Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Sistemas de Informação para a empresa *Maintain4ver*

Grupo 05:

46054 Bernardo Miguel Mira Camarate

46090 Cátia Marina Soares Abreu

44876 Pedro Henrique Muzachio Castanheira

Relatório para a Unidade Curricular de Sistemsas de Informação 2 da Licenciatura em Engenharia Informática e de Computadores

Professor: Nuno Datia

04 – 12 – 2021

<< Esta página foi intencionalmente deixada em branco >>

Resumo

Neste trabalho pretende-se implementar um sistema de informação para a gestão de manutenção de ativos físicos de uma empresa de manutenção (*Maintain4ver*). Este sistema de informação tem como elementos principais: *skill, intervention, team_member, maintenance_team, register, asset, type, employee*.

A empresa *Maintain4ver* gere um conjunto de ativos, que contém um tipo, mantém-se um registo histórico do valor comercial do ativo. Na manutenção dos ativos físicos existem intervenções. Cada intervenção é realizada por uma equipa de manutenção constituída, no mínimo, por 2 elementos. Um deles é o supervisor. Cada equipa só consegue realizar uma intervenção de cada vez, podendo ter até 3 intervenções atribuídas. A empresa tem vários funcionários, que têm um registo das suas competências.

Em resposta ao problema desenvolvemos um modelo de dados adequado aos requisitos do sistema; foram criados *scripts* de criação, remoção e inserção de dados no modelo físico. Foram ainda implementadas as seguinte funcionalidades: obtenção do código de uma equipa livre, dada uma descrição de intervenção, capaz de resolver o problema; criação de um procedimento que permite criar uma intervenção; implementação de um mecanismo que permite criar uma equipa; actualização dos elementos de uma equipe e associar as respectivas competências; criação uma funçãoo para produzir a listagem das intervenções de um determinado ano; actualização do estado de uma intervenção; criação duma vista que mostra o resumo das intervenções e que possibilita a alteração do estado de uma ou mais intervenções. Para realizar as funcionalidades descritas recorreu-se a funções, procedimentos armazenados, gatilhos e vistas.

A solução adotada tem em conta todos os requisitos do sistema, mas poderia ter sido diferente se fossem assumidos outros modelos de funcionamento.

Índice

1. Introdução 1

1.1 Objetivos e a realização prática dos mesmos 1

2. Formulação do Problema 2

2.1 Elementos essenciais na empresa *Maintain4ver* 2

2.2 Relações entre os componentes na empresa *Maintain4ver* 2

3. Solução proposta 3

3.1 Considerações da solução proposta 3

3.2 Implementação das funcionalidades do sistema de informação 4

3.2.1 Criação do modelo físico 4

3.2.2 Remoção do modelo físico 4

3.2.3 Preenchimento inicial da base de dados 4

3.2.4 Mecanismo que permite inserir, remover e atualizar informação de um funcionário. 5

3.2.4.1 Inserção 5

3.2.4.2 Remoção 5

3.2.4.3 Atualização 5

3.2.5 Obtenção do código de uma equipa livre, dada uma descrição de intervenção, capaz de resolver o problema. 5

3.2.6 Criação do procedimento *p_criaInter* que permite criar uma intervenção 5

3.2.7 Implementação do mecanismo que permite criar uma equipa 6

3.2.8 Actualização dos elementos de uma equipe e associação das respectivas competências 6

3.2.9 Criação duma função para produzir a listagem das intervenções de um determinado ano 6

3.2.10 Actualização do estado de uma intervenção 6

3.2.11 Criação duma vista que mostra o resumo das intervenções que possibilita a alteração do estado de uma ou mais intervenções 7

