Especialização em *Business Intelligence*
Unidade Curricular de Sistemas Operacionais

Ano Letivo de 2016/2017
1º Semestre

Site XXI

Gil Gonçalves (a67738)

Luís Paulo Pedro (a70415)

José Pedro Monteiro (a73014)

Bruno Ribeiro (a73269)

Dezembro, 2016

Data de Recepção	
Responsável	
Avaliação	
Observações	

SITE XXI

Gil Gonçalves (a67738)

Luís Paulo Pedro (a70415)

José Pedro Monteiro (a73014)

Bruno Ribeiro (a73269)

Dezembro, 2017

Resumo

O presente documento apresenta o desenvolvimento de um projeto contém toda a informação sobre o processo de elaboração de uma arquitetura de base de dados, que será a base para a informatização da venda de jogos online. Este projeto surgiu com a necessidade modificar o seu sistema de vendas tornando este mais atual, e mais acessível ao consumidor. Este relatório mostra toda a fase de desenvolvimento do Sistema de Base de Dados que irá albergar a futura loja online, desde da sua fase embrionária, ou seja, desde a analise do caso de estudo, até á fase de implementação propriamente dita.

Sendo assim, ao longo deste trabalho vamos apresentar todas as fases necessárias para a construção de uma base de dados, à qual nos baseamos no livro "*Database Systems, A Practical Approach to Design, Implementation, and Management*" de Thomas Connolly e Carolyn Begg (M.Connolly & E.Begg, 2015).

Área de Aplicação: Desenho e arquitectura de Sistemas de Bases de Dados.

Palavras-Chave: Bases de Dados Relacionais, Sistemas Operacionais, *Business Intelligence,* Entidades, Relacionamentos, Metodologia, Modelo Conceptual, Modelo Lógico, Modelo Físico*,* SGBD, DBDL.

Índice

Resumo iii

Índice iv

Índice de Figuras vi

Índice de Tabelas vii

1. Introdução 1

1.1. Contextualização 1

1.2. Apresentação do Caso de Estudo 1

1.3. Motivação e Objetivos 2

1.4. Estrutura do Relatório 3

2. Especificação e Análise de Requisitos 4

2.1. Propósito da Base de Dados 4

2.2. Objetivo da Base de Dados 4

2.3. Clientes, Proprietário, Produtores e outros *Stackholders* 4

2.4. Utilizadores do Produto 5

2.5. Requisitos Funcionais 6

2.6. Requisitos Operacionais 7

3. Modelo de Dados Conceptual 8

3.1. Identificar Entidades 8

3.2. Identificar Relacionamentos entre Entidades 9

3.3. Identificar e Associar Atributos com Entidades ou Relacionamentos 11

3.4. Determinar o Domínio dos Atributos 13

3.5. Determinar Chaves Candidatas, Primárias e Alternativas 14

3.6. Verificar Existência de Redundâncias 15

3.7. Validar o Modelo de Dados Conceptual através das Transações do Utilizador 15

3.8. Revisão do Modelo de Dados Conceptual com os Utilizadores 16

4. Modelo de Dados Lógico 18
4.1. Derivar Relações para o Modelo de Dados Lógico 18
4.2. Validar Relações através da Normalização 20
4.3. Validar Relações através das Transações do Utilizador 21
4.4. Verificar Restrições de Integridade 23
4.5. Tamanho Inicial e Análise do Crescimento Futuro 25
4.6. Revisão do Modelo de Dados Lógico com os Utilizadores 26
5. Modelo de Dados Físico 27
5.1. Traduzir o Modelo de Dados Lógico para o SGBD 27
5.1.1 Desenho das Relações Base 27
5.1.2 Desenho das Restrições 30
5.2. Análise das Transações 31
5.3. Escolha dos Índices 36
5.4. Estimativa dos Requisitos do Espaço em Disco 37
5.5. Definição das Vistas dos Utilizadores e Regras de Acesso 38
6. Análise da Qualidade dos Dados 40
7. Ferramentas Utilizadas 42
8. Conclusões e Trabalho Futuro 43
Bibliografia 44
Lista de Siglas e Acrónimos 45
Anexos 46
I. *Script* completo da criação do esquema físico 47
II. *Script* completo do povoamento do sistema operacional "sitexxi" 52
III. Definição de algumas vistas dos utilizadores 56

