-22','1234');

insert into turno values('2016-01-02 08:00:00','2016-01-02 17:00:00','89764','89964',

'20-AA-22','1235');

insert into turno values('2016-01-03 08:00:00','2016-01-03 17:00:00','234554','234954',

'21-AA-22','1236');

insert into turno values('2016-01-18 08:00:00','2016-01-18 17:00:00','234554','234954',

'19-AA-22','1236');

insert into turno values('2016-01-03 08:00:00','2016-01-03 17:00:00','123098','123498',

'19-AA-22','1237');

insert into turno values('2016-01-02 07:00:00','2016-01-02 17:00:00','98756','99280',

'13-QE-89','2220');

insert into turno values('2016-01-02 09:00:00','2016-01-02 18:00:00','50000','50256',

'25-XX-15','2550');

insert into turno values('2016-01-03 09:00:00','2016-01-03 18:00:00','50000','50256',

'13-QE-89','2550');

insert into turno values('2016-01-04 09:00:00','2016-01-04 18:00:00','50000','50256',

'19-AA-22','2550');

insert into turno values('2016-01-05 09:00:00','2016-01-05 18:00:00','50000','50256',

'20-AA-22','2550');

insert into turno values('2016-01-06 09:00:00','2016-01-06 18:00:00','50000','50256',

'21-AA-22','2550');

insert into turno values('2016-01-07 09:00:00','2016-01-07 18:00:00','50000','50256',

'22-AA-22','2550');

insert into turno values('2016-01-08 09:00:00','2016-01-08 18:00:00','50000','50256',

'25-XX-15','1234');

insert into turno values('2016-01-09 09:00:00','2016-01-09 18:00:00','50000','50256',

'13-QE-89','1234');

insert into turno values('2016-01-10 09:00:00','2016-01-10 18:00:00','50000','50256',

'19-AA-22','1234');

insert into turno values('2016-01-11 09:00:00','2016-01-11 18:00:00','50000','50256',

'20-AA-22','1234');

insert into turno values('2016-01-12 09:00:00','2016-01-12 18:00:00','50000','50256',

'21-AA-22','1234');

insert into turno values('2016-01-13 09:00:00','2016-01-13 18:00:00','50000','50256',

'22-AA-22','1234');

/\*serviços\*/

insert into servico values('2016-01-02 08:12:00','2016-01-02 08:32:00','12','5.25',

'19-AA-22','0.75','0.76');

insert into servico values('2016-01-02 08:43:00','2016-01-02 08:52:00','7','3.25',

'19-AA-22','0.76','0.77');

insert into servico values('2016-01-02 08:53:00','2016-01-02 09:59:00','98','53.25',

'19-AA-22','0.78','1.56');

insert into servico values('2016-01-02 10:13:00','2016-01-02 10:29:00','18','19.25',

'19-AA-22','2.30','2.36');

insert into servico values('2016-01-02 11:10:00','2016-01-02 11:39:00','23','22.25',

'19-AA-22','3.00','3.43');

insert into servico values('2016-01-02 12:00:00','2016-01-02 13:39:00','21','42.25',

'19-AA-22','3.45','2.65');

insert into servico values('2016-01-02 15:20:00','2016-01-02 15:39:00','9','12.25',

'19-AA-22','2.06','1.96');

insert into servico values('2016-01-03 09:36:00','2016-01-03 10:39:00','12','16.25',

'13-QE-89','6.06','5.96');

insert into servico values('2016-01-02 11:03:00','2016-01-02 11:59:00','10','14.25',

'13-QE-89','5.56','5.16');

insert into servico values('2016-01-02 15:43:00','2016-01-02 16:59:00','20','34.25',

'13-QE-89','5.56','5.16');

insert into servico values('2016-01-02 09:43:00','2016-01-02 10:39:00','18','23.25',

'25-XX-15','7.06','5.42');

insert into servico values('2016-01-02 15:43:00','2016-01-02 16:02:00','10','13.25',

'25-XX-15','6.06','5.42');

**Exercício 5**

1. Quais as matrículas dos táxis da marca Mercedes?

* Álgebra Relacional:  
  Matricula(Marca=’Mercedes’(modelotaxi))
* SQL:

select Matricula   
from modelo natural inner join táxi  
where Marca='Mercedes';

1. Indique o nome dos motoristas que já fizeram um turno num táxi da marca Mercedes.

* Álgebra Relacional:

Nome(Marca=’Mercedes’(modelotaxiturnomotorista))

* SQL:

Select distinct( motorista.Nome)

from modelo natural inner join taxi natural inner join turno natural inner join motorista

where Marca='Mercedes';

1. Quais os telefones dos motoristas que já fizeram um serviço para satisfazer um pedido do Cliente com o Nif 600700800900?

