Mestrado em Inteligência Artificial e Sistemas Inteligentes

Bases de Dados e Programação

24 de Janeiro de 2005

Trabalho de Bases de Dados

Empresa Rodoviária

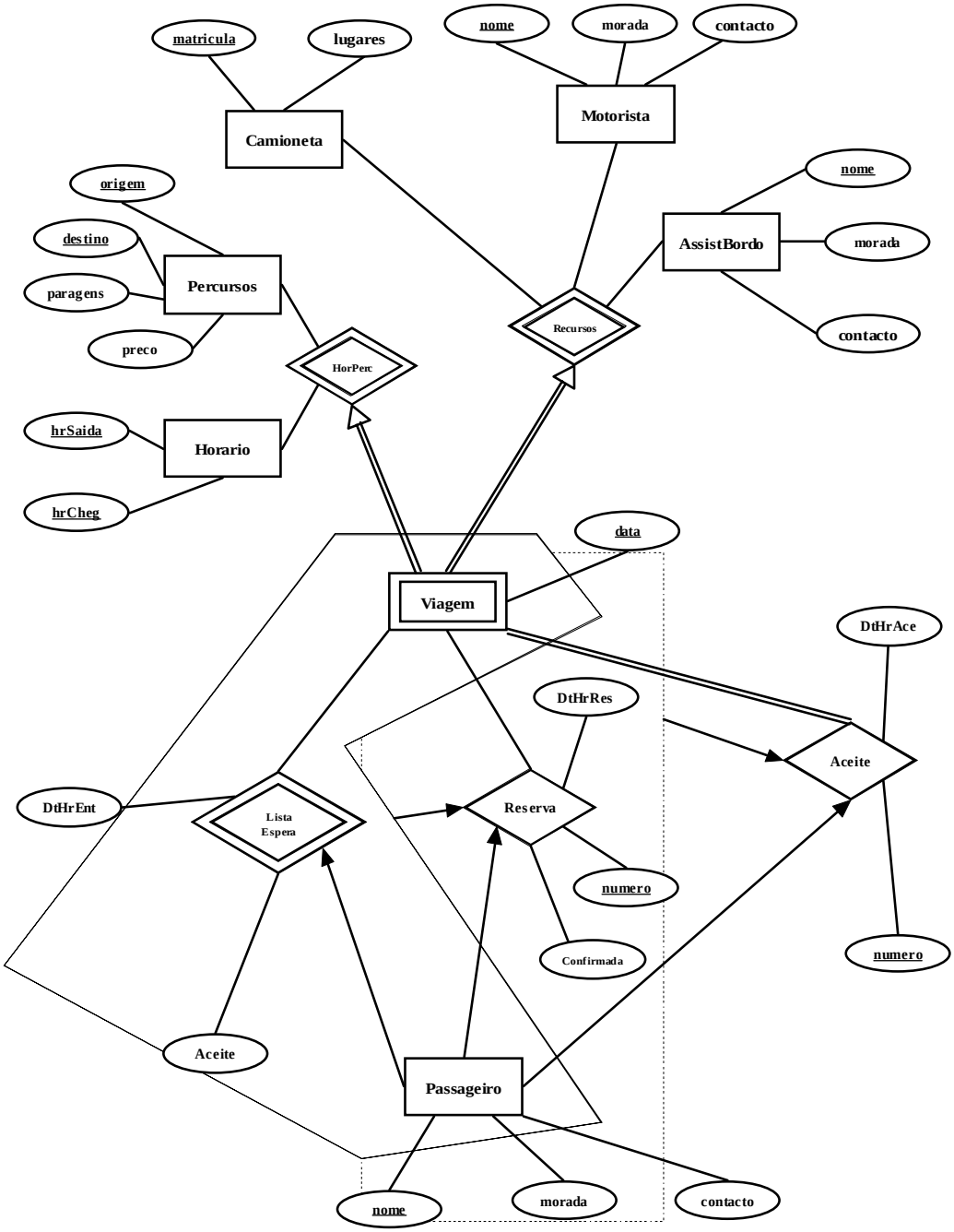
António Jesus Monteiro de Castro

Aluno 040594004

Faculdade de Engenharia da Universidade do Porto

ÍNDICE

Diagrama de Entidades e Relacionamentos



Notas:

(1) Da leitura que fiz parte do princípio que uma viagem corresponde a um só percurso. No entanto, numa situação real, teria de esclarecer se é verdade ou não, pois poderia dar-se o caso de uma viagem poder englobar mais do que um percurso.

(2) A relação [ACEITE] contém os passageiros que PAGARAM e que, como tal, estão aceites como passageiros da viagem. A venda de bilhetes normais (conforme enunciado) é realizada por uma inclusão em [ACEITE] pois pressupõe um pagamento.

(3) A passagem da lista da espera para a reserva é dada pela agregação acima nas condições indicadas no enunciado.

(4) Tal como em (3) a passagem da reserva para passageiro aceite é dada pela agregação acima nas condições do enunciado.

(5) De notar que parti do princípio (é um princípio discutível) de que não há "passagens" directas de lista de espera para passageito aceite, ou seja, um passageiro em lista de espera que esteja em condições de ser aceite será necessário em primeiro lugar criar a reserva e, depois, ser transformada em aceite.

(6) A ordem dos passageiros na lista de espera é garantida pela data e hora de entrada na lista.