4. Avaliação experimental 8

4.1 Testes às funcionalidades implementadas 8

4.2 Análise de resultados 8

5. Referências 9

Lista de Tabelas

Tabela 1 employee 11
Tabela 2 type 12
Tabela 3 asset 12
Tabela 4 register 13
Tabela 5 maintenance_team 13
Tabela 6 team_member 14
Tabela 7 intervention 14
Tabela 8 periodic 15
Tabela 9 non_periodic 15
Tabela 10 scheculing 15
Tabela 11 skill 16
Tabela 12 employee_skill 16

1. Introdução

A execução do trabalho tem como principal objetivo a criação de um sistema de informação para a gestão de manutenção de ativos físicos da empresa Maintain4ver e para tal, foi necessária a criação de tabelas para a devida projeção do modelo bem como a criação de procedimentos armazenados, triggers, vistas e funções para execução dos requisitos pedidos.

1.1 Objetivos e a realização prática dos mesmos

Na secção [2] são apresentados os elementos que constituem este sistema e os detalhes de cada elemento de modo a que se possa construir uma solução que englobe os mesmos tendo em atenção as relações que têm entre si.

A solução apresentada na secção [3] para a construção do sistema de informação é feita através da criação do modelo físico.

Na elaboração desta solução foram tidos em conta os diversos objetivos de aprendizagem que são: Desenvolver um modelo de dados adequado aos requisitos do sistema, normalizado até à 3NF; Conceber e implementar uma solução baseada em bases de dados dinâmicas, adequada aos requisitos; Utilizar corretamente controlo transacional; Utilizar corretamente níveis de isolamento; Utilizar corretamente vistas, justificando o seu uso na solução; Utilizar corretamente procedimentos armazenados, justificando o seu uso na solução; Utilizar corretamente gatilhos, justificando o seu uso na solução; Utilizar corretamente funções, justificando o seu uso na solução; Desenvolver código de teste, em T-SQL, para cada uma das funcionalidades requerida nos requisitos;

Para garantir o bom funcionamento do sistema de informação desenvolveu-se testes como é explicado na secção [4].

2. Formulação do Problema

Neste capítulo são referidos os detalhes que devem ser implementados no sistema de informação.

2.1 Elementos essenciais na empresa *Maintain4ver*

Para a gestão de manutenção de ativos físicos de *Maintain4ver* temos como elementos principais: *skill, intervention, team_member, maintenance_team, register, asset, type, employee*.

2.2 Relações entre os componentes na empresa *Maintain4ver*

Cada elemento correlaciona-se com um ou mais elementos. Do enunciado consegue-se extrair as seguintes relações:

- Um ativo pode ser constituido por outros ativos
- O tipo do ativo de topo da hierarquia tem de ser igual ao(s) tipo(s) do(s) ativo(s) "filho(s)".
- A data de intervenção deve ser superior à data de aquisição do ativo intervencionado.
- Equipa de manutenção é constituida, no mínimo, por 2 elementos. Um deles será o supervisor.
- Cada equipa só consegue realizar uma intervenção de cada vez, podendo ter até 3 intervenções atribuídas. Se não for possivel attribuir uma equipa, o estado da intervenção deve ser "por atribuir".
- Existe uma pessoa da empresa que gere o ativo e que não pode participar na equipa que faz a intervenção desse ativo.
- Uma pessoa só pode pertencer a uma equipa.
- A atribuição de uma intervenção a uma equipa so é possivel se a descrição for compativel com as competências da equipa.

3. Solução proposta

Neste capítulo é proposta uma solução para a implementação do sistema de informação. Na secção [3.1] são apresentados os processos para a elaboração do modelo conceptual e físico.

3.1 Considerações da solução proposta

Para desenvolver um modelo de dados adequado aos requisitos impostos começou-se por realizar um modelo Entidade-Associação (EA), presente no anexo [A]. Na concretização do modelo EA foram tomadas algumas decisões de funcionamento do sistema de informação.

