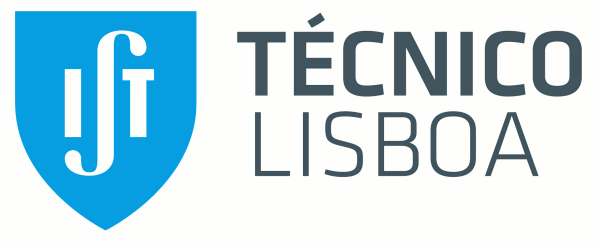

Sistemas de Informação e Base de Dados

MEEC

Relatório do Projeto – Parte 2

Grupo N.º 2

Autores:

Leandro Almeida – n.º 84112
Sofia Estrela – n.º 84186
Vasco Candeias – n.º 84196

Turno:

Segunda-feira 15h30-17h

9 de Novembro de 2018

1 Criação da base de dados

A criação da base de dados, cujo código é apresentado de seguida, teve como base não só o modelo relacional fornecido mas também o enunciado original do projeto, que permite adicionar algumas restrições de cardinalidade e decidir os atributos que devem ser `not null`.

```
drop table if exists produced_indicator;
drop table if exists test_procedure;
drop table if exists radiography;
drop table if exists performed;
drop table if exists proced;
drop table if exists indicator;
drop table if exists prescription;
drop table if exists medication;
drop table if exists consult_diagnosis;
drop table if exists diagnosis_code;
drop table if exists participation;
drop table if exists consult;
drop table if exists animal;
drop table if exists generalization_species;
drop table if exists species;
drop table if exists assistant;
drop table if exists veterinary;
drop table if exists client;
drop table if exists phone_number;
drop table if exists person;

create table person(
  VAT varchar(20),
  name varchar(255) not null,
  address_street varchar(255) not null,
  address_city varchar(255) not null,
  address_zip varchar(255) not null,
  primary key(VAT)
);

create table phone_number(
  VAT varchar(20),
  phone numeric(9,0) not null,
  primary key(VAT, phone),
  foreign key(VAT) references person(VAT)
);

create table client(
  VAT varchar(20),
  primary key(VAT),
  foreign key(VAT) references person(VAT)
);

create table veterinary(
  VAT varchar(20),
  specialization varchar(255) not null, -- is required additional
     information
  bio varchar(255) not null,
  primary key(VAT),
  foreign key(VAT) references person(VAT)
);
```

```
create table assistant(
  VAT varchar(20),
  primary key(VAT),
  foreign key(VAT) references person(VAT)
);

create table species(
  name varchar(255),
  description varchar(255) not null,
  primary key(name)
);

create table generalization_species(
  name1 varchar(255),
  name2 varchar(255),
  primary key(name1),
  foreign key(name1) references species(name),
  foreign key(name2) references species(name)
);

create table animal(
  name varchar(255),
  VAT varchar(20),
  species_name varchar(255),
  colour varchar(255) not null,
  gender varchar(255) not null,
  birth_year date not null,
  age integer not null,
  primary key(name, VAT),
  foreign key(VAT) references client(VAT) on delete cascade,
  foreign key(species_name) references species(name),
  check(birth_year <= current_date),
  check(age = timestampdiff(YEAR, birth_year, NOW())),
  check(gender='male' or gender='female' or gender='other')
);

create table consult(
  name varchar(255),
  VAT_owner varchar(20),
  date_timestamp timestamp not null,
  s varchar(255),
  o varchar(255),
  a varchar(255),
  p varchar(255),
  VAT_client varchar(20),
  VAT_vet varchar(20),
  weight numeric(5,2) not null,
  primary key(name, VAT_owner, date_timestamp),
  foreign key(name, VAT_owner) references animal(name, VAT) on delete
     cascade,
  foreign key(VAT_client) references client(VAT) on delete cascade,
  foreign key(VAT_vet) references veterinary(VAT),
  check(weight>0),
  check(date_timestamp <= current_date)
);

create table participation(
```

```
  name varchar(255),
  VAT_owner varchar(20),
  date_timestamp timestamp,
  VAT_assistant varchar(20),
  primary key(name, VAT_owner, date_timestamp, VAT_assistant),
  foreign key(name, VAT_owner, date_timestamp) references
     consult(name, VAT_owner, date_timestamp) on delete cascade,
  foreign key(VAT_assistant) references assistant(VAT)
);

create table diagnosis_code(
  code varchar(255),
  name varchar(255) not null,
  primary key(code)
);

create table consult_diagnosis(
  code varchar(255),
  name varchar(255),
  VAT_owner varchar(20),
  date_timestamp timestamp,
  primary key(code, name, VAT_owner, date_timestamp),
  foreign key(code) references diagnosis_code(code),
  foreign key(name, VAT_owner, date_timestamp) references
     consult(name, VAT_owner, date_timestamp) on delete cascade
);

create table medication(
  name varchar(255),
  lab varchar(255),
  dosage numeric(20, 2),
  primary key(name, lab, dosage),
  check(dosage>=0)
);

create table prescription(
  code varchar(255),
  name varchar(255),
  VAT_owner varchar(20),
  date_timestamp timestamp,
  name_med varchar(255),
  lab varchar(255),
  dosage numeric(20, 2),
  regime varchar(255),
  primary key(code, name, VAT_owner, date_timestamp, name_med, lab,
     dosage),
  foreign key(code, name, VAT_owner, date_timestamp) references
     consult_diagnosis(code, name, VAT_owner, date_timestamp) on
     update cascade on delete cascade,
  foreign key(name_med, lab, dosage) references medication(name, lab,
     dosage)
);

create table indicator(
  name varchar(255),
  reference_value numeric(10, 2) not null,
  units varchar(255) not null,
```

```
  description varchar(255) not null,
  primary key(name)
  );

create table proced(
  name varchar(255),
  VAT_owner varchar(20),
  date_timestamp timestamp,
  num numeric(20, 0),
  description varchar(255) not null,
  primary key(name, VAT_owner, date_timestamp, num),
  foreign key(name, VAT_owner, date_timestamp) references
     consult(name, VAT_owner, date_timestamp) on delete cascade
);

create table performed(
  name varchar(255),
  VAT_owner varchar(20),
  date_timestamp timestamp,
  num numeric(20, 0),
  VAT_assistant varchar(20),
  primary key(name, VAT_owner, date_timestamp, num, VAT_assistant),
  foreign key(name, VAT_owner, date_timestamp, num) references
     proced(name, VAT_owner, date_timestamp, num) on delete cascade,
  foreign key(VAT_assistant) references assistant(VAT) on delete
     cascade
);

create table radiography(
  name varchar(255),
  VAT_owner varchar(20),
  date_timestamp timestamp,
  num numeric(20, 0),
  file varchar(255) not null,
  primary key(name, VAT_owner, date_timestamp, num),
  foreign key(name, VAT_owner, date_timestamp, num) references
     proced(name, VAT_owner, date_timestamp, num) on delete cascade
);

create table test_procedure(
  name varchar(255),
  VAT_owner varchar(20),
  date_timestamp timestamp,
  num numeric(20, 0),
  type varchar(255) not null,
  primary key(name, VAT_owner, date_timestamp, num),
  foreign key(name, VAT_owner, date_timestamp, num) references
     proced(name, VAT_owner, date_timestamp, num) on delete cascade,
  check(type = 'blood' or type = 'urine')
);

create table produced_indicator(
  name varchar(255),
  VAT_owner varchar(20),
  date_timestamp timestamp,
  num numeric(20, 0),
  indicator_name varchar(255),
```

```
  value numeric(20, 1) not null,
  primary key(name, VAT_owner, date_timestamp, num, indicator_name),
  foreign key(name, VAT_owner, date_timestamp, num) references
     test_procedure(name, VAT_owner, date_timestamp, num) on delete
     cascade,
  foreign key(indicator_name) references indicator(name)
);
```