* Álgebra Relacional:

Telefone(Nif=’600700800900’ pedido.Matricula≥turno.DataInicio pedido.DataInicio≤turno.DataFim (pedidoturnomotoristatelefone))

* SQL:

select Telefone

from pedido, turno natural inner join motorista natural inner join telefone

where Nif='600700800900' and pedido.Matricula = turno.Matricula and pedido.DataInicio >= turno.DataInicio and pedido.DataInicio <= turno.DataFim;

1. Quais os táxis que já foram conduzidos pelo ’Aníbal Silva’?

* Álgebra Relacional:

Marca,Modelo(Nome=’Anibal Silva’(modelotaxiturnomotorista))

* SQL:

select Marca,Modelo

from motorista natural inner join turno natural inner join taxi natural inner join modelo

where Nome='Anibal Silva';

1. Quais os nomes dos motoristas que nunca fizeram um serviço para satisfazer um pedido do Sr. José Silva?

* Álgebra Relacional:

Nome(motorista) – Motorista.Nome(cliente=’José Silva’ pedido.Matricula≥turno.DataInicio pedido.DataInicio≤turno.DataFim (cliente pedido turnomotorista))~

* SQL:

(select Nome from motorista)

except

(select motorista.Nome

from cliente natural inner join pedido,turno natural inner join motorista

where cliente.Nome = 'José Silva'and pedido.Matricula = turno.Matricula and pedido.DataInicio >= turno.DataInicio and pedido.DataInicio <= turno.DataFim);

* Explicação:

Seleciona-se em primeiro o nome de todos os motoristas e em seguida através de um except vamos retirar todos os motoristas que satisfizeram um pedido do Sr. José Silva.

1. Quais os motoristas que nunca fizeram um turno num táxi da marca

Mercedes?

* Álgebra Relacional:

Nome(motorista) – Motorista.Nome(Marca=’Mercedes’ (modelotaxi turnomotorista))

* SQL:

(select Nome from motorista)

except

(select motorista.Nome

from modelo natural inner join taxi natural inner join turno natural inner join motorista

where Marca='Mercedes');

* Explicação:

Usando a mesma logica que o exercício da alinha anterior resolvemos o exercício.

1. Quais os motoristas que já fizeram serviços em todos os táxis?

* Álgebra Relacional:

Nome,Matricula(taxiturnomotorista) ÷ Matricula(taxi)

* SQL:

select tabela.Nome

from

(select Nome, count(distinct(Matricula)) as NUM

from turno natural inner join motorista

group by Nome) as tabela

where

NUM=(select count(Matricula)from taxi);

1. Para cada motorista indique o número total de serviços em todos os turnos.

* Álgebra Relacional:

motorista.Nome count(\*)(turno.DataInicio≤servico.DataInicio servico.DataInicio≤turno.DataFim  turno.Matricula=serviço.Matricula(motoristaturno, servico))

* SQL:

select motorista.Nome, count(\*)

from motorista natural inner join turno, servico

where turno.DataInicio<=servico.DataInicio and

turno.DataFim>=servico.DataInicio and

turno.Matricula = servico.Matricula

group by motorista.Nome;

* Explicação:

Nesta alínea e nas restantes nunca podemos fazer inner join entre os atributos turno e serviço pois iriamos obter uma tabela apenas com os serviços que começassem e acabassem à mesma hora do turno (pois ambos têm um atributo “datainicio” e “datafim”).

