INSTITUTO SUPERIOR
DE ENGENHARIA DE LISBOA

Licenciatura Informática e de Computadores

Sistemas de Informação II

Relatório da 1ª Fase do Trabalho

Semestre de Inverno

2016/2017

Realizado por:
Tiago Santos nº40590
João Correia nº40604
João Lopes nº39120

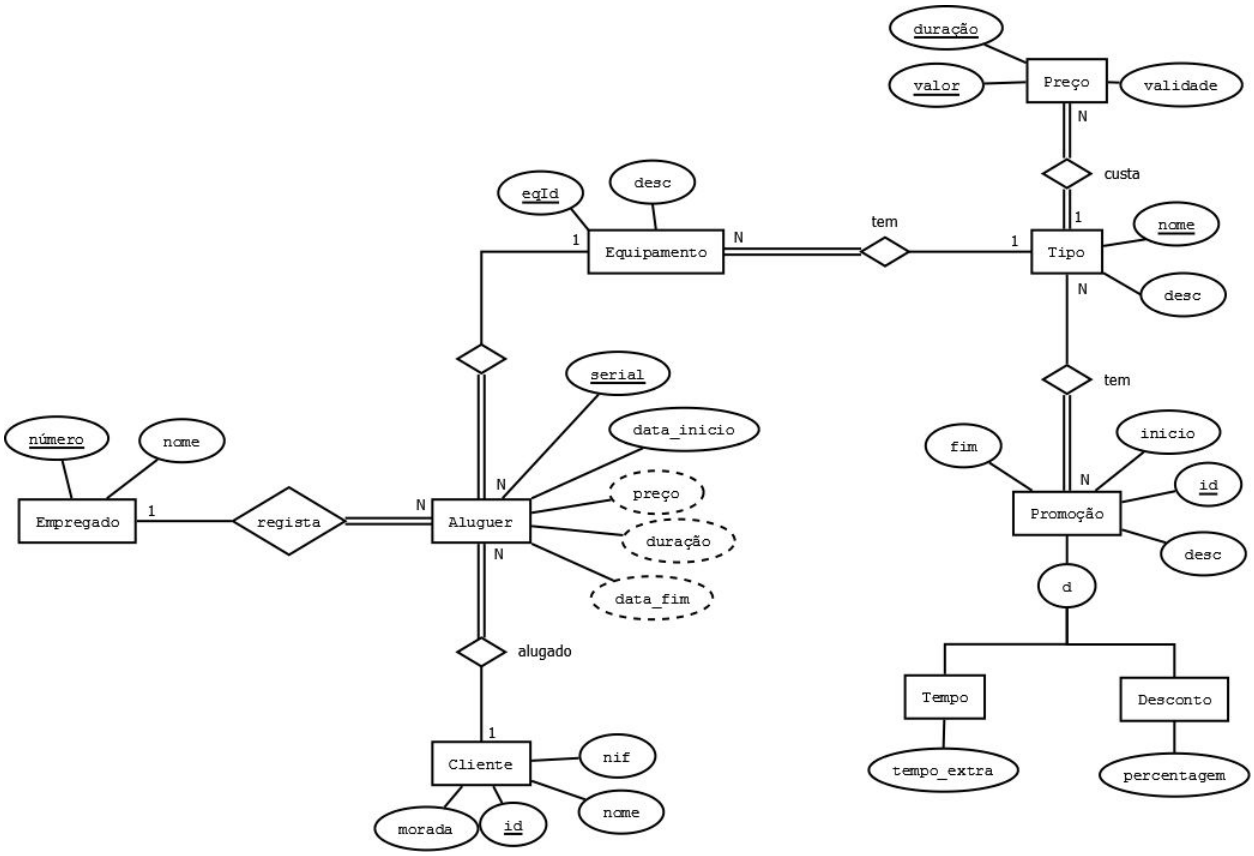
INTRODUÇÃO

Para este trabalho foi-nos pedido para desenvolver um sistema informático para melhorar a gestão dos alugueres de equipamentos de praia para a empresa denominada Ænima.

Com base no que aprendemos nas aulas e conhecimentos de SI1 começámos por criar um Modelo EA que fosse consistente com as entidades descritas no enunciado, depois de termos o modelo de dados (conceptual e relacional), incluindo as restrições de integridade começámos a criar o código T-SQL que permitisse implementar as funcionalidades pedidas no enunciado.