Código 1: Criação das tabelas

2 Registos na base de dados

Para a população das tabelas, tentou-se dificultar as seleções das *queries*, introduzindo dados relevantes. No entanto, foi difícil manter uma base de dados com todos os casos para todas as *queries*, pelo que os testes feitos ao longo do trabalho utilizaram ainda outros dados específicos para cada, testando-se os casos limite. Ao longo deste relatório, apresentam-se os resultados para a base de dados mais geral.

```
insert into person values
('00000000', 'John Smith', 'Main Street', 'New York', '1000-193'),
('00000001', 'John Smith', 'Second Street', 'Brooklyn', '1001-439'),
('00000002', 'John Doe', 'Main Street', 'Cleveland', '1010-798'),
('00000003', 'Oliver Watts', 'Big Avenue', 'Los Angeles', '1050-375'),
('00000004', 'Chandler Webbs', 'Small Avenue', 'Brooklyn',
   '1001-373'),
('00000005', 'John Smith', 'Main Street', 'New York', '1000-278'),
('00000006', 'William Lawrence', 'Second Street', 'Brooklyn',
   '1001-438'),
('00000007', 'Kelly Jenkins', 'Main Street', 'Cleveland', '1010-348'),
('00000008', 'Jacob Chambers', 'Second Street', 'Brooklyn',
   '1001-230'),
('00000009', 'Harry Spencer', 'Big Avenue', 'Brooklyn', '1001-436'),
('00000010', 'Jack Lawson', 'Third Street', 'California', '1070-089'),
('00000011', 'Rhys Woods', 'Main Street', 'Dallas', '1100-324'),
('00000012', 'Thomas Ress', 'Big Avenue', 'Chicago', '1078-375'),
('00000013', 'George Fraser', 'Small Street', 'Brooklyn', '1001-850'),
('00000014', 'Damian Black', 'Big Avenue', 'Dallas', '1100-279'),
('00000015', 'Joe Fletcher', 'Third Street', 'Houston', '1073-384'),
('00000016', 'Jones', 'Small Avenue', 'Brooklyn', '1001-379'),
('00000017', 'Noah Taylor', 'Small Avenue', 'Fort Worth', '1356-167'),
('00000019', 'Ethan Brown', 'Second Street', 'San Diego', '1263-368'),
('00000020', 'Joseph Davis', 'Third Street', 'Fort Worth',
   '1356-738');

insert into phone_number values
('00000000', 934368287),
('00000001', 963154632),
('00000002', 926842214),
('00000003', 913546541),
('00000005', 933987048),
('00000006', 966984516),
('00000007', 925424700),
('00000008', 917813568),
('00000009', 933534684),
('00000010', 927132454),
('00000010', 938713541),
('00000010', 968103912),
```

```
('00000013', 926846513),
('00000014', 936841324),
('00000015', 961351153),
('00000016', 963154632),
('00000017', 963154632),
('00000019', 918468344),
('00000020', 910360480);

insert into client values
('00000000'),
('00000002'),
('00000003'),
('00000005'),
('00000006'),
('00000007'),
('00000009'),
('00000010'),
('00000014'),
('00000015'),
('00000017'),
('00000020');

insert into veterinary values
('00000001', 'Doctor', 'I wanted to be a doctor but I dont like
   people.'),
('00000004', 'Critical Care Veterinary', 'Since I was 1, I wanted to
   be a veterinary!'),
('00000005', 'Anaesthsiologist', 'Since I was 5, I wanted to be a
   veterinary!'),
('00000008', 'Cardiologist', 'Since I was 10, I wanted to be a
   veterinary!'),
('00000009', 'Oncologist', 'Since I was 15, I wanted to be a
   veterinary!'),
('00000011', 'Parasitologist', 'Since I was 20, I wanted to be a
   veterinary!'),
('00000013', 'Toxicologist', 'My mum wanted me to be a veterinary. So
   here I am.');

insert into assistant values
('00000010'),
('00000012'),
('00000016'),
('00000019');

insert into species values
('Dog', 'The domestic dog is the most widely abundant terrestrial
   carnivore.'),
('Bulldog', 'It is a muscular, hefty dog with a wrinkled face and a
   distinctive pushed-in nose.'),
('Pug', 'The Pug is a breed of dog with physically distinctive
   features of a wrinkly, short-muzzled face, and curled tail.'),
('Rottweiler', 'Rottweilers are used as search and rescue dogs, as
   guard dogs, and as police dogs.'),
('Husky', 'Husky is a general name for a sled-type of dog used in
   northern regions.'),
('Afghan Hound', 'The Afghan Hound is a hound with thick, fine, silky
   coat and a tail with a ring curl at the end.'),
```

```
('Akita', 'The Akita is a large breed of dog originating from the
   mountainous regions of northern Japan.'),
('Bird', 'Birds have feathers, toothless beaked jaws, hard-shelled
   eggs and a strong yet lightweight skeleton.'),
('Parakeet bird', 'A parakeet is a small to medium-sized species of
   parrot with generally long tail feathers.'),
('Cockatiel bird', 'Cockatiels are prized as companion parrots and
   are relatively easy to breed.'),
('Hamster', 'Hamsters are rodents belonging to the subfamily
   Cricetinae.'),
('Cat', 'A cat is a small, typically furry, carnivorous mammal.'),
('Siamese', 'The Siamese cat is one of the first distinctly
   recognized breeds of Asian cat.'),
('Donskoy', 'The Donskoy is a mostly hairless cat breed of Russian
   origin.'),
('Fish', 'Fish are gill-bearing aquatic craniate animals that lack
   limbs with digits.');

insert into generalization_species values
('Bulldog', 'Dog'),
('Pug', 'Dog'),
('Rottweiler', 'Dog'),
('Husky', 'Dog'),
('Afghan Hound', 'Dog'),
('Akita', 'Dog'),
('Parakeet bird', 'Bird'),
('Cockatiel bird', 'Bird'),
('Siamese', 'Cat'),
('Donskoy', 'Cat');

insert into animal values
('Puma', '00000000', 'Bulldog', 'Black', 'Male', '1997-12-18', 20),
('Theo', '00000000', 'Siamese', 'Grey', 'Female', '2000-12-19', 17),
('Cally', '00000014', 'Parakeet bird', 'Green', 'Male', '2001-10-16',
   17),
('Doggy', '00000014', 'Dog', 'Cream', 'Female', '2002-09-23', 16),
('Severin', '00000015', 'Pug', 'Brown', 'Male', '2003-08-29', 15),
('Wanikiy', '00000015', 'Afghan Hound', 'Golden', 'Female',
   '2004-07-30', 14),
('Fluffy', '00000002', 'Akita', 'Brown', 'Male', '2005-06-27', 13),
('Bonzo', '00000002', 'Rottweiler', 'Grey', 'Female', '2006-05-29',
   12),
('Marlie', '00000006', 'Husky', 'White', 'Male', '2007-04-21', 11),
('Bolt', '00000007', 'Bird', 'Yellow', 'Female', '2008-03-19', 10),
('Spark', '00000003', 'Cockatiel bird', 'Black', 'Female',
   '2009-02-05', 9),
('Garfield', '00000005', 'Cat', 'Orange', 'Male', '2015-01-02', 3),
('Toby', '00000006', 'Husky', 'White', 'Male', '2017-10-08', 1),
('Trojan', '00000006', 'Fish', 'Blue', 'Female', '2017-10-09', 1),
('Blue', '00000003', 'Bird', 'Blue', 'Female', '2009-02-05', 1),
('Blue', '00000002', 'Bird', 'Blue', 'Female', '2009-02-05', 1),
('Foxy', '00000005', 'Bulldog', 'Brown', 'Male', '2008-02-05', 2),
('Goofy', '00000005', 'Husky', 'Brown', 'Male', '2008-02-05', 2),
('Nugget', '00000005', 'Rottweiler', 'Brown', 'Female', '2007-02-05',
   3);

insert into consult values
```

```
('Puma', '00000000', '2017-7-27 10:30:00.75', 's', 'Suffers from
   obesity.', 'a', 'p', '00000020', '00000001', 35),
('Puma', '00000000', '2017-8-27 09:00:00.75', 's', 'Extreme obesity.
   Poor Boy.', 'a', 'p', '00000020', '00000001', 34),
('Severin', '00000015', '2016-7-28 11:00:00.75', 's', 'o', 'a', 'p',
   '00000015', '00000001', 17),
('Severin', '00000015', '2016-7-29 11:45:00.75', 's', 'Obese but
   working on it.', 'a', 'p', '00000015', '00000001', 40),
('Marlie', '00000006', '2016-11-29 19:00:00.75', 's', 'o', 'a', 'p',
   '00000020', '00000001', 30),
('Blue', '00000003', '2016-11-29 09:00:00.75','s', 'o', 'a', 'p',
   '00000020', '00000001', 1),
('Blue', '00000002', '2016-11-30 09:00:00.75','s', 'o', 'a', 'p',
   '00000002', '00000001', 1),
('Bolt', '00000007', '2005-7-29 09:00:00.75', 's', 'o', 'a', 'p',
   '00000007', '00000009', 20),
('Fluffy', '00000002', '2005-7-29 09:00:00.75', 's', 'o', 'a', 'p',
   '00000002', '00000001', 20),
('Garfield', '00000005', '2005-7-29 09:00:00.75', 's', 'It is an
   obese cat.', 'a', 'p', '00000005', '00000009', 40),
('Severin', '00000015', '2018-7-30 09:00:00.75', 's', 'Almost obese
   but no.', 'a', 'p', '00000015', '00000004', 25),
('Severin', '00000015', '2005-7-31 09:00:00.75', 's', 'o', 'a', 'p',
   '00000015', '00000004', 20),
('Severin', '00000015', '2005-8-29 09:00:00.75', 's', 'o', 'a', 'p',
   '00000015', '00000005', 20),
('Doggy', '00000014', '2005-9-29 12:30:00.75', 's', 'o', 'a', 'p',
   '00000014', '00000008', 26),
('Theo', '00000000', '2005-10-29 15:30:00.75', 's', 'o', 'a', 'p',