1. Para cada motorista indique o total ganho em todos os serviços que fez.

* Álgebra Relacional:

motorista.Nome sum(Valor)(turno.DataInicio≤servico.DataInicio servico.DataInicio≤turno.DataFim  turno.Matricula=serviço.Matricula(motoristaturno, servico))

* SQL:

select motorista.Nome, sum(valor)

from motorista natural inner join turno, servico

where turno.DataInicio <= servico.DataInicio and

turno.DataFim >= servico.DataInicio and

turno.Matricula = servico.Matricula

group by motorista.Nome;

1. Indique o nome do motorista que fez o turno mais lucrativo.

* Álgebra Relacional:

tabela1 🡨 motorista.Nome sum(Valor) as Soma (turno.DataInicio≤servico.DataInicio servico.DataInicio≤turno.DataFim  turno.Matricula=serviço.Matricula(motoristaturno,servico))

Nome(tabela1(motorista.Nome max(tabela1.Soma)(tabela1)

* SQL:

select Nome

from (select motorista.Nome, sum(valor) as Soma

from motorista natural inner join turno,

servico

where turno.DataInicio <= servico.DataInicio and

turno.DataFim >= servico.DataInicio and

turno.Matricula = servico.Matricula

group by motorista.Nome) as tabela1,

(select max(tabela2.Soma) as Soma2

from (select motorista.Nome, sum(valor) as Soma

from motorista natural inner join turno,

servico

where turno.DataInicio <= servico.DataInicio and

turno.DataFim >= servico.DataInicio and

turno.Matricula = servico.Matricula

group by motorista.Nome) as tabela2) as tabela3

where tabela1.Soma=tabela3.Soma2;

* Explicação:

Para encontrar o nome do motorista recorremos a duas tabelas. A primeira tabela fornece-nos o valor total que cada motorista fez. Enquanto a segunda recorre à primeira e procura o valor máximo da mesma. Intersectando as duas tabelas obtemos o nome do motorista que realizou o turno mais lucrativo.

1. Indique a matricula, a marca e modelo do táxi que fez o turno com mais quilómetros percorridos.

* Álgebra Relacional:

tabela1 🡨Matricula, (KmsFim-KmsInicio) as Dif (turno)

táxi.Matricula,Marca,táxi.Modelo(tabela1( max(tabela1.Dif)(tabela1)) modelotaxi)

* SQL:

select taxi.Matricula, Marca, taxi.Modelo

from (select Matricula, KmsFim-KmsInicio as Dif

from turno) as tabela1,

(select max(tabela2.Dif) as Max

from (select Matricula, KmsFim-KmsInicio as Dif

from turno) as tabela2) as tabela3,

modelo, taxi

where taxi.Matricula = tabela1.Matricula and

taxi.modelo = modelo.Modelo and tabela1.Dif = tabela3.Max;

* Explicação:

Usando a mesma logica que o exercício da alinha anterior resolvemos a query usando duas tabelas.

1. Indique o tempo médio de espera para cada pedido à companhia.

* Álgebra Relacional:

avg(DataInicio-DataPedido)(pedido)

* SQL:

select avg(cast(DataInicio as time)-cast(DataPedido as time))

from pedido;

* Explicação:

A partir do cast conseguimos obter o tempo do timestamp, o qual fomos subtrair as horas obtendo uma tabela com o tempo de espera. Recorrendo ao avg fizemos o tempo médio dessa mesma tabela.

1. Indique o nome do cliente que fez mais pedidos.

* Álgebra Relacional:

tabela1🡨 Nome count(Nif) as N (clientepedido)

Nome (tabela1( max(tabela1.N) (tabela1)))

* SQL:

select nome

from (select nome, count(nif) as N

from cliente natural inner join pedido

group by nome) as tabela1,

(select max(tabela2.N) as H

from (select nome, count(nif) as N

from cliente natural inner join pedido

group by nome) as tabela2) as tabela3

where tabela1.N = tabela3.H;

* Explicação:

Resolveu-se esta query utilizando a mesma logica que as alíneas j e k.

1. Indique o táxi, matricula marca e modelo, mais lucrativo da companhia (o que rendeu mais dinheiro por km percorrido, não se esqueça de retirar o gasto, o consumo do veiculo).