Para que as funcionalidades implementadas possam ser realizadas em concorrência tiveram de ser consideradas possíveis falhas dos procedimentos armazenados e triggers, em resposta a este problema foram definidos níveis de isolamento para alguns procedures. Recorremos a dois tipos de níveis de isolamento: *Read Commited* e *Repetable Read*. O nível de isolamento por omissão é *Read Commited*, este protege o código da existência de *Dirty Reads*. Alguns *procedures* consideramos que precisavam de um nível de isolamento *Repeatable Read*, neste nível para além da proteção dado por read committed, este permite que nenhuma outra transação possa modificar os dados da transação currente até esta fazer commit, e evitamo *lost updates.*

Para o modelo físico foram criadas doze tabelas:

EMPLOYEE com os atributos: *ssn, f_name, l_name, birth_date, address, postal_code, city, job, phone_number, mail*. Tendo como chave primária, o ssn, que representa o número de contribuinte do funcionário.

ASSET com os atributos: *id, asset_name, acquisition_date, state, brand, model, location, asset_reference, manager, type*. Tendo como chave primária o id, e como chaves estrangeiras asset_reference que faz referência ao id da tabela ASSET; manager que faz referência a ssn da tabela EMPLOYEE; e type que faz referência a id da tabela TYPE

TYPE com os atributos: *id* e *description*. Tendo como chave primária de identificação o id.

REGISTER com os atributos: *asset_id, alteration_date* e *price*. Tendo como chaves primárias o atributo alteration_date e a chave estrangeira asset_id que faz referência ao id da tabela ASSET.

MAINTENANCE_TEAM com os atributos: *team_code, location, n_elements* e *supervisor*. Tendo como chave primária team_code e como chave estrangeira o atributo supervisor que faz referência a um empregado da tabela EMPLOYEE que fica encarregue desta equipa.

TEAM_MEMBER com os atributos: *id* e *team_code*. Onde a chave primária é composta por estes dois atributos e cada um é uma chave estrangeira que faz referência, respectivamente, a o ssn da tabela EMPLOYEE e team_code da tabela MAINTENANCE_TEAM.

INTERVENTION com os atributos: *intervention_code, description, state, price, start_date, end_date* e *asset_id*. Onde a chave primária é o atributo interventation_code e a chave estrangeira por sua vez é o asset_id que faz referência ao id da tabela ASSET.

PERIODIC com os atributos: *id* e *frequency*. Onde a chave primária e estrangeira é a mesma, sendo esta o id e faz referência ao interventation_code da tabela INTERVENTATION.

NON_PERIODIC com os atributos: *id.* Este único atributo que é chave estrangeira e primária faz referência ao atributo interventatio_code da tabela INTERVENTATION.

SCHEDULING com os atributos: *team_code, interventation_code* e *scheduling_date*. Esta tabela tem como chaves primárias e estrangeiras os atributos team_code e interventation_code que fazem referência, respectivamente, às chaves primárias das tabelas MAINTENANCE_TEAM e INTERVENTATION.

SKILL com os atributos: *id* e *description.* Tendo como chave primária o atributo id.

EMPLOYEE_SKILL com os atributos: *employee_code* e *skill_code*. Em que este dois são a chave primária e estrangeira. Onde employee_code faz referência à chave primária da tabela EMPLOYEE e o skill_code faz referência ao atributo id de SKILL que também é chave primária da tabela em questão.

3.2 Implementação das funcionalidades do sistema de informação

Em seguida, são apresentadas as soluções adotadas para a implementação de cada uma das funcionalidades propostas, assim como as restrições do modelo para cada uma das mesmas.