Certificamo-nos também que o modelo de dados estava normalizado até à 3FN porque todos os valores das colunas das tabelas eram atómicos (1FN), cada atributo não chave dependía de toda a chave primária e não apenas parte dela (2FN) e que cada atributo não-chave não possui dependências transitivas.

Modelo EA


duração
valor
Preço
validade
N
custa
1
eqId
desc
1
Equipamento
N
tem
1
Tipo
name
desc
N
serial
tem
número
name
data_inicio
fim
inicio
N
N
preço
id
Empregado
1
regista
N
Aluguer
Promoção
duração
desc
N
d
data_fim
alugado
Tempo
Desconto
1
Cliente
nif
tempo_extra
percentagem
morada
id
name

Modelo Relacional

Tipo(nome, descTipo):
- CC = {{nome}}

RI 1: Cada tipo tem de ter pelo menos um Preço

Equipamento(id, eqDesc, tipo):
- CC = {{id}}
- CE = {{tipo REF Tipo(nome)}}

RI 2: Tipo é obrigatório;

Cliente(id, nome, NIF, morada):
- CP = {{id}}

RI 3: Nome, NIF, morada são obrigatórios, caso não seja "cliente final"
RI 4: Cliente final é o único que tem nome, nif e morada a nulo
RI 5: NIF tem 6 dígitos e é maior que 0

Empregado(número, nome):
- CP = {{número}}

RI 6: número é maior que 0

Preço(valor, duração, validade, tipo)
- CE = {{tipo REF Tipo(nome)}}

RI 7: duração > 0
RI 8: valor >= 0
RI 9: Cada Preço tem de ter um tipo

Aluguer(serial, eqId, cliente, empregado, data_inicio)
- CC = {{serial}}
- CE = {{eqId REF Equipamento(eqId)}}
- CE = {{cliente REF Cliente(id)}}
- CE = {{empregado REF Empregado(id)}}

RI 10: eqId, cliente, empregado, data_inico são obrigatórios

AluguerPreçoDuração(serial, preço, duração)

- CC = {{serial}}
- CE = {{serial REF Aluguer(serial)}}

AluguerFim(serial, data_fim)

- CC = {{serial}}
- CE = {{serial REF Aluguer(serial)}}

RI 11: Um aluguer só pode ser inserido em AluguerFim depois de ter sido inserido em AluguerPreçoDuração

Tipo_Promoção(idPromoção, tipo)

- CC = {{idPromoção, eqId}}
- CE = {{tipo REF Tipo(nome)}}
- CE = {{idPromoção REF Promoção(id)}}

Promoção(id, inicio, fim, desc):

- CC = {{id}}

RI 12: fim > inicio
RI 13: início e fim são obrigatórios

PromoçãoTemporal(id, tempoExtra):

- CC = {{id}}
- CE = {{id REF Promoção(id)}}

RI 13: Os ids em PromoçãoTemporal não aparecem em PromoçãoDesconto
RI 14: tempoExtra é obrigatório
RI 15: tempoExtra > 0

PromoçãoDesconto(id, desconto):

- CC = {{id}}
- CE = {{id REF Promoção(id)}}

RI 16: Os ids em PromoçãoDesconto não aparecem em PromoçãoTemporal
RI 17: Desconto é obrigatório
RI 18: Desconto > 0

A maior parte do código que dá suporte às funcionalidades requeridas no enunciado está contida em procedimentos armazenados. Optou-se por esta abordagem para evitar que qualquer utilizador do sistema de informação tivesse de se preocupar com a manutenção do estado de várias tabelas quando efetuasse qualquer alteração. Além deste benefício esta abordagem permite restringir acesso arbitrário a tabelas, através de permissões, mantendo as funcionalidades, através dos procedimentos, mas impedindo quaisquer alterações que deixariam o sistema num estado inconsistente.

De certa forma o conjunto de procedimentos/funções que manipulam as tabelas agem como interface do sistema.

Criar o modelo físico e remover o modelo físico

O ficheiro CRIAR_TABELAS.sql contém o código necessário para criar e remover o modelo físico da base de dados mas apenas contém a definição e criação de tabelas, não de outro tipo de objetos.
O código que cria os procedimentos armazenados e funções está alojado noutros ficheiros em separado para facilitar a organização da solução.