   '00000000', '00000008', 1),
('Cally', '00000014', '2005-1-29 09:00:00.75', 's', 'o', 'a', 'p',
   '00000009', '00000004', 1),
('Wanikiy', '00000015', '2005-2-27 09:00:00.75', 's', 'o', 'a', 'p',
   '00000010', '00000001', 31),
('Bonzo', '00000002', '2005-3-29 09:00:00.75', 's', 'o', 'a', 'p',
   '00000002', '00000008', 15),
('Spark', '00000003', '2005-8-29 09:00:00.75', 's', 'o', 'a', 'p',
   '00000020', '00000013', 1),
('Toby', '00000006', '2005-9-29 17:40:00.75', 's', 'o', 'a', 'p',
   '00000020', '00000013', 2),
('Trojan', '00000006', '2005-10-29 16:30:00.75', 's', 'o', 'a', 'p',
   '00000005', '00000008', 1),
('Trojan', '00000006', '2017-10-29 09:30:00.75', 's', 'o', 'a', 'p',
   '00000005', '00000008', 1),
('Bonzo', '00000002', '2017-3-29 10:30:00.75', 's', 'o', 'a', 'p',
   '00000002', '00000008', 15),
('Spark', '00000003', '2017-8-29 11:30:00.75', 's', 'o', 'a', 'p',
   '00000020', '00000013', 1),
('Foxy', '00000005', '2018-1-10 11:00:00.75', 's', 'o', 'a', 'p',
   '00000005', '00000013', 10),
('Goofy', '00000005', '2018-1-10 12:30:00.75', 's', 'o', 'a', 'p',
   '00000005', '00000013', 10),
('Nugget', '00000005', '2018-1-10 14:00:00.75', 's', 'o', 'a', 'p',
   '00000005', '00000013', 15),
('Wanikiy', '00000015', '2018-2-27 09:00:00.75', 's', 'o', 'a', 'p',
   '00000010', '00000001', 31);
```

```
insert into participation values
('Puma', '00000000', '2017-7-27 10:30:00.75', '00000010'),
('Puma', '00000000', '2017-8-27 09:00:00.75', '00000010'),
('Puma', '00000000', '2017-8-27 09:00:00.75', '00000012'),
('Severin', '00000015', '2016-7-28 11:00:00.75', '00000012'),
('Theo', '00000000', '2005-10-29 15:30:00.75', '00000012'),
('Cally', '00000014', '2005-1-29 09:00:00.75','00000010'),
('Cally', '00000014', '2005-1-29 09:00:00.75','00000012'),
('Bonzo', '00000002', '2005-3-29 09:00:00.75','00000012'),
('Trojan', '00000006', '2017-10-29 09:30:00.75', '00000010'),
('Trojan', '00000006', '2017-10-29 09:30:00.75', '00000012'),
('Trojan', '00000006', '2017-10-29 09:30:00.75', '00000016'),
('Trojan', '00000006', '2017-10-29 09:30:00.75', '00000019'),
('Bonzo', '00000002', '2017-3-29 10:30:00.75', '00000019');

insert into diagnosis_code values
('0000', 'High Blood Pressure'),
('1111', 'Kidney Failure'),
('2222', 'Cron disease'),
('3333', 'Flu'),
('4444', 'Fever'),
('5555', 'Broken Nose'),
('6666', 'End-stage renal disease');

insert into medication values
('Gaviscom', 'Gaviscom Inc.', 800),
('Ben-u-ron', 'Ben-u-ron Inc.', 1000),
('Ben-u-ron', 'Ben-u-ron Inc.', 500),
('Brufen', 'Brufen Inc.', 100),
('Brufen', 'Brufen Inc.', 50);

insert into consult_diagnosis values
('0000', 'Puma', '00000000', '2017-8-27 09:00:00.75'),
('1111', 'Puma', '00000000', '2017-7-27 10:30:00.75'),
('2222', 'Puma', '00000000', '2017-7-27 10:30:00.75'),
('0000', 'Foxy', '00000005', '2018-1-10 11:00:00.75'),
('1111', 'Foxy', '00000005', '2018-1-10 11:00:00.75'),
('3333', 'Severin', '00000015', '2016-7-28 11:00:00.75'),
('0000', 'Wanikiy', '00000015', '2005-2-27 09:00:00.75'),
('1111', 'Wanikiy', '00000015', '2005-2-27 09:00:00.75'),
('0000', 'Wanikiy', '00000015', '2018-2-27 09:00:00.75'),
('2222', 'Marlie', '00000006', '2016-11-29 19:00:00.75'),
('3333', 'Marlie', '00000006', '2016-11-29 19:00:00.75'),
('2222', 'Goofy', '00000005', '2018-1-10 12:30:00.75'),
('4444', 'Bonzo', '00000002', '2017-3-29 10:30:00.75'),
('5555', 'Bonzo', '00000002', '2017-3-29 10:30:00.75'),
('4444', 'Nugget', '00000005', '2018-1-10 14:00:00.75'),
('5555', 'Nugget', '00000005', '2018-1-10 14:00:00.75'),
('5555', 'Fluffy', '00000002', '2005-7-29 09:00:00.75'),
('0000', 'Trojan', '00000006', '2005-10-29 16:30:00.75'),
('1111', 'Doggy', '00000014', '2005-9-29 12:30:00.75'),
('2222', 'Theo', '00000000', '2005-10-29 15:30:00.75'),
('4444', 'Toby', '00000006', '2005-9-29 17:40:00.75'),
('0000', 'Toby', '00000006', '2005-9-29 17:40:00.75'),
('3333', 'Spark', '00000003', '2017-8-29 11:30:00.75');

insert into prescription values
```

```
('0000', 'Puma', '00000000', '2017-8-27 09:00:00.75', 'Gaviscom',
   'Gaviscom Inc.', 800, 'One in the morning.'),
('0000', 'Foxy', '00000005', '2018-1-10 11:00:00.75', 'Brufen',
   'Brufen Inc.', 100, 'Every now and then.'),
('0000', 'Trojan', '00000006', '2005-10-29 16:30:00.75', 'Ben-u-ron',
   'Ben-u-ron Inc.', 1000, 'Every once in a while.'),
('1111', 'Puma', '00000000', '2017-7-27 10:30:00.75', 'Brufen',
   'Brufen Inc.', 100, 'Day yes day no.'),
('1111', 'Doggy', '00000014', '2005-9-29 12:30:00.75', 'Ben-u-ron',
   'Ben-u-ron Inc.', 1000, 'After eating.'),
('2222', 'Puma', '00000000', '2017-7-27 10:30:00.75',
   'Ben-u-ron','Ben-u-ron Inc.', 1000, '30 minutes before bed.'),
('2222', 'Theo', '00000000', '2005-10-29 15:30:00.75',
   'Ben-u-ron','Ben-u-ron Inc.', 500, '1 hour before bed.'),
('4444', 'Toby', '00000006', '2005-9-29 17:40:00.75',
   'Brufen','Brufen Inc.', 50, 'When feeling intense pain.');

insert into indicator values
('White Blood Cells', 100, 'milligrams', 'Cells of the immune system
   that protected the body against foreign invaders.'),
('Red Blood Cells', 50, 'grams', 'Cells that deliverer oxygen (O2) to
   the body tissues'),
('Erythrocytes', 500, 'grams', 'Cells of the immune system that
   protected the body against foreign invaders.'),
('Fatty acids', 110, 'milligrams', 'A carboxylic acid with a long
   aliphatic chain, which is either saturated or unsaturated.'),
('Creatinine level', 0.6, 'milligrams per deciliter', 'It is a
   breakdown product of creatine phosphate in muscle.'),
('Cholesterol', 200, 'milligrams', 'Organic molecule.'),
('Insulin', 350, 'milligrams', 'It is a peptide hormone produced by
   beta cells of the pancreatic islets.');

insert into proced values
('Puma', '00000000', '2017-7-27 10:30:00.75', 1, 'testing wbc'),
('Puma', '00000000', '2017-7-27 10:30:00.75', 2, 'testing rbc'),
('Puma', '00000000', '2017-7-27 10:30:00.75', 3, 'testing
   creatinine'),
('Puma', '00000000', '2017-8-27 09:00:00.75', 1, 'testing wbc'),
('Doggy', '00000014', '2005-9-29 12:30:00.75', 1, 'testing
   creatinine'),
('Doggy', '00000014', '2005-9-29 12:30:0.75', 2, 'radiography'),
('Trojan', '00000006', '2017-10-29 09:30:00.75', 1, 'radiography'),
('Trojan', '00000006', '2017-10-29 09:30:00.75', 2, 'testing wbc'),
('Bonzo', '00000002', '2017-3-29 10:30:00.75', 1, 'radiography');

insert into test_procedure values
('Puma', '00000000', '2017-7-27 10:30:00.75', 1, 'blood'),
('Puma', '00000000', '2017-7-27 10:30:00.75', 2, 'blood'),
('Puma', '00000000', '2017-8-27 09:00:00.75', 1, 'blood'),
('Doggy', '00000014', '2005-9-29 12:30:00.75', 1, 'blood'),
('Puma', '00000000', '2017-7-27 10:30:00.75', 3, 'blood'),
('Trojan', '00000006', '2017-10-29 09:30:00.75', 2, 'blood');


insert into produced_indicator values
('Puma', '00000000', '2017-7-27 10:30:00.75', 1, 'White Blood Cells',
   1.3),
```

```
('Trojan', '00000006', '2017-10-29 09:30:00.75', 2, 'White Blood
   Cells', 1.4),
('Puma', '00000000', '2017-7-27 10:30:00.75', 2, 'Red Blood Cells',
   500),
('Puma', '00000000', '2017-8-27 09:00:00.75', 1, 'White Blood Cells',
   1.1),
('Doggy', '00000014', '2005-9-29 12:30:00.75', 1, 'Creatinine level',
   0.7),
('Puma', '00000000', '2017-7-27 10:30:00.75', 3, 'Creatinine level',
   1.1);

insert into performed values
('Puma', '00000000', '2017-7-27 10:30:00.75', 1, '00000010'),
('Puma', '00000000', '2017-7-27 10:30:00.75', 2, '00000016'),
('Puma', '00000000', '2017-7-27 10:30:00.75', 3, '00000019'),
('Puma', '00000000', '2017-8-27 09:00:00.75', 1, '00000016'),
('Doggy', '00000014', '2005-9-29 12:30:00.75', 1, '00000010'),
('Doggy', '00000014', '2005-9-29 12:30:00.75', 2, '00000016'),
('Trojan', '00000006', '2017-10-29 09:30:00.75', 1, '00000010'),
('Trojan', '00000006', '2017-10-29 09:30:00.75', 2, '00000016'),
('Bonzo', '00000002', '2017-3-29 10:30:00.75', 1, '00000019');

insert into radiography values
('Doggy','00000014', '2005-9-29 12:30:00.75', 2, '/path/to/file.img'),
('Trojan', '00000006', '2017-10-29 09:30:00.75', 1,
   '/path/to/file.img'),
('Bonzo', '00000002', '2017-3-29 10:30:00.75', 1,
   '/path/to/file.img');
```