Esquemas Relacionais na 3NF

Os esquemas relacionais tirados do ER anteriores são os seguintes:

Camioneta(matricula:texto, lugares:inteiro)
Motorista(nome:texto, morada:texto, contacto:texto)
AssistBordo(nome:texto, morada:texto, contacto:texto)
Percurso(origem:texto, destino:texto, paragens:texto, preco:decimal)
Horario(hrsaida:time, hrcheg:time)
Passageiro(nome:texto, morada:texto, contacto:texto)
HorPerc(origem:texto, destino:texto, hrsaida:time, hrcheg:time)
Recurso(matricula:texto, nomemot:texto, nomeass:texto)
ListaEspera(nomepass:texto, CHAVE[viagem],dthrent:datetime, aceite:boolean)
Reserva(numero:inteiro, nomepass:texto,CHAVE[viagem], dthrres:datetime, confirmada:boolean)
Aceite(numero:inteiro, nomepass:texto,CHAVE[viagem], dthrace:datetime)
Viagem(data:datetime, CHAVE[horperc], CHAVE[recurso], numeroaceite:inteiro)

Para facilitar a leitura deste esquema, utilizei a expressão CHAVE[nome entidade ou relação] que significa substituir pelos atributos chave da entidade/relação referida.
Nesta fase do trabalho vou "fugir" um pouco à metodologia ensinada nas aulas, embora vá justificar a forma de o fazer. No entanto e se o fizesse seguindo a metodologia ensinada nas aulas, agora teria que fazer o seguinte para que o esquema relacional estivesse na 3NF:

1) Identificar as dependências funcionais para cada uma das relações.
2) Verificar/identificar as chaves das relações e corrigir se necessário.
3) Para todas as dependências funcionais não trivais (uma DF A1A2..An->B1B2..Bn é não trivial se pelo menos um dos B's não pertence aos A's), verificar se violam ou não a BCNF, ou seja, os A's duma DF são uma superchave (um conjunto de atributos que contenham uma chave) da relação.
4) Se violassem a BCNF seria preciso decompor a relação. Isso seria feito decompondo cada relação R em R1 e R2 sendo R1=R-{B's da DF] e R2={A's da DF}.
5) Finalmente, se houvessem dependências não preservadas, seria necessário acrescentá-las através duma nova relação R3={A's da DF e B's da DF sendo os A's as chaves}.

Se fizesse correctamente os passos anteriores, obteria o esquema relacional pedido na 3NF.

Antes de apresentar a forma diferente de obter o esquema relacional, gostaria de salientar que o meu objectivo não é "atalhar" a questão de forma a simplificar a realização do trabalho. O que vou apresentar em seguida reflecte a minha forma de tratar estas questões no meu dia a dia profissional e incorpora alguns aspectos que têm sido muito úteis e evitam alguns dos problemas que, depois, na prática acontecem.

Chave única com um único atributo para cada entidade/relação

O que vou fazer é aquilo que em alguma documentação é referida como Chaves Artificais (*Artificial Keys*). Normalmente, os puristas do modelo relacional não são adeptos da utilização destas chaves porque elas nada dizem aos restantes atributos da entidade/relação. Básicamente funcionam como apontadores de/para outras entidades/relações. De qualquer das formas posso dizer que a tenho utilizado em todo o trabalho que desenvolvo e posso garantir que resolve mais problemas do que aqueles que cria. Normalmente isso é feito através de um campo numérico e incremental (*Identity*) ou, então, usando um atributo alfanumérico com um valor que se garanta que nunca se repetirá (normalmente é gerado um valor GUID). Assim, para cada tabela vou acrescentar um atributo com um nome da seguinte forma: nome_entidade + "ID".

Camioneta(camionetaID:identity, *matricula*:texto, lugares:inteiro)
Motorista(motoristaID:identity, nome:texto, morada:texto, contacto:texto) (3)
AssistBordo(assistbordoID:identity, nome:texto, morada:texto, contacto:texto) (3)
Percurso(percursoID:identity, *origem*:texto, *destino*:texto, paragens:texto, preco:decimal)
Horario(horarioID:incremental, *hrsaida*:time, *hrcheg*:time)
Passageiro(passageiroID:incremental, nome:texto, morada:texto, contacto:texto) (3)
HorPerc(horpercID:incremental, *percursoID*:inteiro, *horarioID*:inteiro)
Recurso(recursoID:incremental, *camionetaID*:inteiro, *motoristaID*:inteiro, *assistbordoID*:inteiro)
ListaEspera(listaesperaID:identity, *passageiroID*:inteiro, *viagemID*:inteiro, dthrent:datetime, aceite:boolean)
Reserva(reservaID:identity, *passageiroID*:inteiro, *viagemID*:inteiro, dthrres:datetime, confirmada:boolean)
Aceite(aceiteID:identity, *passageiroID*:inteiro, *viagemID*:inteiro, dthrace:datetime)
Viagem(viagemID:identity, *data*:datetime, *horpercID*:inteiro, *recursoID*:inteiro)

Notas:

(1) nas relações Reserva e Aceite em que existia um campo chave número, o mesmo foi retirado pois os campos reservaID e aceiteID servem o mesmo objectivo (assumindo que, para o cliente, servia um número sequencial para identificar a reserva e a aceitação. Se não fosse o caso, então ter-se-ia que manter um campo adicional que significasse a reserva e a aceitação, respectivamente.

(2) As chaves alternativas estão assinaladas a itálico dentro da relação. Ver também o texto sobre a 1NF abaixo onde explico o que são e porque as uso.

(3) Nesta relação poderá ser discutível a utilização ou não de chaves alternativas. Se quisermos garantir que, de facto, não se regista o mesmo motorista duas ou mais vezes, os atributos nome e morada poderiam ser candidatos. Somente o nome não chegaria porque poderia haver dois motoristas com o mesmo nome. Não assinalei nenhuma chave alternativa porque campos texto desta natureza nunca são bons candidatos para chaves únicas. Basta alguém escrever o nome de forma diferente e lá se vai a unicidade.

(4) Convém referir que no esquema relacional coloquei logo as chaves artificiais. No entanto a utilização das chaves artificiais ou das chaves alternativas como chave primária da entidade/relação é uma decisão que costuma ser tomada quando se faz o desenho físico da base de dados.