3.2.1 Criação do modelo físico

Foram criadas de 12 (doze) tabelas (CREATE TABLE) , dentre as quais:

- EMPLOYEE com os atributos: ssn, f_name, l_name, birth_date, address, postal_code, city, job, phone_number, mail.
- ASSET com os atributos: id, asset_name, acquisition_date, state, brand, model, location, asset_reference, manager, type.
- TYPE com os atributos: id, description
- REGISTER com os atributos: asset_id, alteration_date, price
- MAINTENANCE_TEAM com os atributos: team_code, location, n_elements, supervisor
- TEAM_MEMBER com os atributos: id, team_code
- INTERVENTION com os atributos: intervention_code, description, state, price, start_date, end_date, asset_id
- PERIODIC com os atributos: id, frequency
- NON_PERIODIC com os atributos: id
- SCHEDULING com os atributos: team_code, intervention_code, scheduling_date
- SKILL com os atributos: id, descriptiom
- EMPLOYEE_SKILL com os atributos: employe,skill

3.2.2 Remoção do modelo físico

Para a remoção do modelo físico foram feitas execuções DROP TABLE IF EXISTS para cada uma das tabelas citadas em [3.2.1] de modo a verificar se a tabela existe antes de excluir, caso a mesma não exista a instrução não é executada.

3.2.3 Preenchimento inicial da base de dados

Para o preenchimento inicial da base de dados foram feitas instruções de INSERT INTO, decidimos preencher a BD com alguns dados para mais tarde usarmos nos testes.

3.2.4 Mecanismo que permite inserir, remover e atualizar informação de um funcionário.

3.2.4.1 Inserção

Tendo em vista a inserção de um empregado foi criado o procedimento armazenado *dbo.insertEmployee*. Para começar foi escolhido a utilização de um procedimento armazenado e não a utilização de uma função. Visto isto, o procedimento verifica se o funcionário a ser inserido já tem registo na base de dados. Caso não tenha registo é inserido, caso contrário é lançada uma exceção a indicar que a inserção deste empregado não é possível.

3.2.4.2 Remoção

Para a funcionalidade de remoção de um funcionário foi criado um procedimento armazenado (*CREATE PROCEDURE*) *dbo.removeEmployee*. Este procedimento, primeiro verifica se pode ou não remover o funcionário. Se o funcinário pertencer a alguma equipa de manutenção, ou se for supervisor de um ativo, ou ainda se for supervisor de uma equipa de manutenção não pode ser removido. Após estas verificações, que lançam exceções caso ocorram, removemos da tabela *EMPLOYEE_SKILL* todas as competencias relativas ao funcionário a remover. Por fim removemos da tabela *EMPLOYEE*.

3.2.4.3 Atualização

Para atualizar críamos um procedimento armazenado dbo.updateEmployee que recebe todos os atributos de EMPLOYEE como parâmetro e se aquele employee já existir na BD a sua informação é atualizada.

3.2.5 Obtenção do código de uma equipa livre, dada uma descrição de intervenção, capaz de resolver o problema.

Para obtenção do código de uma equipa livre foi criada uma função que recebe como parâmetro uma descrição da intervenção que queremos associar a uma equipa e retorna o código da equipa mais adequada, caso não encontre retorna NULL. Primeiro vamos a procura dos membros mais indicados para a resolução da intervenção, ou seja, aqueles cuja competência corresponde à descrição recebida como parâmetro. Depois de já termos os membros, verificamos em que equipas estes estão inseridos. No caso de haver varias equipas escolhe-se a que teve uma intervenção atribuída à mais tempo, calculamos isto fazendo um JOIN entre as equipas com as competências requisitadas e a tabela SCHEDULING.

3.2.6 Criação do procedimento *p_criaInter* que permite criar uma intervenção

Tal como foi pedido, foi criado um procedimento armazenado dbo.p_criaInter, que permite criar uma intervenção. Para tal, este procedimento recebe como parâmetros a descrição da intervenção, o estado da intervenção, o preço da intervenção, da data de começo e fim, a frequência, o ativo correspondente a intervenção e a descrição da competência necessaria para realizar esta intervenção. Este último parâmetro é necessário para encontrar a equipa mais apropriada para a intervenção em causa.