Código 2: Criação das tabelas

3 *Queries*

Em todas as *queries* realizadas, de modo a minimizar os acessos às tabelas, aplicaram-se primeiro as condições mais restritas, colocando, por exemplo nos `and`, a igualdade com menor diversidade de valores primeiro.

3.1 Veterinário *John Smith*

```
select distinct animal.name, O.name as owner_name, species_name,
   animal.age
from animal
inner join consult on animal.VAT = consult.VAT_owner and animal.name
   = consult.name
inner join person V on V.VAT = VAT_vet
inner join person O on O.VAT = consult.VAT_owner
where V.name = 'John Smith';
```

Código 3: *Query* 1

Uma vez que na tabela `person` é necessário filtrar por `VAT` de duas maneiras independentes (uma para os veterinários e outra para os donos dos animais) agrupou-se duas vezes esta tabela, uma para cada pessoa. Realça-se o uso de *distinct* pois o mesmo animal pode ir em datas diferentes a uma consulta com o mesmo veterinário.

A execução e resultado da *query* apresentam-se na figura abaixo. Pode-se reparar que não foi incluído o gato *Theo*, mesmo tendo o seu dono o nome de *John Smith*. Os resultados também

não apareceram repetidos o que poderia ter acontecido por se ter o mesmo animal com duas consultas com o mesmo médico veterinário. No entanto, nesta *query* pode ser considerado mais que um médico no caso de também se chamar *John Smith*, fazendo a listagem das consultas de ambos.

```
MySQL [ist425496]> select distinct
    -> animal.name, O.name as owner_name, species_name, animal.age
    -> from animal
    -> inner join consult on animal.VAT = consult.VAT_owner
    -> and animal.name = consult.name
    -> inner join person V on V.VAT = VAT_vet
    -> inner join person O on O.VAT = consult.VAT_owner
    -> where V.name = 'John Smith';
+---------+-----------------+--------------+-----+
| name    | owner_name      | species_name | age |
+---------+-----------------+--------------+-----+
| Blue    | John Doe        | Bird         |   1 |
| Blue    | Oliver Watts    | Bird         |   1 |
| Fluffy  | John Doe        | Akita        |  13 |
| Marlie  | William Lawrence| Husky        |  11 |
| Puma    | John Smith      | Bulldog      |  20 |
| Severin | Joe Fletcher    | Pug          |  15 |
| Wanikiy | Joe Fletcher    | Afghan Hound |  14 |
+---------+-----------------+--------------+-----+
7 rows in set (0.01 sec)
```

Figura 1: Resultado da *query* 1

3.2 Indicadores em *milligrams*

```
select name, reference_value
from indicator
where units = 'milligrams' and reference_value > 100
order by reference_value desc;
```

Código 4: *Query* 2

Apresenta-se na figura seguinte o resultado desta *query*. É de notar que existem 4 indicadores a miligramas, no entanto só figuram 3 pois o quarto não tem o valor de referência acima dos 100. Pode ser comprovada ainda a ordem decrescente, devido à reordenação exigida por esta *query*. Por fim, é também de realçar a ausência de um indicador que tinha como por unidades *milligrams per decimeter*.

```
MySQL [ist425496]> select name, reference_value
    -> from indicator
    -> where units = 'milligrams' and reference_value > 100
    -> order by reference_value desc;
+-------------+-----------------+
| name        | reference_value |
+-------------+-----------------+
| Insulin     |          350.00 |
| Cholesterol |          200.00 |
| Fatty acids |          110.00 |
+-------------+-----------------+
3 rows in set (0.01 sec)
```

Figura 2: Resultado da *query* 2

3.3 Obesidade

```
select animal.name, person.name as owner_name, species_name, age
from client
natural join animal
inner join consult on animal.VAT = consult.VAT_owner and animal.name
   = consult.name
```

```
inner join person using(VAT)
where (o like '%obesity%' or o like '%obese%') and weight > 30
and date_timestamp in(
  select max(date_timestamp) from animal as a natural join consult
  where a.name=animal.name and a.VAT=animal.VAT
  group by a.VAT, a.name);
```

Código 5: *Query* 3

Para saber qual é o peso mais atualizado, teve de se fazer um agrupamento por cada animal, selecionando o `max(date_timestamp)` das consultas a si associadas.

Os resultados para esta *query*, tendo em conta a população inicial, correspondem ao esperado e encontram-se representados na figura abaixo. Pode-se comprovar que o caso mais simples funciona com a entrada *Garfield* contendo a palavra *obese*. Testou-se ainda um conjunto de consultas onde o animal cumpriu os parâmetros desejados mas que na última consulta teve um peso menor que 30 kg. Por fim, pela entrada *Puma* pode-se aferir o reconhecimento da palavra *obesity*.

```
MySQL [ist425496]> select
    -> animal.name, person.name as owner_name, species_name, age
    -> from client
    -> natural join animal
    -> inner join consult on animal.VAT = consult.VAT_owner
    -> and animal.name = consult.name
    -> inner join person using(VAT)
    -> where (o like '%obesity%' or o like '%obese%') and weight > 30
    -> and date_timestamp in(
    ->   select max(date_timestamp) from animal as a natural join consult
    ->   where a.name=animal.name and a.VAT=animal.VAT
    ->   group by a.VAT, a.name);
+----------+------------+--------------+-----+
| name     | owner_name | species_name | age |
+----------+------------+--------------+-----+
| Puma     | John Smith | Bulldog      |  20 |
| Garfield | John Smith | Cat          |   3 |
+----------+------------+--------------+-----+
2 rows in set (0.05 sec)
```

Figura 3: Resultado da *query* 3

3.4 Clientes sem animais

```
select name, VAT, address_street, address_city, address_zip
from person natural join client
where VAT not in (select a.VAT from animal a);
```

Código 6: *Query* 4

A figura abaixo mostra os resultados obtidos para esta *query*. Foram reconhecidos os clientes que se esperavam, pois nenhum destes possui de facto animais em seu nome, mesmo que alguns possam aparecer a acompanhar animais a consultas.

```
MySQL [ist425496]> select
    -> name, VAT, address_street, address_city, address_zip
    -> from person natural join client
    -> where VAT not in (select a.VAT from animal a);
+---------------+----------+----------------+--------------+-------------+
| name          | VAT      | address_street | address_city | address_zip |
+---------------+----------+----------------+--------------+-------------+
| Harry Spencer | 00000009 | Big Avenue     | Brooklyn     | 1001-436    |
| Jack Lawson   | 00000010 | Third Street   | California   | 1070-089    |
| Noah Taylor   | 00000017 | Small Avenue   | Fort Worth   | 1356-167    |
| Joseph Davis  | 00000020 | Third Street   | Fort Worth   | 1356-738    |
+---------------+----------+----------------+--------------+-------------+
4 rows in set (0.06 sec)
```

Figura 4: Resultado da *query* 4

3.5 Medicamentos por diagnóstico

```
select code, diagnosis_code.name as diagnosis, count(distinct
    name_med) as counter
from prescription right outer join diagnosis_code using(code)
group by code
order by counter asc;
```

Código 7: *Query* 5

Apresentam-se abaixo os resultados desta *query*. Comprova-se que existe ordem ascendente na apresentação dos resultados. Foram testados (e passados) diversos casos entre os quais existir um diagnóstico sem prescrições associadas (*Flu* e *Broken Nose*) e existirem duas prescrições com dosagens diferentes do mesmo medicamento (*Cron disease*).

```
MySQL [ist425496]> select
    -> code, diagnosis_code.name as diagnosis, count(distinct name_med) as counter
    -> from prescription right outer join diagnosis_code using(code)
    -> group by code
    -> order by counter asc;
+------+--------------------+---------+
| code | diagnosis          | counter |
+------+--------------------+---------+
| 3333 | Flu                |       0 |
| 5555 | Broken Nose        |       0 |
| 2222 | Cron disease       |       1 |
| 4444 | Fever              |       1 |
| 1111 | Kidney Failure     |       2 |
| 0000 | High Blood Pressure |       3 |
+------+--------------------+---------+
6 rows in set (0.01 sec)
```

