Gestão de um Hotel

Projeto de BDAD, 3ª entrega

Grupo 107

Gonçalo Vasconcelos Cunha Miranda Moreno up201503871@fe.up.pt

José Manuel Faria Azevedo up201506448@fe.up.pt

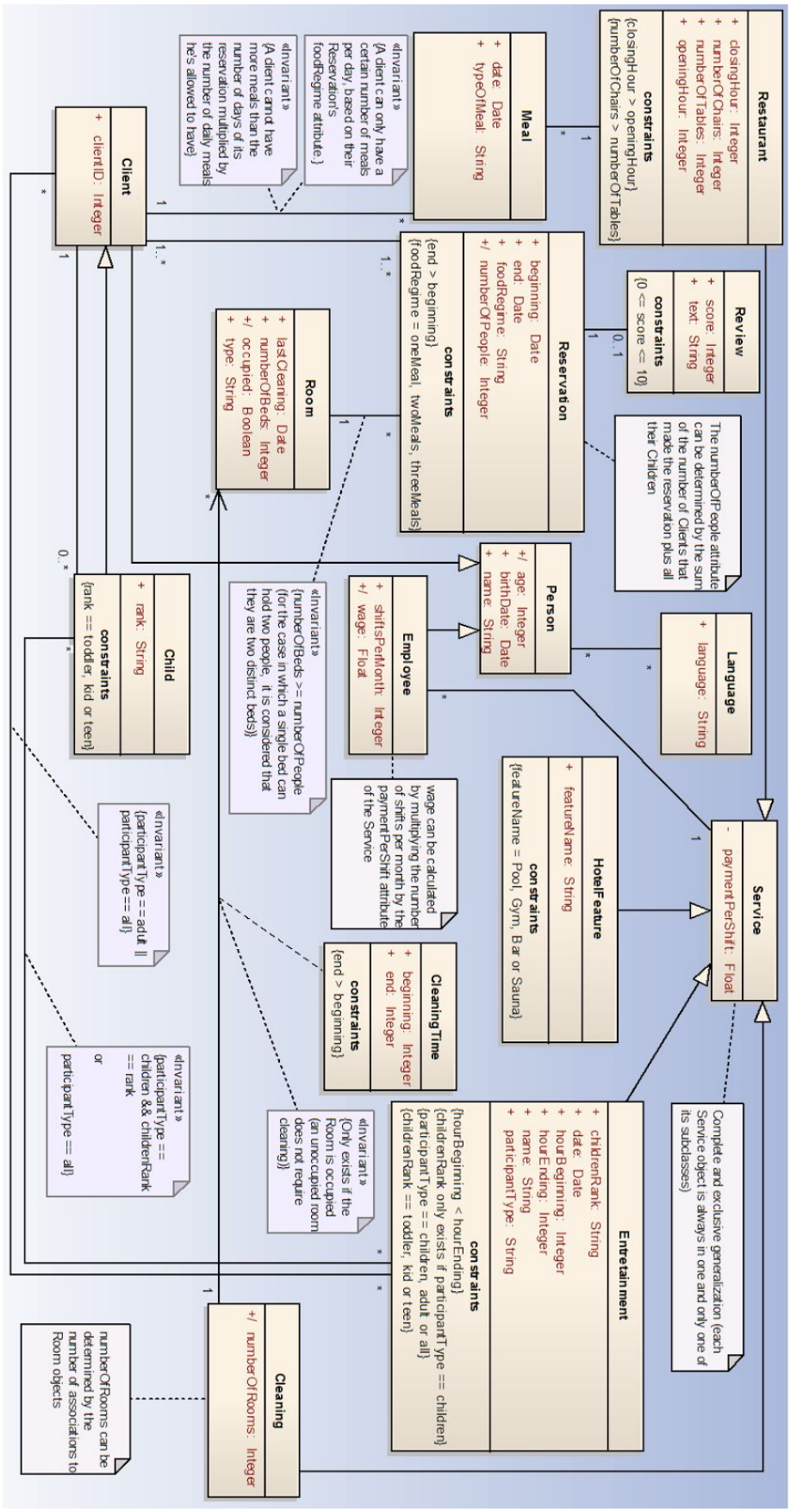
Tiago Lascasas dos Santos up201503616@fe.up.pt

Contexto do Tema do Projeto

Para este projeto idealizou-se um sistema para a gestão de um hotel, que se encarrega dos clientes, das suas reservas, dos funcionários do hotel e de todos os serviços e tarefas do hotel que envolvam ambas as entidades. Procedeu-se à seguinte verbalização das várias interações entre as entidades e especificidades do sistema:

- Um cliente tem um nome, um identificador, uma data de nascimento, idade e pode falar várias línguas.
- Um cliente pode ser ainda uma criança, sendo que esta tem uma classificação (bebé, infante ou pré-adolescente) e um adulto (cliente) responsável.
- Um ou mais clientes poderão realizar uma reserva.
- Cada reserva tem um e um só quarto associado, assim como um de três regimes alimentares.
- Cada quarto tem de ter um número mínimo de camas para todas as pessoas da reserva. O número de pessoas da reserva pode ser determinado pela soma do número de clientes associados à reserva mais as suas crianças.
- Um cliente poderá usufruir do restaurante do hotel consoante o seu regime alimentar (existe um regime de uma, duas ou três refeições diárias).
- Um quarto tem ainda um horário de limpeza associado, assim como a informação de estar ocupado ou livre para reservar. Só é limpo quando está ocupado.
- Um funcionário pode trabalhar no restaurante, no entretenimento, nos serviços do hotel (piscina, ginásio, bar e sauna) ou nos serviços de limpeza.
- Cada funcionário não pode trabalhar em mais do que um sítio e pode falar também várias línguas.
- Cada serviço do hotel tem um pagamento por turno (dependente do serviço), e cada funcionário tem informação sobre quantos turnos realiza por mês.
- As atividades de lazer são feitas ou para adultos, ou para crianças ou para ambos, sendo que necessitam de uma inscrição prévia. Ainda no caso das crianças, depende também do tipo de criança.
- Cada reserva poderá ser sujeita a uma análise por parte dos seus clientes, sendo que esta inclui um texto e uma nota de 0 a 10.

Procedeu-se, de seguida, à definição de um modelo conceptual para o sistema em UML, o qual composto por 15 relações e 11 associações:



Nota: o modelo acima foi revisto, tendo sido mudada a multiplicidade da relação Room-Reservation de 1 para *

Definição do Esquema Relacional

Procedeu-se à seguinte conversão do modelo conceptual para o modelo relacional, sendo que os atributos sublinhados fazem parte da chave primária da relação e os atributos a itálico fazem parte de chaves estrangeiras:

Relações sem derivação:

Language (idLanguage, language)

Reservation (idReservation, beginning, ending, foodRegime, *idRoom* → ***Room***)

Room (idRoom, lastCleaning, numberOfBeds)

Meal (idMeal, date, typeOfMeal, *idService -> Restaurant, idPerson* → ***Client***)

Review (idReview, score, text, *idReservation* → ***Reservation***)

CleaningTime (*idRoom* → ***Room***, *idService* → ***Cleaning***, beginning, ending)

Service e relações derivadas:

Service (idService, paymentPerShift)

HotelFeature (*idService* → ***Service***, featureName)

Entertainment (i*dService* → ***Service***, childrenRank, date, hourBeginning, hourEnding, name, participantType)

Restaurant (*idService* → ***Service***, closingHour, numberOfChairs, numberOfTables, openingHour)

Cleaning (*idService* → ***Service***)

Person e relações derivadas:

Person (idPerson, birthDate, name)

Employee (*idPerson* → ***Person***, shiftsPerMonth, *idService* → ***Service***)

Client (*idPerson* → ***Client***, isChild, childRank, *idPerson* → ***Client***)

Relações adicionais para associações muitos-para-muitos:

ClientReservation (*idPerson* → ***Client***, *idReservation* → ***Reservation***)

LanguagePerson (*idPerson* → ***Person***, *idLanguage* → ***Language***)

ClientEntertainment (*idPerson* → ***Client***, *idService* → ***Entertainment***)

A derivações das classes Service, Person e Client foram realizadas de acordo com as seguintes considerações:

- De **Service**, derivam Restaurant, Cleaning, Entertainment e HotelFeature. Para converter estas classes em relações adotámos um modelo "Entity-Relation", porque cada uma das classes derivadas continha o único atributo de Service (paymentPerShift) e, na maior parte dos casos, eram acrescentados muitos mais. Dado que se trata de uma generalização disjunta e completa, poderia ter sido usado uma abordagem "Object-Oriented" (adicionando o atributo paymentPerShift a cada uma das classes derivadas e eliminado a classe Service), mas de modo a tornar genérica a chave estrangeira que Employee tem para um Service optou-se pelo modelo anterior.
- De **Person**, derivam Client e Employee. Neste caso também usamos uma abordagem "Entity-Relation" pelas mesmas razões de Service. Também poderia ter sido usado um modelo "Object-Oriented", visto também ser uma generalização completa e disjunta mas, mais uma vez, de modo a tornar genérica a associação entre Language e Person, optou-se por mapear na mesma a classe Person para o relacional, tal como foi feito com Service.
- De **Client**, deriva Child. Neste caso adotámos uma abordagem com base em "nulls", pois só havia uma classe derivada (Child) com apenas um atributo a mais (rank) quando comparada com Client.