(5) Para mim, uma das grandes vantagens da utilização das chaves artificiais verifica-se quando se pretende alterar o valor de um atributo e esse atributo é um campo chave. Se não se utilizasse uma chave artificial seria necessário garantir que todas as entidades onde esse atributo é referido fossem devidamente actualizadas. Embora as bases de dados tenham mecanismos para o fazer existe sempre um preço a pagar quanto mais não seja em termos de processamento. Se se usar chaves artificiais o problema está resolvido uma vez que elas funcionam como apontadores.

Vamos então verificar se a BCNF bem como a 3NF são respeitadas:

1NF

Três condições:

a) Todos os atributos têm de ser atómicos, ou seja, somente um valor representado num atributo numa única instância da entidade.
b) Todas as instâncias da entidade têm o mesmo número de valores.
c) Todas as instâncias da entidade são diferentes.

Com a inclusão do atributo *Identity* a 1NF está garantida. De qualquer das formas aceito a crítica de que, por exemplo, no caso da Percurso, aparentemente nada impede de estando formalmente garantida a 1NF não termos duas instâncias (excluindo o *Identity*) com os mesmos valores. Veja-se por exemplo: Percurso(1, Porto, Lisboa, Coimbra, 15) e Percurso(2, Porto, Lisboa, Coimbra, 15). Estas instâncias seriam aceites com o esquema aqui representado mas é uma situação que não é aceitável. Isto resolve-se criando como chave alternativa única uma chave composta pelos dois atributos Origem e Destino. Logicamente que vai haver uma dependência funcional entre a chave primária e a chave alternativa, ou seja, a chave primária determinará funcional a chave alternativa e vice-versa. Esta é uma operação que seria feita ao verificarmos a BCNF como veremos mais à frente.

2NF

Duas condições:

a) Tem de estar na 1NF.
b) Cada atributo deve ser um facto descrevendo toda a chave.

Normalmente a condição b) é relevante quando estamos perante uma chave primária composta, o que não é o caso aqui devido à técnica usada (embora na definição das chaves alternativas, tenho sido levado em conta).

3NF

Duas condições:

a) Tem de estar na 2NF.
b) Não pode ter atributos não chave que sejam factos acerca de outro atributo não chave. Dito de outra forma, todos os atributos devem ser factos sobre a chave e nada mais do que a chave. Na prática significa ver as relações dos atributos uns com os outros.

Também me parece que está salvaguardada esta forma.

BCNF

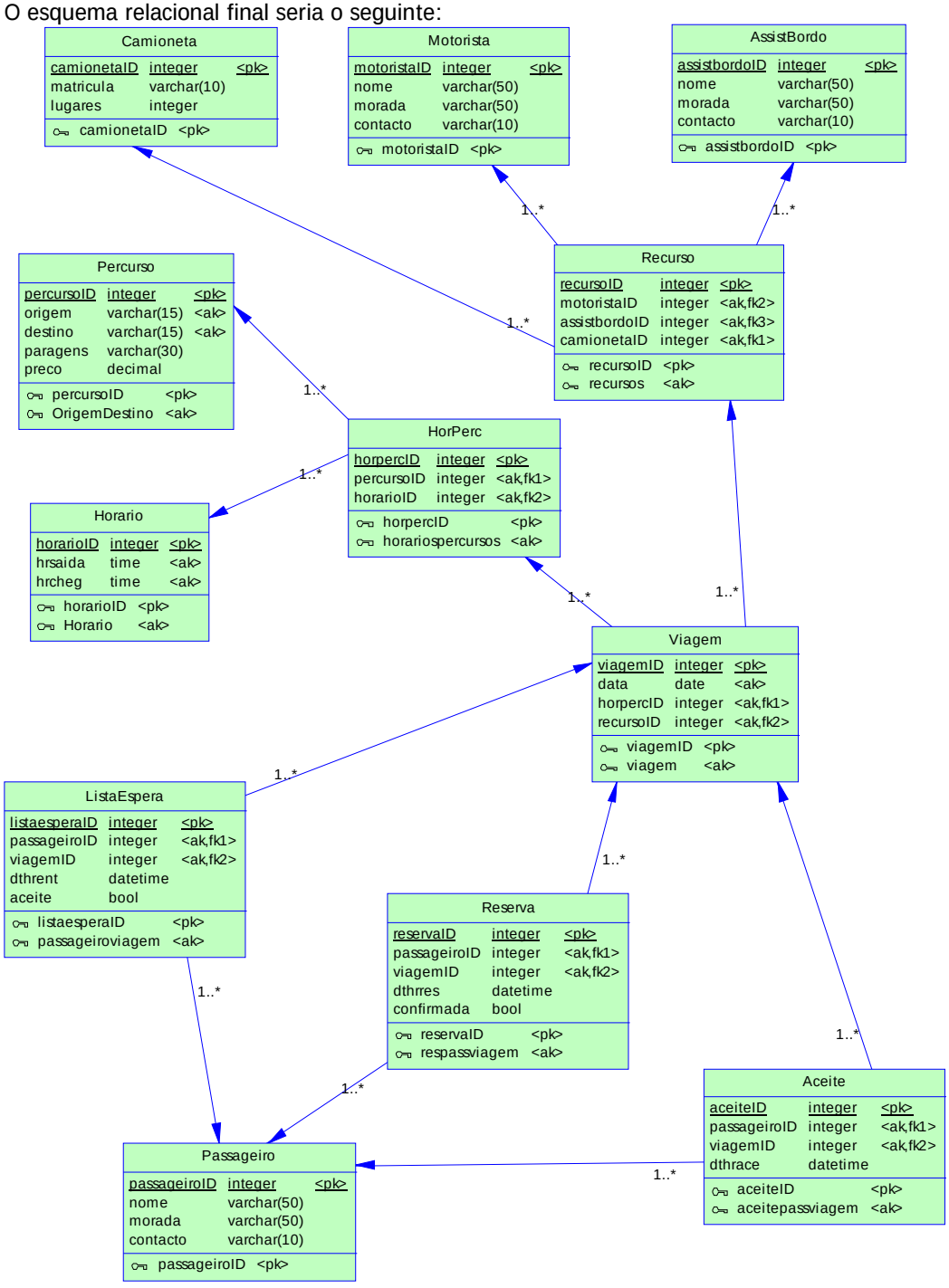
Duas condições:

a) Todos os atributos dependem da chave.
b) Todos os determinantes (ie, qualquer atributo ou combinação de atributos sobre os quais qualquer atributo ou combinação de atributos sejam dependentes funcionalmente)forem uma chave.