Inserir informação de um cliente

A inserção de informação de um cliente está implementada no procedimento armazenado InserirCliente alojado no ficheiro PROCEDURE_INSERIR_CLIENTE.sql

A primeira instrução deste procedimento executa uma leitura para verificar se o NIF do cliente a inserir é único. Isto é algo peculiar porque existe uma keyword(UNIQUE) que garante que uma coluna é única numa dada tabela. A razão pela qual não se recorre ao UNIQUE deriva do facto de um Aluguer não poder ser permanentemente apagado.

Imagine-se o cenário:

É terminado um aluguer e este é apagado de forma não permanente(mais detalhes em baixo) e este Aluguer contínua na tabela marcado como apagado. Mais tarde é apagado o cliente que pagou esse aluguer. Neste momento se um auditor tencionar rever os alugueres feitos irá encontrar entradas na tabela Aluguer que referenciam clientes inexistentes.

Por este motivo é necessário marcar também o cliente como apagado mas mantê-lo na tabela.

Suponha-se agora que esse mesmo cliente tenciona efetuar um novo aluguer. Ao inserir esse cliente nesta tabela ir-se-ia violar a restrição UNIQUE.

É por este facto que a coluna NIF não está marcada como UNIQUE e a identificação de um cliente é feita por um identificador auto-gerado.

A inserção do cliente é feita num âmbito transacional com nível de isolamento READ COMMITTED para garantir que aquando da leitura à tabela Cliente esta está estável.

Actualizar informação de um cliente

Localizada no ficheiro PROCEDURE_ATUALIZAR_CLIENTE.sql, no procedimento AtualizarCliente.
Não é necessário nenhum âmbito transacional pois a atualização consiste apenas numa única instrução UPDATE.

Remover um cliente

Ficheiro PROCEDURE_REMOVER_CLIENTE.sql, procedimento RemoverCliente.

Para esta instrução é usada uma transacção com nível de isolamento READ UNCOMMITTED pois apenas é executado um UPDATE para afetar a coluna ‘valido’ da tabela Cliente com valor 0. Executa-se de seguida o procedimento RemoverAlugueresParaCliente para marcar como removidos quaisquer alugueres efetuados pelo cliente em questão.

A razão pela qual não é feito um DELETE está explicada em detalhe mais em cima neste documento.

Inserir informação de um equipamento

Funcionalidade implementada nos procedimentos do ficheiro PROCEDURE_INSERIR_EQUIPAMENTO.sql. São disponibilizados dois procedimentos, um para inserir um equipamento com um tipo já existente e outro sem tipo existente. O segundo é meramente de caráter utilitário.

O primeiro procedimento insere um equipamento numa transacção com nível de isolamento READ COMMITTED para garantir que a verificação da existência do tipo(SELECT sobre a tabela Tipo) é consistente e caso não exista é inserida então uma nova linha em Tipo De de seguida é inserido o equipamento.

Actualizar informação de um equipamento

Esta funcionalidade é suportada pelos procedimentos contidos no ficheiro PROCEDURE_ATUALIZAR_EQUIPAMENTO.sql.

Para esta funcionalidade existem dois procedimentos:

ActualizarEquipamento que simplesmente actualiza a descrição do equipamento e o seu tipo, caso o tipo não exista lança excepção, e ActualizarEquipamentoComNovoTipo.

Este procedimento tem uma transacção com nível de isolamento READ COMMITTED para verificar se o tipo para o qual se tenciona atualizar existe. Em caso de inexistênica este é criado seguido da actualização da informação do equipamento.

Remover um equipamento

Implementação contida no procedimento RemoverEquipamento contido no ficheiro PROCEDURE_REMOVER_EQUIPAMENTO.sql.

Nesta instrução usa-se uma transacção com nível de isolamento READ COMMITTED para garantir que são lidos dados consistentes, nomeadamente na verificação do uso do equipamento a remover.

Caso se verifique que este equipamento está, de facto, em uso é lançada uma excepção a indicar tal situação e no caso contrário é atualizada a coluna valid para 0 marcando assim o equipamento como removido.

A razão pela qual um equipamento não é removido por completo é a mesma pela qual um cliente não é removido por completo, para garantir visibilidade depois da remoção de um aluguer que utilize este equipamento.