Começamos por verificar se a data de intervenção é superior à data de aquisição do ativo intervencionado, caso contrário lança uma exceção. Depois, com auxilio à função descrita

anteriormente, obtemos o código da equipa mais apropriada para a intervenção. Caso não se encontre a equipa (função retornar NULL) criamos a intervenção e definimos o estado da intervenção como 'por atribuir'. No caso de termos encontrado uma equipa criamos a intervenção e inserimo-la tambem no SCHEDULING com a equipa correspondente. Se o parâmetro "frequência" for diferente de NULL, temos uma intervenção periodica, logo adicionamos à tabela PERIODIC, caso contrário, adicionamos à tabela NON_PERIODIC.

3.2.7 Implementação do mecanismo que permite criar uma equipa

Para criarmos uma equipa, surgiu o problema de como é que iamos passar vários elementos de uma equipa ao procedimento. A solução que propomos é o procedimento receber, como parâmetro, uma tabela auxiliar. Essa tabela auxilar é composta pelos *ids* do elementos que queremos acrescentar à equipa (neste caso os *ssn* dos *employees*), e optámos por receber num parâmetro à parte o *id* do *supervisor*. Se o número de tuplos da tabela passada não for maior que 2 lançamos uma exceção uma vez que as equipas são formadas pelo menos por 2 elementos. Com recurso a um *while* percorremos a tabela recebida e caso os *ssns* recebidos sejam válidos, vamos associar o employee a uma equipa através da relação *team_member*.

3.2.8 Actualização dos elementos de uma equipe e associação das respectivas competências

Usamos uma lógica parecida à alínea anterior para resolver este problema uma vez que íamos receber vários elementos de uma equipa. Decidímos simplificar e receber uma flag(que é uma váriavel do tipo *BIT*), *@toDelete*, que se estiver = 1, significa que é para apagar os *employees* da equipa, caso contrário é para atualizar a equipa com mais *employees* . Garantimos que a equipa tem sempre pelo menos 2 elementos, pois caso seja para eliminar veríficamos o número de elementos existentes menos o número de elementos a apagar, e esta subtração tem de ser maior que 2 elementos. Verificamos também se os *employees* que nos estão a passar são *employees* válidos, caso contrário lançamos uma exceção. Verificamos também se o *employee* a ser apagado existia antigamente na tabela. No fim de tudo atualizamos *maintenace*_team com o novo número de elementos.

3.2.9 Criação duma função para produzir a listagem das intervenções de um determinado ano

Como foi pedido, foi criado uma função de forma a produzir uma listagem das intervenções realizadas num determinado ano, para tal, esta função recebe como parâmetro o ano do qual queremos mostrar as intervenções e retorna uma tabela com o código de intervenção e a respetiva descrição. É feito um SELECT, de forma a obter os dados requisitados e colocá-los na tabela a retornar.

3.2.10 Actualização do estado de uma intervenção

Para implementar esta funcionalidade recorremos a um procedimento armazenado *dbo.updateStateIntervention.* Neste procedimento recebemos como parâmetro o código da intervenção e o estado para o qual queremos atualizar, caso a intervenção não exista lançamos ums exceção. Caso contrário fazemos UPDATE à intervenção em questão.

3.2.11 Criação duma vista que mostra o resumo das intervenções que possibilita a alteração do estado de uma ou mais intervenções

Visto que foi pedida a criação de uma vista nós começamos por criar uma vista, *dbo.vw_summary_intervetion*, com todos os atributos da tabela INTERVETION e do ASSET respetivo. Mesmo tendo em vista uma das vantagens das vistas que é a ocultação de atributos selecionados de uma tabela tendo como objetivo a privacidade e segurança dos dados, optámos por selecionar tudo referente à tabela INTERVETION e ASSET, pois achamos que todos os dados tinham a sua importância. Posto isto de forma a possibilitar a alteração do estado de uma ou mais intervenções, criamos um trigger *dbo.trg_updateInterState,* sendo um procedimento que é executado automaticamente, como resultado de um evento gerado no SGBD. Este *trigger* dispara quando é executado o UPDATE sobre a vista *dbo.vw_summary_interventation*, utilizamos um trigger *instead of update*, ou seja, é executado em de vez do update. Dentro do trigger executamos o procedimento armazenado *dbo.updateStateIntervention*, que altera o estado do procedimento.