Figura 5: Resultado da *query* 5

3.6 Valores médios de 2017

```
select
  count(distinct par.name, par.VAT_owner, par.date_timestamp,
     par.VAT_assistant)/count(distinct con.name, con.VAT_owner,
     con.date_timestamp) as average_assistants,
  count(distinct proced.name, proced.VAT_owner,
     proced.date_timestamp, proced.num)/count(distinct con.name,
     con.VAT_owner, con.date_timestamp) as average_procedures,
  count(distinct cd.code, cd.name, cd.VAT_owner,
     cd.date_timestamp)/count(distinct con.name, con.VAT_owner,
     con.date_timestamp) as average_diagnostic_codes,
  count(distinct p.name, p.VAT_owner, p.date_timestamp, p.code,
     p.name_med, p.lab, p.dosage)/count(distinct con.name,
     con.VAT_owner, con.date_timestamp) as average_prescriptions
from (consult as con)
natural left outer join (participation as par)
natural left outer join (consult_diagnosis as cd)
natural left outer join (prescription as p)
natural left outer join proced
where YEAR(date_timestamp) = 2017;
```

Código 8: *Query* 6

Na figura abaixo encontramos os resultados desta *query* comprovando-se assim que executa o que é pedido. É de notar que todas as consultas de 2017 entram para as médias incluindo aquelas que não contém assistentes/procedimentos/diagnósticos/prescrições (entrando com o valor zero).

```
MySQL [ist425496]> select
    ->   count(distinct par.name, par.VAT_owner, par.date_timestamp, par.VAT_assistant)
    ->   /count(distinct con.name, con.VAT_owner, con.date_timestamp) as average_assistants,
    ->   count(distinct proced.name, proced.VAT_owner, proced.date_timestamp, proced.num)
    ->   /count(distinct con.name, con.VAT_owner, con.date_timestamp) as average_procedures,
    ->   count(distinct cd.code, cd.name, cd.VAT_owner, cd.date_timestamp)
    ->   /count(distinct con.name, con.VAT_owner, con.date_timestamp) as average_diagnostic_codes,
    ->   count(distinct p.name, p.VAT_owner, p.date_timestamp, p.code, p.name_med, p.lab, p.dosage)
    ->   /count(distinct con.name, con.VAT_owner, con.date_timestamp) as average_prescriptions
    -> from (consult as con)
    -> natural left outer join (participation as par)
    -> natural left outer join (consult_diagnosis as cd)
    -> natural left outer join (prescription as p)
    -> natural left outer join proced
    -> where YEAR(date_timestamp) = 2017;
+--------------------+--------------------+--------------------------+----------------------+
| average_assistants | average_procedures | average_diagnostic_codes | average_prescriptions |
+--------------------+--------------------+--------------------------+----------------------+
|             1.6000 |             1.4000 |                   1.2000 |               0.6000 |
+--------------------+--------------------+--------------------------+----------------------+
1 row in set (0.01 sec)
```

Figura 6: Resultado da *query* 6

Realça-se que se testou casos como o de haver uma *consult_diagnosis* sem uma *prescription*.

3.7 Doenças mais comuns por raça de cães

```sql
select name1 as species, diagnosis_code.name as diagnosis
from (generalization_species as g)
inner join animal on species_name = name1
inner join consult_diagnosis on consult_diagnosis.name = animal.name
   and consult_diagnosis.VAT_owner = animal.VAT
inner join diagnosis_code using(code)
where name2 = 'dog' group by name1, diagnosis_code.name
having count(*) >= all(
  select count(*)
  from animal as a
  inner join (consult_diagnosis as cd) on cd.name = a.name and
     cd.VAT_owner = a.VAT
  inner join (diagnosis_code as dc) using(code)
  where a.species_name=g.name1 group by dc.name);
```

Código 9: *Query* 7

O que se faz dentro de parênteses do `all` é contar o número de códigos existentes para cada nome de diagnóstico e para a espécie em questão. Depois, para essa mesma espécie, vê-se qual é o nome do diagnóstico que contém mais ocorrências. Considerou-se que, em caso de haver dois diagnósticos com o mesmo número máximo de ocorrências, se mostram ambos os resultados.

Encontram-se na imagem abaixo os resultados que foram obtidos para esta *query*. Os casos de *Akita* e *Pug* foram diretos, apenas existindo um animal desta raça na base de dados e com apenas uma consulta e um diagnóstico. Para os restantes, os resultados estão de acordo com os dados inseridos.

```
MySQL [ist425496]> select
    -> name1 as species, diagnosis_code.name as diagnosis
    -> from (generalization_species as g)
    -> inner join animal on species_name = name1
    -> inner join consult_diagnosis
    -> on consult_diagnosis.name = animal.name
    -> and consult_diagnosis.VAT_owner = animal.VAT
    -> inner join diagnosis_code using(code)
    -> where name2 = 'dog' group by name1, diagnosis_code.name
    -> having count(*) >= all(
    ->   select count(*)
    ->   from animal as a
    ->   inner join (consult_diagnosis as cd)
    ->   on cd.name = a.name and cd.VAT_owner = a.VAT
    ->   inner join (diagnosis_code as dc) using(code)
    ->   where a.species_name=g.name1 group by dc.name);
+-------------+--------------------+
| species     | diagnosis          |
+-------------+--------------------+
| Afghan Hound | High Blood Pressure |
| Akita       | Broken Nose        |
| Bulldog     | High Blood Pressure |
| Bulldog     | Kidney Failure     |
| Husky       | Cron disease       |
| Pug         | Flu                |
| Rottweiler  | Broken Nose        |
| Rottweiler  | Fever              |
+-------------+--------------------+
8 rows in set (0.01 sec)
```

Figura 7: Resultado da *query* 7

3.8 Clientes empregados

```sql
select name
from person natural join client
where VAT in(
  select VAT from veterinary
  union
  select VAT from assistant);
```

Código 10: *Query* 8

Na figura abaixo podem ser observados os resultados desta *query*.

```
MySQL [ist425496]> select name
    -> from person natural join client
    -> where VAT in(
    ->   select VAT from veterinary
    ->   union
    ->   select VAT from assistant);
+---------------+
| name          |
+---------------+
| John Smith    |
| Harry Spencer |
| Jack Lawson   |
+---------------+
3 rows in set (0.02 sec)
```

Figura 8: Resultado da *query* 8

3.9 Clientes com pássaros

```sql
select distinct p.name, address_zip, address_city, address_street
from person as p inner join animal using(VAT)
where p.vat not in(
  select VAT
  from animal
```

```
where species_name not like '%bird%');
```

Código 11: *Query* 9

Na figura abaixo estão apresentados os resultados obtidos. De modo a comprovar a *query*, testou-se com *owners* que possuem *birds* mas também uma outra espécie de animal. Correu como esperado.

```
MySQL [ist425496]> select distinct
    -> p.name, address_zip, address_city, address_street
    -> from person as p inner join animal using(VAT)
    -> where p.vat not in(
    ->   select VAT
    ->   from animal
    ->   where species_name not like '%bird%');
+---------------+-------------+--------------+----------------+
| name          | address_zip | address_city | address_street |
+---------------+-------------+--------------+----------------+
| Oliver Watts  | 1050-375    | Los Angeles  | Big Avenue     |
| Kelly Jenkins | 1010-348    | Cleveland    | Main Street    |
+---------------+-------------+--------------+----------------+
2 rows in set (0.01 sec)
```

Figura 9: Resultado da *query* 9

4 Índices

Por se ter poucos registos na base de dados, o *MySql* opta por implementar os índices em B^+-*tree*. No entanto, para justificar se a solução mais adequada é realizada com o tipo *Hash* ou com B^+-*tree* assume-se que o número de registos da base de dados é elevado.