No esquema relacional apresentado acima está garantida a BCNF em ambas as condições. A primeira pela inclusão das chaves artificiais e a segunda pela colocação das chaves alternativas.

Diagrama com o esquema relacional final

O esquema relacional final seria o seguinte:



Notas:

(1) Utilizei um diagrama semelhante aos utilizados para se descrever o modelo físico da base de dados.

Instruções SQL para criar as relações

As instruções que aqui coloco foram usadas para criar a base de dados no MySQL. Por isso é natural que tenha instruções específicas deste motor de base de dados.

```
/*==============================================================*/
/* Database name:  BDP                                          */
/* DBMS name:      MySQL 4.0                                    */
/* Created on:     23-01-2005 22:52:02                          */
/*==============================================================*/


use BDP;

drop database if exists BDP;

/*==============================================================*/
/* Database: BDP                                                */
/*==============================================================*/
create database BDP;

use BDP;

/*==============================================================*/
/* Table: ACEITE                                                */
/*==============================================================*/
create table ACEITE
(
   ACEITEID             integer              not null,
   PASSAGEIROID            integer              not null,
   VIAGEMID             integer              not null,
   DTHRACE              datetime,
   primary key (ACEITEID),
   unique AK_ACEITEPASSVIAGEM (PASSAGEIROID, VIAGEMID)
)
type = InnoDB;

/*==============================================================*/
/* Index: "ACEITEPASSAGEIRO_FK"                                 */
/*==============================================================*/
create index ACEITEPASSAGEIRO_FK
(
   PASSAGEIROID
);
/*==============================================================*/
/* Index: "ACEITEVIAGEM_FK"                                     */
/*==============================================================*/
create index ACEITEVIAGEM_FK
(
   VIAGEMID
);

/*==============================================================*/
/* Index: ACEITEID                                              */
/*==============================================================*/
create unique index ACEITEID on ACEITE
(
   ACEITEID
);

/*==============================================================*/
/* Index: ACEITE_AK                                             */
/*==============================================================*/
create index ACEITE_AK on ACEITE
(
   PASSAGEIROID,
   VIAGEMID
);
```

```
/*==============================================================*/
/* Table: ASSISTBORDO                                           */
/*==============================================================*/
create table ASSISTBORDO
(
   ASSISTBORDOID              integer                 not null,
   NOME                       varchar(50),
   MORADA                     varchar(50),
   CONTACTO                   varchar(10),
   primary key (ASSISTBORDOID)
)
type = InnoDB;

/*==============================================================*/
/* Index: ASSISTBORDO_PK                                        */
/*==============================================================*/
create unique index ASSISTBORDO_PK on ASSISTBORDO
(
   ASSISTBORDOID
);

/*==============================================================*/
/* Table: CAMIONETA                                             */
/*==============================================================*/
create table CAMIONETA
(
   CAMIONETAID                integer                 not null,
   MATRICULA                  varchar(10),
   LUGARES                    integer,
   primary key (CAMIONETAID)
)
type = InnoDB;

/*==============================================================*/
/* Table: HORARIO                                               */
/*==============================================================*/
create table HORARIO
(
   HORARIOID                  integer                 not null,
   HRSAIDA                    time                    not null,
   HRCHEG                     time                    not null,
   primary key (HORARIOID),
   unique AK_HORARIO (HRSAIDA, HRCHEG)
)
type = InnoDB;

/*==============================================================*/
/* Index: HORARIOID                                             */
/*==============================================================*/
create unique index HORARIOID on HORARIO
(
   HORARIOID
);

/*==============================================================*/
/* Index: HORARIO_AK                                            */
/*==============================================================*/
create unique index HORARIO_AK on HORARIO
(
   HRSAIDA,
   HRCHEG
);

/*==============================================================*/
/* Table: HORPERC                                               */
/*==============================================================*/
create table HORPERC
(
   HORPERCID                  integer                 not null,
   PERCURSOID                 integer                 not null,
   HORARIOID                  integer                 not null,
   primary key (HORPERCID),
```

```
  unique AK_HORARIOSPERCURSOS (PERCURSOID, HORARIOID)
)
type = InnoDB;

/*==============================================================*/
/* Index: "HORPERCPERCURSO_FK"                                  */
/*==============================================================*/
create index HORPERCPERCURSO_FK
(
  PERCURSOID
);
/*==============================================================*/
/* Index: "HORPERCHORARIO_FK"                                   */
/*==============================================================*/
create index HORPERCHORARIO_FK
(
  HORARIOID
);

/*==============================================================*/
/* Index: HORPERCID                                             */
/*==============================================================*/
create unique index HORPERCID on HORPERC
(
  HORPERCID
);

/*==============================================================*/
/* Index: HORPERC_AK                                            */
/*==============================================================*/
create index HORPERC_AK on HORPERC
(
  PERCURSOID,
  HORARIOID
);

/*==============================================================*/
/* Table: LISTAESPERA                                           */
/*==============================================================*/
create table LISTAESPERA
(
  LISTAESPERAID          integer              not null,
  PASSAGEIROID           integer              not null,
  VIAGEMID               integer              not null,
  DTHRENT                datetime,
  ACEITE                 bool,
  primary key (LISTAESPERAID),
  unique AK_PASSAGEIROVIAGEM (PASSAGEIROID, VIAGEMID)
)
type = InnoDB;

/*==============================================================*/
/* Index: "LISTAESPERAPASSAGEIRO_FK"                            */
/*==============================================================*/
create index LISTAESPERAPASSAGEIRO_FK
(
  PASSAGEIROID
);
/*==============================================================*/
/* Index: "LISTAESPERAVIAGEM_FK"                                */
/*==============================================================*/
create index LISTAESPERAVIAGEM_FK
(
  VIAGEMID
);

/*==============================================================*/
/* Index: LISTAESPERAID                                         */
/*==============================================================*/
create unique index LISTAESPERAID on LISTAESPERA
(
  LISTAESPERAID
```

```
);

/*==============================================================*/
/* Index: LISTAESPERA_AK                                        */
/*==============================================================*/
create index LISTAESPERA_AK on LISTAESPERA
(
   PASSAGEIROID,
   VIAGEMID
);

/*==============================================================*/
/* Table: MOTORISTA                                             */
/*==============================================================*/
create table MOTORISTA
(
   MOTORISTAID          integer              not null,
   NOME                 varchar(50),
   MORADA               varchar(50),
   CONTACTO             varchar(10),
   primary key (MOTORISTAID)
)
type = InnoDB;

/*==============================================================*/