4. Avaliação experimental

De forma a garantir que todas as funcionalidades realizadas se encontram corretas e coerentes com o funcionamento do sistema de informação realizaram-se testes.

4.1 Testes às funcionalidades implementadas

Realizamos testes particulares ao sistema, através da execução de todos os procedimentos armazenados realizados e verificação de conteúdos sobre a vista criada com recurso aos dados posteriormente inseridos no modelo físico.

4.2 Análise de resultados

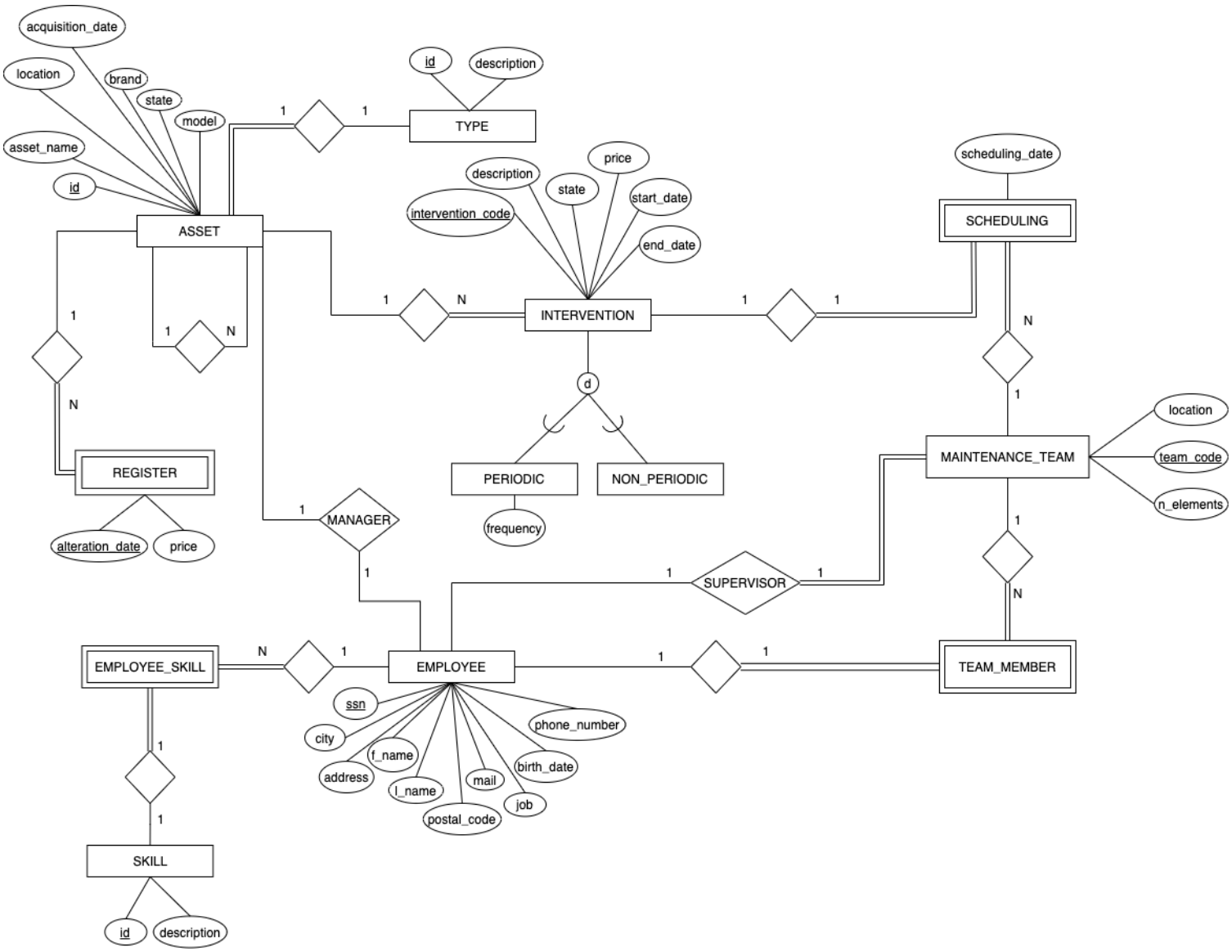
Fez-se uma avaliação positiva das funcionalidades implementadas. Observou-se os resultados de cada execução de procedimentos armazenados nas tabelas do sistema e o estado destas correspondia ao pretendido.