4.1 Procura dos *owners* cujos animais tiveram consulta com *John Smith*

Para acelerar o processo de procura na *query* 1, criou-se um índice secundário simples:

```
create index idx_person_name on person(name);
```

Código 12: Índice para a primeira *query*

Este índice é simples pois faz referência apenas a uma coluna da tabela. Considerou-se ainda associar um índice a `VAT_vet` em `consult`, no entanto, como se pode verificar na seguinte figura, este já existe.

```
MySQL [ist425496]> show index from consult;
+---------+------------+------------+--------------+---------------+-----------+-------------+----------+--------+------+------------+
| Table   | Non_unique | Key_name   | Seq_in_index | Column_name   | Collation | Cardinality | Sub_part | Packed | Null | Index_type |
+---------+------------+------------+--------------+---------------+-----------+-------------+----------+--------+------+------------+
| consult |          0 | PRIMARY    |            1 | name          | A         |          28 |     NULL | NULL   |      | BTREE      |
| consult |          0 | PRIMARY    |            2 | VAT_owner     | A         |          28 |     NULL | NULL   |      | BTREE      |
| consult |          0 | PRIMARY    |            3 | date_timestamp| A         |          28 |     NULL | NULL   |      | BTREE      |
| consult |          1 | VAT_client |            1 | VAT_client    | A         |          28 |     NULL | NULL   | YES  | BTREE      |
| consult |          1 | VAT_vet    |            1 | VAT_vet       | A         |          14 |     NULL | NULL   | YES  | BTREE      |
+---------+------------+------------+--------------+---------------+-----------+-------------+----------+--------+------+------------+
5 rows in set (0.02 sec)
```

Figura 10: Índices de `consult`

Uma vez que o índice secundário criado é simples e utilizado para igualdades, numa situação de existência de muitos registos optar-se-ia por implementar uma solução do tipo *Hash*, assumindo que existe uma boa função de dispersão. Para uma única igualdade (com índice simples) a *Hash* é a melhor solução pois não é necessário gastar tempo a ordenar (algo utilizado na B^+-*tree*), e consegue-se (através da função de dispersão) aceder diretamente ao *bucket* onde o valor desejado se encontra. Depois de encontrar o *bucket*, é necessário percorrê-lo até encontrar o valor, caso tenha havido colisões (idealmente contém poucos valores, algo que depende da função de dispersão e dos registos). Assim a complexidade média na procura é

$\mathcal{O}(1)$. A implementação deste índice com o tipo B^+-*tree* não é tão adequada pois podiam ser necessárias bastantes comparações até se encontrar o valor desejado. A melhoria pode ser verificada através do comando `explain` antes e depois da criação do índice.

```
MySQL [ist425496]> explain select distinct
    -> animal.name, O.name as owner_name, species_name, animal.age
    -> from animal
    -> inner join consult
    -> on animal.VAT = consult.VAT_owner
    -> and animal.name = consult.name
    -> inner join person V on V.VAT = VAT_vet
    -> inner join person O on O.VAT = consult.VAT_owner
    -> where V.name = 'John Smith';
+----+-------------+---------+-------+---------------+---------+---------+-----------------------------------------+------+
| id | select_type | table   | type  | possible_keys | key     | key_len | ref                                     | rows |
+----+-------------+---------+-------+---------------+---------+---------+-----------------------------------------+------+
|  1 | SIMPLE      | animal  | ALL   | PRIMARY,VAT   | NULL    | NULL    | NULL                                    |   19 |
|  1 | SIMPLE      | O       | eq_ref | PRIMARY      | PRIMARY | 22      | ist425496.animal.VAT                    |    1 |
|  1 | SIMPLE      | consult | ref   | PRIMARY,VAT_vet | PRIMARY | 279   | ist425496.animal.name,ist425496.animal.VAT |    1 |
|  1 | SIMPLE      | V       | eq_ref | PRIMARY      | PRIMARY | 22      | ist425496.consult.VAT_vet               |    1 |
+----+-------------+---------+-------+---------------+---------+---------+-----------------------------------------+------+
4 rows in set (0.01 sec)

MySQL [ist425496]> create index idx_person_name on person(name);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

MySQL [ist425496]> explain select distinct
    -> animal.name, O.name as owner_name, species_name, animal.age
    -> from animal
    -> inner join consult
    -> on animal.VAT = consult.VAT_owner
    -> and animal.name = consult.name
    -> inner join person V on V.VAT = VAT_vet
    -> inner join person O on O.VAT = consult.VAT_owner
    -> where V.name = 'John Smith';
+----+-------------+---------+--------+-----------------------+----------------+---------+-----------------------------------------------+------+
| id | select_type | table   | type   | possible_keys         | key            | key_len | ref                                           | rows |
+----+-------------+---------+--------+-----------------------+----------------+---------+-----------------------------------------------+------+
|  1 | SIMPLE      | V       | ref    | PRIMARY,idx_person_name | idx_person_name | 257   | const                                         |    3 |
|  1 | SIMPLE      | consult | ref    | PRIMARY,VAT_vet       | VAT_vet        | 23      | ist425496.V.VAT                               |    2 |
|  1 | SIMPLE      | O       | eq_ref | PRIMARY               | PRIMARY        | 22      | ist425496.consult.VAT_owner                   |    1 |
|  1 | SIMPLE      | animal  | eq_ref | PRIMARY,VAT           | PRIMARY        | 279     | ist425496.consult.name,ist425496.consult.VAT_owner |    1 |
+----+-------------+---------+--------+-----------------------+----------------+---------+-----------------------------------------------+------+
4 rows in set (0.01 sec)
```

Figura 11: Comparação dos acessos às tabelas

4.2 Procura de valores de referências superiores a 100 miligramas

Para o caso da *query* 2, usou-se um índice composto secundário:

```
create index idx_indicator_units_reference_value on indicator(units,
    reference_value);
```

Código 13: Índice para a segunda *query*

Uma vez que se faz uma procura a dois atributos diferentes de uma só tabela através de um `and`, o mais apropriado é a criação de um índice composto. A ordem com que se coloca é algo importante a ser considerado, pois se quer aceder ordenadamente aos `reference_value` dentro de uma certa `unit`. Assim sendo, coloca-se primeiro o atributo da igualdade, e em segundo o do *range*, pois se pretende procurar os que têm uma referência >100 dentro dos que têm as unidades miligramas (sendo assim escusado fazer uma procura do valor de referência em gramas, quilogramas, etc). Caso fosse um `and` com duas igualdades, colocar-se-ia primeiro o atributo que tem menos variedade de valores de modo a minimizar o número de iterações.

Para este caso, o mais apropriado é uma B^+-*tree*. O tipo *Hash* não é apropriado para índices compostos nem para encontrar *ranges*. O tipo B^+-*tree*, ao contrário da *Hash*, permite ordenar e agrupar os dados pelo que apenas é necessário encontrar o primeiro registo com unidades `milligrams` e valor de referência superior a 100 e varrer até ao último com a mesma unidade. A melhoria é clara quando analisado, novamente, com o comando `explain`.

```
MySQL [ist425496]> explain select name, reference_value
    -> from indicator
    -> where units = 'milligrams' and reference_value > 100
    -> order by reference_value desc;
+----+-------------+-----------+------+---------------+------+---------+------+------+
| id | select_type | table     | type | possible_keys | key  | key_len | ref  | rows |
+----+-------------+-----------+------+---------------+------+---------+------+------+
|  1 | SIMPLE      | indicator | ALL  | NULL          | NULL | NULL    | NULL |    7 |
+----+-------------+-----------+------+---------------+------+---------+------+------+
1 row in set (0.01 sec)

MySQL [ist425496]> create index idx_indicator_units_reference_value on indicator(units, reference_value);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

MySQL [ist425496]> explain select name, reference_value
    -> from indicator
    -> where units = 'milligrams' and reference_value > 100
    -> order by reference_value desc;
+----+-------------+-----------+-------+-----------------------------------+-----------------------------------+---------+------+------+
| id | select_type | table     | type  | possible_keys                     | key                               | key_len | ref  | rows |
+----+-------------+-----------+-------+-----------------------------------+-----------------------------------+---------+------+------+
|  1 | SIMPLE      | indicator | range | idx_indicator_units_reference_value | idx_indicator_units_reference_value | 262     | NULL |    3 |
+----+-------------+-----------+-------+-----------------------------------+-----------------------------------+---------+------+------+
1 row in set (0.01 sec)
```

Figura 12: Comparação dos acessos às tabelas

5 *Views*

5.1 dim_date

```
create or replace view dim_date as(
 select distinct date_timestamp,
 DAY(date_timestamp) as day,
 MONTH(date_timestamp) as month,
 YEAR(date_timestamp) as year
 from consult
);
```

Código 14: *View* 1

```
MySQL [ist425496]> create or replace view dim_date as(
    -> select distinct date_timestamp,
    -> DAY(date_timestamp) as day,
    -> MONTH(date_timestamp) as month,
    -> YEAR(date_timestamp) as year
    -> from consult
    -> );
Query OK, 0 rows affected (0.01 sec)

MySQL [ist425496]> select * from dim_date;
+---------------------+------+-------+------+
| date_timestamp      | day  | month | year |
+---------------------+------+-------+------+
| 2005-10-29 15:30:00 |   29 |    10 | 2005 |
| 2016-11-30 09:00:00 |   30 |    11 | 2016 |
| 2005-03-29 09:00:00 |   29 |     3 | 2005 |
| 2017-03-29 10:30:00 |   29 |     3 | 2017 |
| 2005-07-29 09:00:00 |   29 |     7 | 2005 |
| 2018-01-10 11:00:00 |   10 |     1 | 2018 |
| 2018-01-10 12:30:00 |   10 |     1 | 2018 |
| 2018-01-10 14:00:00 |   10 |     1 | 2018 |
| 2005-10-29 16:30:00 |   29 |    10 | 2005 |
| 2017-10-29 09:30:00 |   29 |    10 | 2017 |
| 2005-01-29 09:00:00 |   29 |     1 | 2005 |
| 2005-02-27 09:00:00 |   27 |     2 | 2005 |
| 2018-02-27 09:00:00 |   27 |     2 | 2018 |
| 2005-09-29 12:30:00 |   29 |     9 | 2005 |
| 2005-07-31 09:00:00 |   31 |     7 | 2005 |
| 2005-08-29 09:00:00 |   29 |     8 | 2005 |
| 2016-07-28 11:00:00 |   28 |     7 | 2016 |
| 2016-07-29 11:45:00 |   29 |     7 | 2016 |
| 2018-07-30 09:00:00 |   30 |     7 | 2018 |
| 2016-11-29 09:00:00 |   29 |    11 | 2016 |
| 2016-11-29 19:00:00 |   29 |    11 | 2016 |
| 2017-07-27 10:30:00 |   27 |     7 | 2017 |
| 2017-08-27 09:00:00 |   27 |     8 | 2017 |
| 2017-08-29 11:30:00 |   29 |     8 | 2017 |
| 2005-09-29 17:40:00 |   29 |     9 | 2005 |
+---------------------+------+-------+------+
25 rows in set (0.01 sec)
```

Figura 13: Resultado da *view* 1

De modo a que *date_timestamp* seja chave primária, colocou-se um `distinct`. Desta forma, não podem aparecer duas datas iguais.