/* Index: MOTORISTA_PK                                          */
/*==============================================================*/
create unique index MOTORISTA_PK on MOTORISTA
(
   MOTORISTAID
);

/*==============================================================*/
/* Table: PASSAGEIRO                                            */
/*==============================================================*/
create table PASSAGEIRO
(
   PASSAGEIROID         integer              not null,
   NOME                 varchar(50),
   MORADA               varchar(50),
   CONTACTO             varchar(10),
   primary key (PASSAGEIROID)
)
type = InnoDB;

/*==============================================================*/
/* Index: PASSAGEIROID                                          */
/*==============================================================*/
create unique index PASSAGEIROID on PASSAGEIRO
(
   PASSAGEIROID
);

/*==============================================================*/
/* Table: PERCURSO                                              */
/*==============================================================*/
create table PERCURSO
(
   PERCURSOID           integer              not null,
   ORIGEM               varchar(15)          not null,
   DESTINO              varchar(15)          not null,
   PARAGENS             varchar(30),
   PRECO                decimal,
   primary key (PERCURSOID),
   unique AK_ORIGEMDESTINO (ORIGEM, DESTINO)
)
type = InnoDB;

/*==============================================================*/
/* Index: PERCURSO_PK                                           */
/*==============================================================*/
create unique index PERCURSO_PK on PERCURSO
```

```
(
  PERCURSOID
);

/*==============================================================*/
/* Index: ORIGEMDESTINO_AK                                      */
/*==============================================================*/
create unique index ORIGEMDESTINO_AK on PERCURSO
(
  ORIGEM,
  DESTINO
);

/*==============================================================*/
/* Table: RECURSO                                               */
/*==============================================================*/
create table RECURSO
(
  RECURSOID                integer                not null,
  MOTORISTAID              integer                not null,
  ASSISTBORDOID            integer                not null,
  CAMIONETAID              integer                not null,
  primary key (RECURSOID),
  unique AK_RECURSOS (MOTORISTAID, ASSISTBORDOID, CAMIONETAID)
)
type = InnoDB;

/*==============================================================*/
/* Index: "RECURSOCAMIONETA_FK"                                 */
/*==============================================================*/
create index RECURSOCAMIONETA_FK
(
  CAMIONETAID
);
/*==============================================================*/
/* Index: "RECURSOMOTORISTA_FK"                                 */
/*==============================================================*/
create index RECURSOMOTORISTA_FK
(
  MOTORISTAID
);
/*==============================================================*/
/* Index: "RECURSOASSISTBORDO_FK"                               */
/*==============================================================*/
create index RECURSOASSISTBORDO_FK
(
  ASSISTBORDOID
);

/*==============================================================*/
/* Index: RECURSOID                                             */
/*==============================================================*/
create unique index RECURSOID on RECURSO
(
  RECURSOID
);

/*==============================================================*/
/* Index: RECURSO_AK                                            */
/*==============================================================*/
create unique index RECURSO_AK on RECURSO
(
  MOTORISTAID,
  ASSISTBORDOID,
  CAMIONETAID
);

/*==============================================================*/
/* Table: RESERVA                                               */
/*==============================================================*/
create table RESERVA
(
```

```
  RESERVAID                integer                 not null,
  PASSAGEIROID             integer                 not null,
  VIAGEMID                 integer                 not null,
  DTHRRES                  datetime,
  CONFIRMADA               bool,
  primary key (RESERVAID),
  unique AK_RESPASSVIAGEM (PASSAGEIROID, VIAGEMID)
)
type = InnoDB;

/*==============================================================*/
/* Index: "RESERVAPASSAGEIRO_FK"                                */
/*==============================================================*/
create index RESERVAPASSAGEIRO_FK
(
  PASSAGEIROID
);
/*==============================================================*/
/* Index: "RESERVAVIAGEM_FK"                                    */
/*==============================================================*/
create index RESERVAVIAGEM_FK
(
  VIAGEMID
);

/*==============================================================*/
/* Index: RESERVAID                                             */
/*==============================================================*/
create unique index RESERVAID on RESERVA
(
  RESERVAID
);

/*==============================================================*/
/* Index: RESERVA_AK                                            */
/*==============================================================*/
create index RESERVA_AK on RESERVA
(
  PASSAGEIROID,
  VIAGEMID
);

/*==============================================================*/
/* Table: VIAGEM                                                */
/*==============================================================*/
create table VIAGEM
(
  VIAGEMID                 integer                 not null,
  DATA                     date                    not null,
  HORPERCID                integer                 not null,
  RECURSOID                integer                 not null,
  primary key (VIAGEMID),
  key AK_VIAGEM (DATA, HORPERCID, RECURSOID)
)
type = InnoDB;

/*==============================================================*/
/* Index: "VIAGEMHORPERC_FK"                                    */
/*==============================================================*/
create index VIAGEMHORPERC_FK
(
  HORPERCID
);
/*==============================================================*/
/* Index: "VIAGEMRECURSO_FK"                                    */
/*==============================================================*/
create index VIAGEMRECURSO_FK
(
  RECURSOID
);

/*==============================================================*/
```

```
/* Index: VIAGEMID                                */
/*==============================================================*/
create unique index VIAGEMID on VIAGEM
(
   VIAGEMID
);

/*==============================================================*/
/* Index: VIAGEM_AK                               */
/*==============================================================*/
create index VIAGEM_AK on VIAGEM
(
   DATA,
   HORPERCID,
   RECURSOID
);

alter table ACEITE add constraint FK_ACEITEPASSAGEIRO foreign key (PASSAGEIROID)
      references PASSAGEIRO (PASSAGEIROID);

alter table ACEITE add constraint FK_ACEITEVIAGEM foreign key (VIAGEMID)
      references VIAGEM (VIAGEMID);

alter table HORPERC add constraint FK_HORPERCHORARIO foreign key (HORARIOID)
      references HORARIO (HORARIOID);

alter table HORPERC add constraint FK_HORPERCPERCURSO foreign key (PERCURSOID)
      references PERCURSO (PERCURSOID);

alter table LISTAESPERA add constraint FK_LISTAESPERAPASSAGEIRO foreign key (PASSAGEIROID)
      references PASSAGEIRO (PASSAGEIROID);

alter table LISTAESPERA add constraint FK_LISTAESPERAVIAGEM foreign key (VIAGEMID)
      references VIAGEM (VIAGEMID);

alter table RECURSO add constraint FK_RECURSOASSISTBORDO foreign key (ASSISTBORDOID)
      references ASSISTBORDO (ASSISTBORDOID);

alter table RECURSO add constraint FK_RECURSOCAMIONETA foreign key (CAMIONETAID)
      references CAMIONETA (CAMIONETAID);

alter table RECURSO add constraint FK_RECURSOMOTORISTA foreign key (MOTORISTAID)
      references MOTORISTA (MOTORISTAID);

alter table RESERVA add constraint FK_RESERVAPASSAGEIRO foreign key (PASSAGEIROID)
      references PASSAGEIRO (PASSAGEIROID);

alter table RESERVA add constraint FK_RESERVAVIAGEM foreign key (VIAGEMID)
      references VIAGEM (VIAGEMID);

alter table VIAGEM add constraint FK_VIAGEMHORPERC foreign key (HORPERCID)
      references HORPERC (HORPERCID);

alter table VIAGEM add constraint FK_VIAGEMRECURSO foreign key (RECURSOID)
      references RECURSO (RECURSOID);
```