Inserir informação de uma promoção

Funcionalidade implementada pelos procedimentos do ficheiro PROCEDURE_INSERIR_PROMOÇÃO.sql

Para inserir uma promoção usa-se 1 de 2 procedimentos InserirPromocaoTemporal ou InserirPromocaoDesconto. Ambos correm numa transacção com nível de isolamento SERIALIZABLE pois é desejável que as todas as tabelas em questão sejam acedidas em exclusão. Ambos têm um Try...Catch e começam por tentar inserir a promoção executando o procedimento InserirPromocao e depois tentam inserir a promoção na tabela respetiva (PromocaoTemporal ou PromocaoDesconto).

O procedimento InserirPromocao também tem uma transacção com nível de isolamento SERIALIZABLE e trata de inserir uma nova linha na tabela Promoção e inserir o tipo de promoção em TipoPromocao.

Actualizar informação de uma promoção

Funcionalidade implementada pelos procedimentos do ficheiro PROCEDURE_ATUALIZAR_PROMOÇÃO.sql

Para actualizar uma promoção também se usa um dos procedimentos ActualizarPromocaoTemporal e ActualizarPromocaoDesconto. Ambos usam uma transacção com nível de isolamento READ COMMITTED e começam por tentar actualizar a promoção chamando ActualizarPromocao. Na ocorrência de uma excepção a transação é abortada.

Em caso de sucesso tabela respetiva é atualizada (PromocaoTemporal ou PromocaoDesconto) afetando a coluna tempoExtra ou percentagemDesconto.

O procedimento ActualizarPromocao usa uma transacção com nível de isolamento READ COMMITTED e lê da tabela Promocao a promoção que de deseja actualizar mudando os valores das colunas conforme foram fornecidos ou não(null).

Remover uma promoção

Funcionalidade suportada pelos procedimentos contidos no ficheiro PROCEDURE_REMOVER_PROMOÇÃO.sql.

Nesta instrução é usado um dos procedimentos RemoverPromocaoTemporal e RemoverPromocaoDesconto.

Ambos usam uma transacção com nível de isolamento READ UNCOMMITTED pois não é feita nenhuma leitura. Ambos têm um TryCatch e começam por remover a promoção da respectiva tabela PromocaoTemporal ou PromocaoDesconto e depois executam o procedimento RemoverPromocao. Este procedimento usa uma transacção com nível de isolamento READ UNCOMMITTED porque trata apenas de remover uma linha da tabela TipoPromocao e Promocao.

Inserir um aluguer com um cliente existente

Funcionalidade contida no procedimento InserirAluguer contido no ficheiro PROCEDURE_INSERIR_ALUGUER.sql.

O procedimento InserirAluguer recebe os seguintes parâmetros: o número do empregado, o identificador único do cliente, o identificador único do equipamento, a data e hora de início de aluguer, a duração do aluguer, o preço pago e a promoção que este aluguer pode ter, sendo este último parâmetro opcional.

Para verificar se os dados estão correctos, vai ser feita uma leitura a cada tabela correspondente aos identificadores passados como parâmetros (empregado, cliente, equipamento, promoção) excepto os parâmetros preço e duração que são verificados na mesma tabela (Preco). Irá também comparar nas respectivas leituras anteriores se a data do aluguer está dentro da validade da promoção e da validade do preço. Será feito um rollback e lançado uma exceção se não cumprir os requisitos anteriores.

Se aluguer tiver uma promoção vai-se utilizar a função CalcularDuracaoPreco que calcula o preço e duração finais se este tiver uma Promoção seja ela temporal/desconto.

Por fim adiciona às tabelas Aluguer, AluguerPrecoDuracao e AluguerDataFim um novo tuplo. No AluguerDataFim irá ser ser passado ao atributo data_fim a adição entre Data de Início inserida na tabela Aluguer e a duração inserida na tabela AluguerPrecoDuracao. Este procedimento usa uma transacção com nível de isolamento REPEATED READ para proteger

os valores que estamos a ler de alterações, pois as escritas na base de dados irão depender desses valores.

Inserir um aluguer com um novo cliente

Funcionalidade contida no procedimento InserirAluguerComNovoCliente contido no ficheiro PROCEDURE_INSERIR_ALUGUER.sql.