5.2 `dim_animal`

```
create or replace view dim_animal as(
 select name as animal_name,
 VAT as animal_vat,
 species_name as species,
 age
 from animal
);
```

Código 15: *View* 2

Ao contrário da *view* anterior, para este caso não foi necessário colocar nenhum `distinct` para definir as chaves primárias, uma vez que são chaves estrangeiras (das únicas primárias) de animal, garantindo-se assim que não há repetições.

```
MySQL [ist425496]> create or replace view dim_animal as(
    -> select name as animal_name,
    -> VAT as animal_vat,
    -> species_name as species,
    -> age
    -> from animal
    -> );
Query OK, 0 rows affected (0.06 sec)

MySQL [ist425496]> select * from dim_animal;
+-------------+------------+----------------+-----+
| animal_name | animal_vat | species        | age |
+-------------+------------+----------------+-----+
| Blue        | 00000002   | Bird           |   1 |
| Blue        | 00000003   | Bird           |   1 |
| Bolt        | 00000007   | Bird           |  10 |
| Bonzo       | 00000002   | Rottweiler     |  12 |
| Cally       | 00000014   | Parakeet bird  |  17 |
| Doggy       | 00000014   | Dog            |  16 |
| Fluffy      | 00000002   | Akita          |  13 |
| Foxy        | 00000005   | Bulldog        |   2 |
| Garfield    | 00000005   | Cat            |   3 |
| Goofy       | 00000005   | Husky          |   2 |
| Marlie      | 00000006   | Husky          |  11 |
| Nugget      | 00000005   | Rottweiler     |   3 |
| Puma        | 00000000   | Bulldog        |  20 |
| Severin     | 00000015   | Pug            |  15 |
| Spark       | 00000003   | Cockatiel bird |   9 |
| Theo        | 00000000   | Siamese        |  17 |
| Toby        | 00000006   | Husky          |   1 |
| Trojan      | 00000006   | Fish           |   1 |
| Wanikiy     | 00000015   | Afghan Hound   |  14 |
+-------------+------------+----------------+-----+
19 rows in set (0.01 sec)
```

Figura 14: Resultado da *view* 2

5.3 `facts_concults`

```
create or replace view facts_concults as(
 select dim_animal.animal_name as name,
 dim_animal.animal_vat as vat,
 dim_date.date_timestamp as timestamp,
 count(distinct proced.name, proced.VAT_owner, proced.date_timestamp,
    proced.num)
 as num_procedures,
 count(distinct p.name, p.VAT_owner, p.date_timestamp, p.code,
    p.name_med, p.lab, p.dosage)
```

```
 as num_medication
 from consult inner join dim_animal
 on consult.name = dim_animal.animal_name and consult.VAT_owner =
    dim_animal.animal_vat
 natural join dim_date
 natural left outer join proced
 natural left join (prescription as p)
 group by dim_date.date_timestamp, name, VAT_owner
);
```

Código 16: *View* 3

Como se pode ver pelos resultados, consegue-se garantir as chaves primárias pretendidas. Refere-se que se consideram todos os medicamentos prescritos, mesmo que sejam iguais, pois são passados em consultas diferentes.

```
MySQL [ist425496]> create or replace view facts_concults as(
    ->  select dim_animal.animal_name as name,
    ->  dim_animal.animal_vat as vat,
    ->  dim_date.date_timestamp as timestamp,
    ->  count(distinct proced.name, proced.VAT_owner, proced.date_timestamp, proced.num)
    ->  as num_procedures,
    ->  count(distinct p.name, p.VAT_owner, p.date_timestamp, p.code, p.name_med, p.lab, p.dosage)
    ->  as num_medication
    ->  from consult inner join dim_animal
    ->  on consult.name = dim_animal.animal_name and consult.VAT_owner = dim_animal.animal_vat
    ->  natural join dim_date
    ->  natural left outer join proced
    ->  natural left join (prescription as p)
    ->  group by dim_date.date_timestamp, name, VAT_owner
    -> );
Query OK, 0 rows affected (0.01 sec)

MySQL [ist425496]> select * from facts_concults;
+----------+----------+---------------------+---------------+----------------+
| name     | vat      | timestamp           | num_procedures | num_medication |
+----------+----------+---------------------+---------------+----------------+
| Cally    | 00000014 | 2005-01-29 09:00:00 |             0 |              0 |
| Wanikiy  | 00000015 | 2005-02-27 09:00:00 |             0 |              0 |
| Bonzo    | 00000002 | 2005-03-29 09:00:00 |             0 |              0 |
| Bolt     | 00000007 | 2005-07-29 09:00:00 |             0 |              0 |
| Fluffy   | 00000002 | 2005-07-29 09:00:00 |             0 |              0 |
| Garfield | 00000005 | 2005-07-29 09:00:00 |             0 |              0 |
| Severin  | 00000015 | 2005-07-31 09:00:00 |             0 |              0 |
| Severin  | 00000015 | 2005-08-29 09:00:00 |             0 |              0 |
| Spark    | 00000003 | 2005-08-29 09:00:00 |             0 |              0 |
| Doggy    | 00000014 | 2005-09-29 12:30:00 |             2 |              1 |
| Toby     | 00000006 | 2005-09-29 17:40:00 |             0 |              1 |
| Theo     | 00000000 | 2005-10-29 15:30:00 |             0 |              1 |
| Trojan   | 00000006 | 2005-10-29 16:30:00 |             0 |              1 |
| Severin  | 00000015 | 2016-07-28 11:00:00 |             0 |              0 |
| Severin  | 00000015 | 2016-07-29 11:45:00 |             0 |              0 |
| Blue     | 00000003 | 2016-11-29 09:00:00 |             0 |              0 |
| Marlie   | 00000006 | 2016-11-29 19:00:00 |             0 |              0 |
| Blue     | 00000002 | 2016-11-30 09:00:00 |             0 |              0 |
| Bonzo    | 00000002 | 2017-03-29 10:30:00 |             1 |              0 |
| Puma     | 00000000 | 2017-07-27 10:30:00 |             3 |              2 |
| Puma     | 00000000 | 2017-08-27 09:00:00 |             1 |              1 |
| Spark    | 00000003 | 2017-08-29 11:30:00 |             0 |              0 |
| Trojan   | 00000006 | 2017-10-29 09:30:00 |             2 |              0 |
| Foxy     | 00000005 | 2018-01-10 11:00:00 |             0 |              1 |
| Goofy    | 00000005 | 2018-01-10 12:30:00 |             0 |              0 |
| Nugget   | 00000005 | 2018-01-10 14:00:00 |             0 |              0 |
| Wanikiy  | 00000015 | 2018-02-27 09:00:00 |             0 |              0 |
| Severin  | 00000015 | 2018-07-30 09:00:00 |             0 |              0 |
+----------+----------+---------------------+---------------+----------------+
28 rows in set (0.01 sec)
```

Figura 15: Resultado da *view* 3

6 *Updates*

6.1 Mudança de morada

```
update person natural join client
set address_street='Rua da Bela Vista', address_city='Lisboa',
   address_zip='2695'
where name='John Smith';
```

Código 17: *Update* 1

Como se pode verificar, apenas as entradas correspondentes a clientes *John Smith* são alteradas, permanecendo o veterinário inalterado.

```
MySQL [ist425496]> update person natural join client
    -> set address_street='Rua da Bela Vista', address_city='Lisboa', address_zip='2695'
    -> where name='John Smith';
Query OK, 2 rows affected (0.01 sec)
Rows matched: 2  Changed: 2  Warnings: 0

MySQL [ist425496]> select * from person;
+----------+------------------+--------------------+--------------+-------------+
| VAT      | name             | address_street     | address_city | address_zip |
+----------+------------------+--------------------+--------------+-------------+
| 00000000 | John Smith       | Rua da Bela Vista  | Lisboa       | 2695        |
| 00000001 | John Smith       | Second Street      | Brooklyn     | 1001-439    |
| 00000002 | John Doe         | Main Street        | Cleveland    | 1010-798    |
| 00000003 | Oliver Watts     | Big Avenue         | Los Angeles  | 1050-375    |
| 00000004 | Chandler Webbs   | Small Avenue       | Brooklyn     | 1001-373    |
| 00000005 | John Smith       | Rua da Bela Vista  | Lisboa       | 2695        |
| 00000006 | William Lawrence | Second Street      | Brooklyn     | 1001-438    |
| 00000007 | Kelly Jenkins    | Main Street        | Cleveland    | 1010-348    |
| 00000008 | Jacob Chambers   | Second Street      | Brooklyn     | 1001-230    |
| 00000009 | Harry Spencer    | Big Avenue         | Brooklyn     | 1001-436    |
| 00000010 | Jack Lawson      | Third Street       | California   | 1070-089    |
| 00000011 | Rhys Woods       | Main Street        | Dallas       | 1100-324    |
| 00000012 | Thomas Ress      | Big Avenue         | Chicago      | 1078-375    |
| 00000013 | George Fraser    | Small Street       | Brooklyn     | 1001-850    |
| 00000014 | Damian Black     | Big Avenue         | Dallas       | 1100-279    |
| 00000015 | Joe Fletcher     | Third Street       | Houston      | 1073-384    |
| 00000016 | Jones            | Small Avenue       | Brooklyn     | 1001-379    |
| 00000017 | Noah Taylor      | Small Avenue       | Fort Worth   | 1356-167    |
| 00000019 | Ethan Brown      | Second Street      | San Diego    | 1263-368    |
| 00000020 | Joseph Davis     | Third Street       | Fort Worth   | 1356-738    |
+----------+------------------+--------------------+--------------+-------------+
20 rows in set (0.01 sec)
```