Instruções SQL para realizar as consultas

Em todas as instruções SQL procurei colocar os campos devolvidos pela instrução SELECT que possam contribuir para melhor compreender a informação disponibilizada, apesar de não ser pedido na questão. Um dos exemplos é a primeira pergunta em que só deveria ter sido devolvido o campo CONTACTO, uma vez que responderia directamente à questão. No entanto, optei por o completar com o campo NOME uma vez que torna mais legível a informação devolvida. Nas restantes questões utilizei abordagem idêntica.

Qual o número de telefone do motorista “Carlos Silva”?

```
SELECT m.nome, m.contacto FROM bdp.motorista m where m.Nome = 'Carlos Silva'
```

Qual o horário do percurso “Porto-Lisboa”?

```
SELECT p.origem, p.destino, h.hrsaida, h.hrcheg FROM bdp.horperc r, bdp.percurso p,
bdp.horario h
where r.percursoid=p.percursoid and r.horarioid=h.horarioid
and p.origem='Porto' and p.destino='Lisboa'
```

Quais as paragens das viagens “Porto-Braga”?

```
SELECT p.origem, p.destino, p.paragens FROM bdp.percurso p
where p.origem='Porto' and p.destino='Braga'
```

Nota:
Nesta questão interpretei a palavra viagens no sentido de percurso, uma vez que não é dito nada relativamente à data (no modelo que desenhei a noção de viagem está associada à existência de uma data para a mesma). No entanto penso que esta instrução responde adequadamente à questão colocada tendo como base o modelo que desenhei.

Alguém pretende ir para Aveiro. Qual a próxima viagem disponível e a que horas se realiza?

```
SELECT v.data, p.origem, p.destino, h.hrsaida, h.hrcheg, p.paragens, p.preco
FROM bdp.horperc r, bdp.percurso p, bdp.horario h, bdp.viagem v
WHERE v.horpercid=r.horpercid AND r.percursoid=p.percursoid
AND r.horarioid=h.horarioid AND v.data>='2005-01-23' AND p.destino='Aveiro'
ORDER BY 1 ASC
```

(1) Uma vez que a pergunta nada diz sobre o local onde a pessoa se encontra, optei por mostrar toda e qualquer viagem com destino Aveiro.
(2) Se fosse necessário obter as viagens com uma determinada rigem e destino, então bastava acrescentar a condição p.origem=’Porto?, por exemplo, à clausula WHERE.
(3) Para obter a próxima coloquei uma condição que compara a data da viagem como uma data introduzida. Numa situação real penso que seria o mais adequado. No entanto, poderia ser facilmente substituída pela data do sistema.
(4) Pelas mesmas razões anteriores, a consulta obtém não a próxima viagem disponível mas uma lista das próximas viagens ordenada por data. A próxima será então a primeira da lista.