5. Referências

[1] Fundamentals of Database Systems (7th ed.), Ramez Elmasri, Shamkant B. Navathe Pearson Education, 2017

[2] Sistemas de Informação II (ISEL-ADEETC Walter Vieira)

A. Diagrama do modelo Entidade-Associação



B. Modelo Entidade-Relacionamento

EMPLOYEE(ssn, f_name, l_name, birth_date, address, postal_code, city, job, phone_number, mail)

Atributo	Tipo	Restrições de Integridade
ssn	*INT*	
f_name	*VARCHAR(50)*	*NOT NULL*
l_name	*VARCHAR(50)*	*NOT NULL*
birth_date	*DATE*	*NOT NULL*
address	*VARCHAR(50)*	*NOT NULL*
postal_code	*VARCHAR(50)*	*NOT NULL*
city	*VARCHAR(50)*	*NOT NULL*
job	*VARCHAR(50)*	*NOT NULL*
phone_number	*INT*	*NOT NULL*
mail	*VARCHAR(100)*	

Tabela 1 employee

PK{ssn}

TYPE(id, description)

Atributo	Tipo	Restrições de Integridade
id	*INT IDENTITY(1,1)*	
description	*VARCHAR(200)*	

Tabela 2 type

PK{id}

ASSET(id, asset_name, acquisition_date, state, brand, model, location, asset_reference, manager, type)

Atributo	Tipo	Restrições de Integridade
id	*INT IDENTITY(1,1)*	
asset_name	*VARCHAR(30)*	*NOT NULL*
acquisition_date	*DATE*	*NOT NULL*
state	*BIT*	*NOT NULL* Valores possiveis: {“0”(desativado), “1”(operacional)}
brand	*VARCHAR(30)*	
model	*VARCHAR(30)*	
location	*VARCHAR(30)*	*NOT NULL*
asset_reference	*INT*	Se for pai => *NULL*
manager	*INT*	
type	*INT*	

Tabela 3 asset

PK{id}

FKs: { asset_reference} referencia ASSET{id}, {manager} referencia EMPLOYEE{ssn}, {type} referencia TYPE{id}

Restrições de Integridade:

- O tipo do ativo de topo da hierarquia tem de ser igual ao(s) tipo(s) do(s) activo(s) “filho(s)”

REGISTER(asset_id, alteration_date, price)

Atributo	Tipo	Restrições de Integridade
asset_id	*INT*	
alteration_date	*DATE*	
price	*DECIMAL(6,2)*	

Tabela 4 register

PK:{alteration_date, asset_id}

FK: {asset_id} referencia ASSET{id}

Restrições de Integridade:

- Pretende-se manter o registo historico do valor comercial do ativo, em euros, registando-se a data (no formato dd-mm-aaaa) em que a alteração ocorreu.