* Álgebra Relacional:

tabela1 🡨 Matricula ((sum(KmsFim-KmsInicio)\*Consumo/100)\*1,22) as Combustivel (modelotaxiturno)

tabela2 🡨 Matricula (sum(Valor)-tabela1.Combustivel) as Dinheiro (tabela1)

BigPart 🡨 Matricula (tabela2.Dinehiro/sum(KmsFim-KmsInicio)) as Num (tabela2turno)

BigPart3 🡨 Matricula (max(BigPart.Num)) (BigPart)

BigPart.Matricula,Marca,Modelo (BigPartBigPart3taximodelo)

* SQL:

select BigPart.Matricula, Marca, Modelo

from

(select matricula, (tabela2.dinheiro / sum(kmsfim-kmsinicio)) as NUM

from turno natural inner join (select Matricula, (sum(Valor)-tabela1.combustivel) as dinheiro

from servico natural inner join (select taxi.Matricula, ((sum(kmsfim - kmsinicio)\*consumo/100)\*1.22) as combustivel

from turno natural inner join modelo natural inner join taxi

group by taxi.Matricula,consumo) as tabela1

group by Matricula, tabela1.combustivel) as tabela2

group by matricula, tabela2.dinheiro) as BigPart

,

(select max(BigPart2.NUM) as MAX from (select matricula, (tabela2.dinheiro / sum(kmsfim-kmsinicio))as NUM

from turno natural inner join (select Matricula, (sum(Valor)-tabela1.combustivel) as dinheiro

from servico natural inner join (select taxi.Matricula, ((sum(kmsfim - kmsinicio)\*consumo/100)\*1.22) as combustivel

from turno natural inner join modelo natural inner join taxi

group by taxi.Matricula,consumo) as tabela1

group by Matricula, tabela1.combustivel) as tabela2

group by matricula, tabela2.dinheiro) as BigPart2) as BigPart3, taxi natural inner join modelo

where BigPart.NUM = BigPart3.MAX and taxi.Matricula = BigPart.Matricula;

* Explicação:

Para chegar à conclusão de qual foi o táxi que obteve maior lucro, começámos por criar uma tabela que nos informasse acerca da quantidade de combustível gasta pelo veículo, para que depois fosse possível obter o produto dessa mesma quantidade pelo respetivo preço por litro de combustível. De seguida, do valor monetário total contabilizado em serviço foi retirado a quantidade monetária gasta em combustível. Dividindo o lucro ou prejuízo pelos quilómetros percorridos foi obtido o rendimento por quilometro percorrido. Por fim, apenas se tornou necessário procurar o táxi ao qual estava associado o maior rendimento.

1. Para cada motorista indique o número de kms percorridos sem ser em serviço em todos os turnos.

* Álgebra Relacional:

Nome (KmsFim-KmsInicio-sum(Kms))(turno.DataInicio≤servico.DataInicio servico.DataFim≤turno.DataFim  turno.Matricula=serviço.Matricula(motoristaturno, servico))

* SQL:

select nome, KmsFim-KmsInicio-sum(Kms)

from servico, turno natural inner join motorista

where servico.datainicio > turno.datainicio and servico.datafim < turno.datafim and servico.matricula=turno.matricula

group by nome, KmsFim, KmsInicio;

1. Indique o nome do motorista fez o serviço mais rápido, percorreu mais km em menos tempo.

* Álgebra Relacional:

tabela1 🡨 Nome ((serviço.DataFim-Servico.DataInicio)/Kms) as Km (turno.DataInicio≤servico.DataInicio servico.DataFim≤turno.DataFim  turno.Matricula=serviço.Matricula(motoristaturno, servico))

Nome(tabela1( min(tabela1.Km) (tabela1))

* SQL:

select nome

from

(select nome, ((cast(servico.DataFim as time)-cast(servico.DataInicio as time))/Kms) as KM

from servico, turno natural inner join motorista

where servico.datainicio>turno.datainicio and servico.datafim<turno.datafim and servico.matricula=turno.matricula

group by nome, Kms, servico.DataFim, servico.DataInicio) as tabela1,

(select min(tabela2.KM) as MIN

from (select nome, ((cast(servico.DataFim as time)-cast(servico.DataInicio as time))/Kms) as KM

from servico, turno natural inner join motorista

where servico.datainicio>turno.datainicio and servico.datafim<turno.datafim and servico.matricula=turno.matricula

group by nome, Kms, servico.DataFim, servico.DataInicio) as tabela2) as tabela3

where tabela1.KM=tabela3.MIN;

* Explicação:

Para descobrir o nome do motorista que realizou o serviço mais rápido foi decidido verificar qual deles fazia um quilómetro no menor período de tempo. Para tal, dividimos o tempo da viagem pelo número de quilómetros percorridos, obtendo assim o valor desejado. Por fim, sabendo os valores restou-nos procurar o menor, utilizando o mesmo método aplicado nos exercícios de máximo.