Índice de Figuras

Figura 1 - Diagrama ER de relacionamentos entre entidades. 10
Figura 2 - Mapa de transações do modelo conceptual. 16
Figura 3 - Esquema conceptual do projeto desenvolvido no brModelo. 17
Figura 4 - Mapa de transações do modelo lógico. 22
Figura 5 - Mapa de transações dos acessos às tabelas. 33
Figura 6 - Resultado do teste da restrição de integridade. 40
Figura 7 - Resultado do teste da restrição de valor único. 40
Figura 8 - Resultado do teste da análise do conteúdo das tabelas Pais e Cidade. 41

Índice de Tabelas

Tabela 1 - Dicionário de dados das entidades. 9
Tabela 2 - Dicionário de dados dos relacionamentos. 10
Tabela 3 - Dicionário de dados dos atributos das entidades. 12
Tabela 4 - Dicionário de dados dos atributos dos relacionamentos. 12
Tabela 5 - Dicionário de dados com os domínios dos atributos das entidades. 13
Tabela 6 - Dicionário de dados com os domínios dos atributos dos relacionamentos. 13
Tabela 7 - Dicionário de dados das chaves primárias e alternativas das entidades. 15
Tabela 8 - *Cross-referencing* entre transações e relações. 32

1. Introdução

1.1. Contextualização

Com a evolução da Internet, muitas novas oportunidades de negócio surgiram, sendo sem dúvida alguma o comércio online uma das mais rentáveis e que cada vez mais tende a subir. Quando anteriormente era indispensável deslocarmo-nos para comprar um produto, agora este está à distância de alguns cliques, fazendo com que se deixasse de precisar de espaço físico para se poder construir um negócio. Um bom exemplo disto é a venda de videojogos em plataformas online.

O modelo de negócio consiste na compra de *keys* aos produtores dos videojogos para posteriormente colocar à venda no site da empresa. Para efetuar uma compra, o cliente apenas tem de aceder ao site da empresa, escolher o produto e fazer uma transferência bancária, possuindo algumas alternativas para realizar esta, entre as quais o MultiBanco, Paypal, entre outras.

Este modelo é muito mais prático para o utilizador, pois não precisa de se deslocar nem de esperar pela chegada de uma caixa com o videojogo, esperando apenas pela confirmação da transferência bancária, sendo esta processada normalmente com rapidez. Depois de confirmada a transferência, este recebe a *key* do jogo, e acedendo a plataformas como a Steam, pode usufruir da sua recente compra.

Todos estes benefícios fizeram com que este modelo seja atualmente utilizado amplamente, e juntando a isto técnicas de *Business Intelligence* como por exemplo *profiling*, é possível rentabilizar o negócio.

1.2. Apresentação do Caso de Estudo

De forma a recuperar a hegemonia anteriormente conquistada na área da venda de videojogos, esta empresa pretende mudar o seu modelo de negócio antiquado para um modelo de negócio que entende poder a vir ser mais rentável num futuro próximo.

Ao invés de possuir lojas físicas e consequentemente funcionários, esta pretende desfazer-se deste modelo e optar por construir uma plataforma online para a venda dos seus videojogos. Com isto, pretende também deixar de vender os famosos cd de videojogos,

passando assim apenas a vender *keys* que poderão ser utilizadas em plataformas online de videojogos, como, por exemplo, a Steam.