Quais os lugares ocupados numa determinada viagem?

```
SELECT v.data, p.origem, p.destino, h.hrsaida, count(a.aceiteID) as Ocupados
FROM bdp.horperc r, bdp.percurso p, bdp.horario h, bdp.viagem v, bdp.aceite a
WHERE v.horpercid=r.horpercid AND r.percursoid=p.percursoid AND r.horarioid=h.horarioid
AND v.viagemid=a.viagemid
AND v.data='2005-01-25' AND p.origem='Porto' AND p.destino='Lisboa'
AND h.hrsaida='09:00'
GROUP BY v.data, p.origem, p.destino, h.hrsaida
```

(1) Considerei aqui "determinada viagem" como uma viagem com data, origem, destino e hora de saída, uma vez que poderá haver mais viagens com o mesmo destino na data indicada mas com horários diferentes. De notar que também não era muito complicado permitir que haja mais do que uma viagem no mesmo dia, com a mesma origem, destino e hora de saída e feita por recursos (camioneta, motorista, assistente de bordo) diferentes. Se fosse esse o caso, bastaria acrescentar mais um campo (recursoID) e acrescentá-lo, também, ao GROUP BY. No entanto e de acordo com o enunciado, não me parece que seja o caso.

Quantos lugares livres existem numa determinada viagem?

```
SELECT v.data, p.origem, p.destino, h.hrsaida, (c.lugares - x.ocupados) as livres
FROM bdp.horperc r, bdp.percurso p, bdp.horario h, bdp.viagem v, bdp.aceite a, bdp.recurso
rc, bdp.camioneta c,
(select count(a.aceiteid) as ocupados from bdp.horperc r, bdp.percurso p, bdp.horario h,
bdp.viagem v, bdp.aceite a
where v.horpercid=r.horpercid and r.percursoid=p.percursoid and r.horarioid=h.horarioid and
v.viagemid=a.viagemid
and v.data='2005-01-25' and p.origem='Porto' and p.destino='Lisboa' and h.hrsaida='09:00')
x
WHERE v.horpercid=r.horpercid AND r.percursoid=p.percursoid AND r.horarioid=h.horarioid
AND v.viagemid=a.viagemid
AND v.recursoid=rc.recursoid AND rc.camionetaid=c.camionetaid
AND v.data='2005-01-25' AND p.origem='Porto' AND p.destino='Lisboa' AND
h.hrsaida='09:00'
GROUP BY v.data, p.origem, p.destino, h.hrsaida
```

(1) Como disse anteriormente considero que só quando são dados como aceites os passageiros (ou seja, já pagaram o bilhete) é que os lugares são dados como ocupados. Os lugares livres são os lugares que a camioneta possui.

Quais as reservas não confirmadas para uma determinada viagem?

```
SELECT v.data, p.origem, p.destino, h.hrsaida, x.ReservasNaoConfirmadas
FROM bdp.horperc r, bdp.percurso p, bdp.horario h, bdp.viagem v,
(select count(rs.reservaid) as ReservasNaoConfirmadas from bdp.horperc r, bdp.percurso p,
bdp.horario h, bdp.viagem v, bdp.reserva rs
where v.horpercid=r.horpercid and r.percursoid=p.percursoid and r.horarioid=h.horarioid and
v.viagemid=rs.viagemid
and v.data='2005-01-25' and p.origem='Porto' and p.destino='Lisboa' and h.hrsaida='09:00'
and rs.confirmada=0) x
WHERE v.horpercid=r.horpercid AND r.percursoid=p.percursoid AND r.horarioid=h.horarioid
AND v.data='2005-01-25' AND p.origem='Porto' AND p.destino='Lisboa' AND
h.hrsaida='09:00'
GROUP BY v.data, p.origem, p.destino, h.hrsaida
```

Quais os motoristas disponíveis para uma determinada viagem?

```
SELECT m.nome, m.contacto FROM bdp.motorista m
WHERE m.motoristaid NOT IN
(select m.motoristaid
from bdp.horperc r, bdp.percurso p, bdp.horario h, bdp.viagem v, bdp.recurso rc,
bdp.motorista m
where v.horpercid=r.horpercid and r.percursoid=p.percursoid and r.horarioid=h.horarioid
and v.recursoid=rc.recursoid and rc.motoristaid=m.motoristaid
and v.data='2005-01-25' and p.origem='Porto' and destino='Lisboa' and h.hrsaida='09:00')
```

(1) Como não foi especificado nada sobre restrições aplicadas aos motoristas em realizar ou não determinadas viagens, parto do princípio de que todos os motoristas podem fazer todas as viagens/percursos, desde que não estejam ocupados numa outra. Assim, bastará obter todos os motoristas que não estejam ocupados para a viagem indicada.
(2) Numa situação real seria de esclarecer se existem ou não restrições aplicadas aos motoristas.

Qual o número médio de viagens efectuadas por cada motorista?

Hipótese 1

```
SELECT x.nome, x.viagens as ViagensMotorista, y.totalviagens as TodasViagens,
(x.viagens/y.totalviagens) as MediaViagensMotorista FROM
(select m.motoristaid, m.nome, count(*) as viagens from bdp.viagem v, bdp.recurso rc,
bdp.motorista m
where v.recursoid=rc.recursoid and rc.motoristaid=m.motoristaid
group by m.motoristaid, m.nome) x,
(select count(*) totalviagens from bdp.viagem v) y
GROUP BY x.nome
```

Hipótese 2

```
select m.nome, x.viagens as ViagensMotorista, y.totalviagens as TodasViagens,
(x.viagens/y.totalviagens) as MediaViagensMotorista from
(select m.motoristaid, m.nome, count(*) as viagens from bdp.viagem v, bdp.recurso rc,
bdp.motorista m
where v.recursoid=rc.recursoid and rc.motoristaid=m.motoristaid
group by m.motoristaid, m.nome) x RIGHT OUTER JOIN motorista m ON
m.motoristaid=x.motoristaid,
(select count(*) totalviagens from bdp.viagem v) y
group by m.nome
```

(1) Uma vez que nada é dito parti do princípio que a média é calculada sobre o número total de viagens efectuadas por todos os motoristas até ao momento em que a consulta é feita, ou seja, registos existentes.
(2) Na hipótese 1 apresento a média de todos os motoristas que fizeram pelo menos uma viagem. Na minha opinião responde ao pedido. No entanto, caso quiséssemos obter uma lista de todos os motoristas e as respectivas média independentemente de terem alguma fez feito uma viagem, poder-se-ia utilizar a consulta da hipótese 2. Nesta é realizado um RIGHT OUTER JOIN com a tabela dos motoristas.