MAINTENANCE_TEAM (team_code, location, n_elements, supervisor)

Atributo	Tipo	Restrições de Integridade
team_code	*INT IDENTITY(1,1)*	
location	*VARCHAR(50)*	*NOT NULL*
n_elements	*INT*	>= 2 em que um deles é o supervisor
supervisor	*INT*	

Tabela 5 maintenance_team

PK:{team_code}

FK: {supervisor} referencia EMPLOYEE{ssn}

Restrições de Integridade:

- A equipa completa tem pelo menos 2 elementos durante a intervenção ao ativo. Se for necessario, podem-se adicionar/alterar elementos a equipa, durante a intervenção.
- Cada equipa so consegue realizar uma intervenção de cada vez, podendo ter até 3 intervenções atribuidas.
- Se não for possivel atribuir uma equipa, o estado da intervenção deve ser “por atribuir”.
- Note-se que a equipe pode mudar ao longo da resolução da intervenção, devendo ficar registadas todas as equipas envolvidas, garantindo que é possivel ordena-las de forma cronologica.

TEAM_MEMBER (id, team_code)

Atributo	Tipo	Restrições de Integridade
id	*INT*	
team_code	*INT*	

Tabela 6 team_member

PK{id, team_code}

FKs: {id} referencia EMPLOYEE{ssn}, {team_code} referencia MAINTENANCE_TEAM (team_code)

Restrições de Integridade:

- Existe uma pessoa da empresa que gere o ativo e que não pode participar na equipa que faz a intervenção desse ativo. Cada elemento da equipa tem registado as suas competências.

INTERVENTION (intervention_code, description, state, price, start_date, end_date, asset_id)

Atributo	Tipo	Restrições de Integridade
intervention_code	*INT IDENTITY(1,1)*	
description	*VARCHAR(50)*	Valores possiveis{“avaria”, “rutura”, “inspeção”}
state	*VARCHAR(50)*	Valores possiveis{“por atribuir”, “em analise”, “em execução”, “concluido”}
price	*DECIMAL(10,2)*	Em euros
start_date	*DATE*	*NOT NULL*
end_date	*DATE*	*NOT NULL*
asset_id	*INT*	

Tabela 7 intervention

PK{id}

FK: {asset_id} referencia ASSET{id}

Restrições de Integridade:

- A data de intervenção deve ser superior a data de aquisição do ativo intervencionado.

PERIODIC (id, frequency)

Atributo	Tipo	Restrições de Integridade
id	*INT*	
frequency	*INT*	Em meses (valores entre 1 e 12)

Tabela 8 periodic

PK{id}

FK{id} referencia INTERVENTION{ intervention_code }

NON_PERIODIC(id)

Atributo	Tipo	Restrições de Integridade
id	*INT*	

Tabela 9 non_periodic

PK{id}

FK{id} referencia INTERVENTION{ intervention_code }

SCHEDULING (team_code, intervention_code, scheduling_date)

Atributo	Tipo	Restrições de Integridade
team_code	*INT*	
intervention_code	*INT*	
scheduling_date	*DATE*	

Tabela 10 scheculing

PK{team_code, intervention_code}

FKs: {team_code} referencia MAINTENANCE_TEAM{team_code}, {intervention_code} referencia INTERVENTION{intervention_code}

Restrições de Integridade:

- Se não for possível attribuir uma equipa, o estado da intervenção deve ser “por atribuir”

SKILL(id, descriptiom**)**

Atributo	Tipo	Restrições de Integridade
id	*INT*	
description	*VARCHAR(100)*	

Tabela 11 skill

PK {id}

EMPLOYEE_SKILL(employe,skill)

Atributo	Tipo	Restrições de Integridade
employee	*INT*	
skill	*INT*	

Tabela 12 employee_skill

PK{employee_code, skill}

FK: {employee_code} referencia EMPLOYEE{ssn}, {skill} referencia SKILL{id}

Restrições de Integridade:

- Uma pessoa so pode pertencer a uma equipe. A atribuição de uma intervenção a uma equipa so é possível se a descrição for compativel com as competencias da equipa.