Figura 16: Resultado do *update* 1

6.2 Mudança de valores de referência

```
update test_procedure
  natural join produced_indicator
  inner join indicator on produced_indicator.indicator_name =
     indicator.name
set reference_value = reference_value * 1.1
where units = 'milligrams' and type = 'blood';
```

Código 18: *Update* 2

Note-se que *cholesterol* e *fatty acids* são medidos em *milligrams* mas não estão associado a nenhum procedimento do tipo *blood*, pelo que a única alteração é de *white blood cells* de 100 para 110.

```
MySQL [ist425496]> select * from indicator;
+------------------+-----------------+-------------------------+-----------------------------------------------------------------------------------------------+
| name             | reference_value | units                   | description                                                                                   |
+------------------+-----------------+-------------------------+-----------------------------------------------------------------------------------------------+
| Cholesterol      |          200.00 | milligrams              | Organic molecule.                                                                             |
| Creatinine level |            0.60 | milligrams per deciliter | It is a breakdown product of creatine phosphate in muscle.                                    |
| Erythrocytes     |          500.00 | grams                   | Cells of the immune system that protected the body against foreign invaders.                  |
| Fatty acids      |          110.00 | milligrams              | A carboxylic acid with a long aliphatic chain, which is either saturated or unsaturated.      |
| Insulin          |          350.00 | milligrams              | It is a peptide hormone produced by beta cells of the pancreatic islets.                      |
| Red Blood Cells  |           50.00 | grams                   | Cells that deliverer oxygen (O2) to the body tissues                                          |
| White Blood Cells |         100.00 | milligrams              | Cells of the immune system that protected the body against foreign invaders.                  |
+------------------+-----------------+-------------------------+-----------------------------------------------------------------------------------------------+
7 rows in set (0.00 sec)
```

Figura 17: `indicator` antes do *update* 2

```
MySQL [ist425496]> update test_procedure
    ->   natural join produced_indicator
    ->   inner join indicator  on produced_indicator.indicator_name=indicator.name
    -> set reference_value= reference_value*1.1
    -> where units='milligrams' and type='blood';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MySQL [ist425496]> select * from indicator;
+------------------+-----------------+-------------------------+-----------------------------------------------------------------------------------------------+
| name             | reference_value | units                   | description                                                                                   |
+------------------+-----------------+-------------------------+-----------------------------------------------------------------------------------------------+
| Cholesterol      |          200.00 | milligrams              | Organic molecule.                                                                             |
| Creatinine       |            0.60 | milligrams per deciliter | It is a breakdown product of creatine phosphate in muscle.                                    |
| Erythrocytes     |          500.00 | grams                   | Cells of the immune system that protected the body against foreign invaders.                  |
| Fatty acids      |          110.00 | milligrams              | A carboxylic acid with a long aliphatic chain, which is either saturated or unsaturated.      |
| Insulin          |          350.00 | milligrams              | It is a peptide hormone produced by beta cells of the pancreatic islets.                      |
| Red Blood Cells  |           50.00 | grams                   | Cells that deliverer oxygen (O2) to the body tissues                                          |
| White Blood Cells |         110.00 | milligrams              | Cells of the immune system that protected the body against foreign invaders.                  |
+------------------+-----------------+-------------------------+-----------------------------------------------------------------------------------------------+
7 rows in set (0.25 sec)
```

Figura 18: `indicator` depois do *update* 2

6.3 Apagar o cliente *John Smith*

```
delete from client where VAT in(select VAT from person where name =
    'John Smith');
```

Código 19: *Update* 3

Realça-se que só se tem de eliminar os registos quando o *John Smith* é cliente (não se alterando os registos quando este é veterinário ou assistente). Além disso, foi necessário colocar um `on delete cascade` em tudo o que depende da entidade `client`.

```
MySQL [ist425496]> delete from client where VAT in(select VAT from person where name = 'John Smith');
Query OK, 2 rows affected (0.01 sec)

MySQL [ist425496]> select * from client;
+----------+
| VAT      |
+----------+
| 00000002 |
| 00000003 |
| 00000006 |
| 00000007 |
| 00000009 |
| 00000010 |
| 00000014 |
| 00000015 |
| 00000017 |
| 00000020 |
+----------+
10 rows in set (0.01 sec)
```

Figura 19: Resultado do *update* 3

.

6.4 Doença agravada

```
update (consult_diagnosis natural join test_procedure natural join
    produced_indicator)
set code = (select code from diagnosis_code where name = 'end-stage
    renal disease'),
```

```
  date_timestamp = date_timestamp,
  name = name,
  VAT_owner = VAT_owner
where code = (select code from diagnosis_code where name = 'kidney
   failure')
  and type = 'blood'
  and indicator_name = 'creatinine level'
  and value > 1;
```

Código 20: *Update* 4

Para esta situação, teve de se renovar as restantes *Primary keys* para que o *MySql* aceitasse a alteração. Considerou-se ainda que o código de *end-stage renal disease* já se encontrava na base de dados, pois o seu código é estandardizado, isto é, faz parte de uma lista conhecida. Caso isto não fosse assumido, inserir-se-ia o valor antes do *update* se não existisse. Teve ainda de se adicionar no *script* de criação de tabelas um `on update cascade` para que esta entrada pudesse ser alterada sem modificar fortemente as prescrições a si relacionadas, que podem não ter necessariamente de ser apagadas. Pode-se verificar que o diagnóstico pretendido de *Puma* foi alterado para o de *end-stage renal disease*.

```
MySQL [ist425496]> update (consult_diagnosis natural join test_procedure natural join produced_indicator)
    -> set code = (select code from diagnosis_code where name = 'end-stage renal disease'),
    ->   date_timestamp = date_timestamp,
    ->   name = name,
    ->   VAT_owner = VAT_owner
    -> where code = (select code from diagnosis_code where name = 'kidney failure')
    ->   and type = 'blood'
    ->   and indicator_name = 'creatinine level'
    ->   and value > 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MySQL [ist425496]> select * from consult_diagnosis;
+------+---------+-----------+---------------------+
| code | name    | VAT_owner | date_timestamp      |
+------+---------+-----------+---------------------+
| 4444 | Bonzo   | 00000002  | 2017-03-29 10:30:00 |
| 5555 | Bonzo   | 00000002  | 2017-03-29 10:30:00 |
| 1111 | Doggy   | 00000014  | 2005-09-29 12:30:00 |
| 5555 | Fluffy  | 00000002  | 2005-07-29 09:00:00 |
| 0000 | Foxy    | 00000005  | 2018-01-10 11:00:00 |
| 1111 | Foxy    | 00000005  | 2018-01-10 11:00:00 |
| 2222 | Goofy   | 00000005  | 2018-01-10 12:30:00 |
| 2222 | Marlie  | 00000006  | 2016-11-29 19:00:00 |
| 3333 | Marlie  | 00000006  | 2016-11-29 19:00:00 |
| 4444 | Nugget  | 00000005  | 2018-01-10 14:00:00 |
| 5555 | Nugget  | 00000005  | 2018-01-10 14:00:00 |
| 2222 | Puma    | 00000000  | 2017-07-27 10:30:00 |
| 6666 | Puma    | 00000000  | 2017-07-27 10:30:00 |
| 0000 | Puma    | 00000000  | 2017-08-27 09:00:00 |
| 3333 | Severin | 00000015  | 2016-07-28 11:00:00 |
| 3333 | Spark   | 00000003  | 2017-08-29 11:30:00 |
| 2222 | Theo    | 00000000  | 2005-10-29 15:30:00 |
| 0000 | Toby    | 00000006  | 2005-09-29 17:40:00 |
| 4444 | Toby    | 00000006  | 2005-09-29 17:40:00 |
| 0000 | Trojan  | 00000006  | 2005-10-29 16:30:00 |
| 0000 | Wanikiy | 00000015  | 2005-02-27 09:00:00 |
| 1111 | Wanikiy | 00000015  | 2005-02-27 09:00:00 |
| 0000 | Wanikiy | 00000015  | 2018-02-27 09:00:00 |
+------+---------+-----------+---------------------+
23 rows in set (0.01 sec)
```

Figura 20: Resultado do *update* 4