Análise das Dependências Funcionais

As relações acima definidas têm as seguintes dependências funcionais:

Language: idLanguage → language
Reservation: idReservation → beginning, end, foodRegime
Room: idRoom → lastCleaning, numberOfBeds, idReservation
Meal: idMeal → date, typeOfMeal, idService, idPerson
Review: idReview → score, text, idReservation
CleaningTime: idRoom, idService → beginning, end
Service: idService → paymentPerShift
HotelFeature: idService → featureName
Entertainment: idService → childrenRank, date, hourBeginning, hourEnding, name, participantType
Restaurant: idService → closingHour, numberOfChairs, numberOfTables, openingHour
Cleaning: não tem dependências funcionais
Person: idPerson → birthDate, name
Employee: idPerson → shiftsPerMonth, idService
Client: idPerson → clientID, isChild, childRank, idPerson
ClientReservation: não tem dependências funcionais
LanguagePerson: não tem dependências funcionais
ClientEntertainment: não tem dependências funcionais

Após uma análise de todas as relações concluímos que estão todas na Boyce–Codd Normal Form (BCNF), visto que nenhuma relação contém uma dependência não trivial. Não conseguimos a partir de um campo ou atributo determinar outro atributo (sem contar com a chave primária, que gera apenas dependências triviais e é uma super chave da relação). Este resultado é esperado, pois são raros os casos em que a passagem de modelos UML conceptuais para modelos relacionais produzem relações que violam a BCNF. Como a BCNF é uma forma mais restrita da 3ª Forma Normal, e como todas as dependências estão na BCNF, podemos também concluir que todas as dependências estão na 3ª Forma Normal.

Especificação das Restrições

Na conversão do modelo relacional para uma base de dados em SQLite foram consideradas e implementadas várias restrições. É de notar que todos os atributos foram definidos como `not null`*, exceto em casos explicitamente especificados:

Language:

- `Unique` em idLang, visto que é a `primary key` da relação.
- O atributo lang (que contém o nome da língua) está definido como `unique`, visto que não é possível adicionar a mesma língua duas vezes.

Reservation:

- `Unique` em idReservation, visto que é a `primary key` da relação.
- Tem a `foreign key` room_fkey que referencia o quarto a qual foi reservado.
- A hora de início tem que ser antes da hora do fim, por razões lógicas (restrição `check` dateCheck).
- O tipo de reserva só pode ser de 3 tipos (“single”, “double” ou “triple”), implementado pela restrição `check` regimeCheck.

Room:

- `Unique` em idRoom, visto que é a `primary key` da relação.
- No hotel, em cada quarto só existem no máximo 4 camas, portanto o atributo numberOfBeds tem que estar entre 1 e 4 (restrição `check` nBeds).

Meal:

- `Unique` em idMeal, visto que é a `primary key` da relação.
- Tem duas `foreign keys`, res_fkey e cli_fkey, que indicam o restaurante de onde foram feitas (idService) e qual o Client a ela associada (idPerson).
- Tal como o regime de quarto o tipo de refeição tem ser “single”, “double” ou “triple” (restrição `check` mealType).

Review:

- `Unique` em idReview, visto que é a `primary key` da relação.
- Tem uma `foreign key`, resv_fkey, que referencia a reserva de onde provém a review.
- O score da review (atributo score) só pode ser entre 1 e 5 (restrição `check` scoreLimit).

CleaningTime:

- `Unique` em idRoom, visto que é a `primary key` da relação.
- A hora de início tem de ser anterior à hora de fim (restrição `check` hourCheck)
- A hora de início e fim tem de ser única para um único serviço (restrição `check` uniqueHour).
- Tem duas `foreign keys`, room_fkey e cl_fkey, que apontam para o quarto que vai ser limpo e outra para o serviço de limpeza (Cleaning), respectivamente.

Service:

- `Unique` em idService, visto que é a `primary key` da relação.
- Cada serviço tem um pagamento associado que tem que ser maior que 0, logo paymentPerShift > 0 (restrição `check` validPay).