Qual o número máximo de horas diárias trabalhadas por um determinado assistente?

```
SELECT m.nome, v.data, MAX(h.hrcheg - h.hrsaida) as MaxHoras
FROM bdp.viagem v, bdp.recurso rc, bdp.assistbordo m, bdp.horperc hp, bdp.horario h
WHERE v.recursoid=rc.recursoid AND rc.assistbordoid=m.assistbordoid
AND v.horpercid=hp.horpercid AND hp.horarioid=h.horarioid AND m.nome = 'Nome
Assistente Bordo'
GROUP BY m.nome, v.data
```

(1) A interpretação que fiz desta consulta foi a obtenção do máximo de horas trabalhadas por um assistente em cada dia. Aceito que também fosse uma interpretação válida a obtenção do maior número de horas diárias que um determinado assistente alguma vez fez. As instruções que apresento respondem à primeira interpretação.
(2) Convém realçar que para a diferença entre hrcheg e hrsaida ficar correcta era necessário utilizar algumas funções, nomeadamente aqueles que nos permitissem obter o resultado em horas. No entanto e porque as funções normalmente estão dependentes do SGBD, optei por não as fazer aqui. De qualquer das formas e neste exemplo, se o resultado da diferença for divido por 10000, obtemos os valores em horas.

Qual o número médio de passageiros em cada um dos percursos efectuados pela empresa?

```
SELECT x.origem, x.destino, x.Ocupados AS Passageiros, y.TotalViagensPercurso AS
ViagensPercurso, (x.Ocupados / y.TotalViagensPercurso) AS MediaPassageiros FROM

(select p.origem, p.destino, count(a.aceiteID) as Ocupados from bdp.horperc r, bdp.percurso
p, bdp.viagem v, bdp.aceite a
where v.horpercid=r.horpercid and r.percursoid=p.percursoid and v.viagemid=a.viagemid
and v.data <= '2005-01-30'
group by p.origem, p.destino) x,

(select p.origem, p.destino, count(*) totalviagenspercurso from bdp.horperc r, bdp.percurso
p, bdp.viagem v
where v.horpercid=r.horpercid and r.percursoid=p.percursoid and v.data <='2005-01-30'
group by p.origem, p.destino) y

WHERE x.origem=y.origem AND x.destino=y.destino
```

(1) Em x obtenho uma tabela com o número de passageiros por cada percurso que já tem passageiros aceites e que já foram efectuados (esta condição está garantida pela condição v.data <= determinada data).
(2) Em y obtenho o total de viagens efectuadas de cada percurso, independentemente de terem tido passageiros ou não (pode dar-se o caso de a camioneta ter de ir para um destino mesmo sem passageiros de forma a poder realizar a viagem de regresso).
(3) Finalmente faço uma junção natural entre x e y e calculo a média da forma indicada. Também apresentei, embora desnecessário para responder à pergunta, o número total de passageiros e o total de viagens efectuadas por cada percurso.

Quais os percursos com uma ocupação média inferior a 50%?

```
SELECT x.origem, x.destino, x.Ocupados AS Passageiros, y.lugaresdisponiveis AS Capacidade,
(x.ocupados/y.lugaresdisponiveis*100) AS Ocupacao
FROM

(select p.origem, p.destino, count(a.aceiteID) as Ocupados from bdp.horperc r, bdp.percurso
p, bdp.viagem v, bdp.aceite a
where v.horpercid=r.horpercid and r.percursoid=p.percursoid and v.viagemid=a.viagemid
and v.data <= '2005-01-30'
group by p.origem, p.destino) x,

(select p.origem, p.destino, sum(c.lugares) as lugaresdisponiveis from bdp.horperc r,
bdp.percurso p, bdp.viagem v,
bdp.recurso rc, bdp.camioneta c
where v.horpercid=r.horpercid and r.percursoid=p.percursoid and v.recursoid=rc.recursoid
and rc.camionetaid=c.camionetaid
and v.data <='2005-01-30'
group by p.origem, p.destino) y

WHERE x.origem=y.origem AND x.destino=y.destino
HAVING Ocupacao < 50
```

(1) Parti do princípio que interessa a ocupação dos percursos já realizados. Garanto isso pela condição v.data <= determinada data.
(2) Em x obtenho uma tabela com o número de passageiros por cada percurso que já tem passageiros aceites e que já foram efectuados.
(3) Em y obtenho a capacidade de lugares disponibilizada para cada percurso (é a soma dos lugares da camioneta).
(4) Finalmente faço uma junção natural entre x e y e calculo a ocupação média da forma indicada. Também apresentei, embora desnecessário para responder à pergunta, o número total de passageiros e a capacidade disponível para o total de viagens efectuadas por cada percurso.
(5) A clausula HAVING permite seleccionar os que têm uma ocupação inferior a 50%.

Nota Final

Para concluir gostaria de informar que todas as instruções SQL para construir as tabelas, índices, restrições, etc., apresentadas neste trabalho, foram utilizadas no MySQL V4.0 para construir a Base de Dados com o modelo aqui apresentado.

Da mesma forma todas as instruções de consulta foram experimentadas sobre a base de dados criada no MySQL embora com um número restrito de registos.

Finalmente, não foram realizadas nenhuns testes para verificar se as consultas efectuadas estão ou não devidamente optimizadas. De qualquer das formas, penso que também não era esse o objectivo deste trabalho.

Bibliografia

Luís Torgo

Acetatos da cadeira de Bases de Dados e Programação