O procedimento InserirAluguerComNovoCliente irá utilizar os procedimentos: primeiro executa o procedimento InserirCliente e de seguida o InserirAluguer com o novo cliente. Este procedimento recebe como parâmetros, os mesmos parâmetros que estão nos procedimentos que irá utilizar. Como o InserirAluguer usar o nível de isolamento REPEATABLE READ, este irá também usar o mesmo nível de isolamento.

Remover um aluguer

Funcionalidade contida no procedimento RemoverAluguer contido no ficheiro PROCEDURE_REMOVER_ALUGUER.sql.

Este procedimento irá fazer um update para um determinado Aluguer, colocando o atributo flag: deleted a 1.

Remover os alugueres de um cliente

Funcionalidade contida no procedimento RemoverAlugueresParaCliente contido no ficheiro PROCEDURE_REMOVER_ALUGUER.sql.

Este procedimento irá fazer um update para todos os tuplos Aluguer de um determinado, colocando o atributo flag: deleted a 1, assinalando-os como apagados.

Inserir um Preço

Funcionalidade contida no procedimento InserirPreco contido no ficheiro PROCEDURE_INSERIR_PRECO.sql.

Este procedimento começa por verificar se o tipo passado como parâmetro existe na tabela Tipo, se existir irá inserir na tabela Preco. O procedimento InserirPreco usa uma transacção com nível de isolamento READ COMMITTED e faz uma leitura da tabela Tipo e uma escrita na tabela Preco dependendo da primeira leitura.

Actualizar um Preço

Funcionalidade contida no procedimento ActualizarPreco localizado no ficheiro PROCEDURE_ATUALIZAR_PRECO.sql.

Este procedimento irá fazer duas leituras iniciais à View TipoView e à tabela Preco para conferir se o tipo e o preço indicado como parâmetro existe. Caso ambos existirem nas tabelas vai-se remover o preço antigo e inserir o novo preço. O nível de isolamento vai ser Repeatable Read.

Remover um Preço

Funcionalidade contida no procedimento RemoverPreco alojado no ficheiro PROCEDURE_REMOVER_PRECO.sql.

Este procedimento só tem uma instrução de DELETE, ou seja, não irá ter transações.

Funções

Foram utilizadas as funções para os seguintes pontos:

- Listar todos os equipamentos livres, para um determinado tempo e tipo;
- Listar os equipamentos sem alugueres na última semana;

Ambos os pontos acima retornam uma tabela, algo que não seria possível com procedimentos armazenados, também se utilizou funções pois nenhum dos pontos chega a escrever na base de dados, faz apenas leituras para obter cada tabela.

Listar todos os equipamentos livres, para um determinado tempo e tipo

Foi criada a função:

EquipamentosLivres(@inicio, @fim, @tipo = NULL)

em que os parâmetros @início e @fim correspondem a uma datetime e o parâmetro @tipo corresponde a um tipo valido que está na tabela Tipo. Se o tipo for passado a nulo, a função irá considerar todos os tipos. Esta função irá fazer duas leituras, uma para ver todos os alugueres que estão entre as datas @início e @fim, e irá apresentar os alugueres que estejam a ocorrer entre essas duas, depois faz uma leitura de todos os equipamentos excepto aqueles que apareceram com alugueres na primeira leitura para o determinado @tipo.

Listar os equipamentos sem alugueres na última semana

Foi criada a função:

EquipamentosSemAlugueresNaUltimaSemana()

que não irá receber parâmetros e utilizando a função EquipamentosLivres irá apresentar uma tabela com todos os equipamentos entre hoje e sete dias antes passando a variavel @tipo a nulo.

Função Auxiliar

A função auxiliar:

CalcularDuracaoPreco(@pid, @eqId, @preco, @duracao)

retorna um tuplo com os atributos duração preço finais, calculados a partir de uma promoção (@pid), um preço (@preco) e uma duração (@duracao).

Vistas

Foram criadas 4 vistas:

- ClienteView: que irá apresentar os clientes válidos, ou seja, os clientes que tem bit do atributo ‘valido’ a 1
- AluguerView: que irá apresentar os alugueres válidos, ou seja, os alugueres com o bit deleted a 0
- EquipamentoDisponivelView: que irá apresentar os equipamentos com o bit ‘valid’ a 1
- TipoView: que irá apresentar todos os Tipos que têm equipamentos e sejam válidos.