Todas as relações derivadas de Service têm uma `foreign key`, ser_fkey, que aponta para um Service, sendo que é também `primary key` da relação e, portanto, também `unique`. Deste modo, esta restrição será omitida nas classes abaixo, por motivos de simplificação de texto.

HotelFeature:

- No hotel só existem 4 features, “bar”, “pool”, “gym” e “sauna”, logo featureName tem que ser de um destes 4 (restrição `check` featureType).

Entertainment:

- Existem duas restrições adicionais que indicam se o espetáculo é para “children”, “adult” ou “all”: a restrição `check` pType verifica se participantType é igual a uma dessas três alternativas. No caso das crianças, estas podem dividir-se ainda em “toddler”, “kid”, “teen” ou “all children”, sendo que a restrição `check` cRank verifica essa atribuição no atributo childrenRank. Se participantType for diferente de “children” (ou seja, para adultos ou para todos), childrenRank pode ser `null` (condição também verificada na restrição `check` cRank).

Restaurant:

- A hora de abertura tem que ser antes da hora de fecho por razões lógicas, logo closingHour > openingHour (restrição `check` hourCheck).
- O número de cadeiras tem que ser obrigatoriamente maior que o número de mesas, pois cada mesa tem de ter obrigatoriamente uma ou mais cadeiras, ou seja, numberOfChairs > numberOfTables (restrição `check` tableCheck).

Cleaning:

- Não tem restrições, tirando a `foreign key` acima mencionada.

Person:

- `Unique` em idPerson, visto que é a `primary key` da relação.

Tal como em Service, as duas relações derivadas de Person possuem uma `foreign key`, per_fkey, para uma Person, sendo que também é uma `primary key` da relação e, por conseguinte, `unique`. Também será omitida das relações abaixo pela mesma razão.

Employee:

- Tem uma `foreign key`, ser_fkey, que aponta para o Service a que o Employee está associado.

Client:

- Se o Client não for uma criança, isChild toma o valor de 0 (false) e childRank e idParent tomam of valor `null`. Se for uma criança, isChild toma o valor 1 (true) e childRank toma um de três valores, “toddler”, “kid” ou “teen”. idParent torna-se o índice de uma `foreign key`, parent_fkey, que referencia o Client encarregado da criança. Estas condições são avaliadas na restrição `check` rankCheck.

ClientReservation:

- Tem duas `foreign keys`, per_fkey e resv_fkey, que referenciam, respectivamente, o Client da que realizou a Reservation e a Reservation em si. Estas `foreign keys` são também as `primary keys` da relação e, portanto, `unique`.

LanguagePerson:

- Tem duas `foreign keys`, per_fkey e lang_fkey, que referenciam, respectivamente, uma Person e uma Language. Estas `foreign keys` são também as `primary keys` da relação e, portanto, `unique`.

ClientEntertainment:

- Tem duas `foreign keys`, per_fkey e ser_fkey, que associam uma Person a um Entertainment. Estas `foreign keys` são também as `primary keys` da relação e, portanto, `unique`.

* As palavras escritas `neste tipo de letra` são palavras com equivalência direta para a sintaxe do SQLite.

Interrogação da Base de Dados

Foram definidas as seguintes interrogações à base de dados:

1. Número de pessoas que fala cada língua;
2. Quais as pessoas que vão chegar ao hotel, a partir do dia atual, e ordenadas pela chegada mais próxima;
3. O número de dias que cada quarto está ocupado;
4. Os dez empregados com melhor salário mensal;
5. Estatísticas sobre as pontuações das análises (número de análises, média, pontuação mais alta e mais baixa);
6. O cliente com a reserva mais longa (em dias);
7. O número de pessoas que participou em cada atividade de entretenimento;
8. O número total de pessoas de uma reserva (clientes dessa reserva mais as suas crianças);
9. A comparação de cada pontuação de uma análise com o número de vezes que a pessoa que escreveu a análise participou nas atividades de entretenimento;
10. Média do número de atividades de entretenimento por semana (considerando apenas semanas com pelo menos uma atividade de entretenimento).

Gatilhos Implementados

Foram definidos e implementados os três seguintes gatilhos:

1. Um gatilho que implementa uma das restrições de **Meal,** verificando se o tipo de refeição inserido é concordante com o tipo de refeição especificado na reserva do cliente, e verificando ainda se a data da refeição é válida;
2. Um gatilho que, ao remover uma linguagem, remove também a associação dessa linguagem às pessoas que a falam;
3. Um gatilho que implementa uma das restrições de **ClientEntretainment**, em que um certo cliente só pode participar dependendo do seu estatuto como criança ou adulto. Para o caso das crianças, depende ainda do estatuto da criança.