De forma a usufruir dos serviços da empresa, um cliente tem de se registar na plataforma online com a sua informação pessoal (nomeadamente nome, data de nascimento, pais e cidade de residência, sexo e opcionalmente o número de telemóvel) e o seu email. Este, para puder realizar compras online, precisa de fazer posteriormente uma transferência bancária para a referência da empresa, que depois ao confirmar o pagamento, envia a este um email com a(s) *key(s)* que este adquiriu. Com todos estes dados, é possível ao cliente ver o seu histórico de compras, requisitar o envio das faturas para o seu e-mail, consultar as pré-vendas que adquiriu, entre outros.

Já em relação aos jogos para venda, a empresa necessita de possuir algumas informações sobre estes, como os produtores dos mesmos, a data de lançamento, o nome e a idade mínima para se poder jogar estes. De forma a controlar o stock e a definir o preçário, é necessário conter a informação do stock atual de cada jogo, bem como o preço de venda associado a estes, e também possíveis descontos momentâneos que se possam atribuir aos mesmos.

De forma a facilitar ao cliente o processo de escolha dos produtos a comprar, os jogos devem estar categorizados e deverá ser possível aos clientes fornecerem a sua opinião relativa aos jogos que compraram, numa escala de satisfação entre 1 (Muito Mau) e 5 (Excelente). Esta avaliação feita pelos clientes aos jogos que compraram tem como objetivo facilitar a escolha dos produtos a comprar, isto com base na opinião que outros clientes que adquiriram o mesmo produto forneceram.

1.3. Motivação e Objetivos

Os principais motivos para o arranque deste projeto centram-se, como a cima referido, na atualização do modelo de negócio da empresa de forma a acompanhar o desenvolvimento atual na área dos videojogos, e seguir lado a lado com a evolução tecnológica de modo a se manter ativa no mercado o maior tempo possível. Outro dos motivos está relacionado com a necessidade de aumentar os lucros da empresa, e que para isso necessita de um serviço eficaz na atribuição de promoções dirigidas ao perfil do cliente, de fácil usabilidade e que seja rápido a efetuar uma compra, ou seja, a transação do pagamento assim como a atribuição da *key* seja feita com a maior rapidez possível.

Com este sistema operacional, a empresa tem toda a sua informação centralizada e organizada, para posteriormente ser aplicado *Business Intelligence* de forma a analisar a sua evolução e tirar as devidas conclusões com os dados das vendas que vão sendo recolhidos constantemente.

1.4. Estrutura do Relatório

O segundo capítulo aborda o levantamento e analise de requisitos, que irá conter toda a informação relevante para o desenho do sistema de base de dados em questão.

No terceiro capítulo será desenvolvido o modelo conceptual que acontece diretamente a partir do levantamento de requisitos. Para o desenrolar deste capitulo é importante que este seja devidamente documentado assim como testado para os requisitos considerados, de forma a obter o modelo conceptual pretendido.

O quarto capítulo é dedicado ao modelo lógico, sendo que na primeira secção se apresenta a derivação das relações do modelo conceptual para este novo modelo. Estas relações serão validadas através da normalização. De seguida, o modelo irá ser novamente validado segundo as transações definidas pelo utilizador. Neste capítulo é feita também a validação das restrições de integridade do sistema e uma análise do crescimento futuro da base de dados. Por fim, é feita uma revisão do modelo com o utilizador de forma a validar este, para depois prosseguir para as etapas posteriores.

No quinto capítulo aborda-se a conceção do modelo físico, onde é feita a tradução do modelo anterior para o SGBD escolhido. De seguida, são apresentadas duas subsecções com o desenho das relações base do modelo e das restrições gerais, respetivamente. Posteriormente, vamos proceder à análise das principais transações do sistema, à escolha dos índices e também à estimativa dos requisitos de espaço em disco ocupado. Por fim, vamos proceder à definição das vistas dos utilizadores e das regras de acesso ao sistema.

O sexto capitulo é onde é feito a análise da qualidade dos dados, para tal avaliação é documentado todo o processo da utilização da ferramenta Data Cleaner.

No sétimo capítulo são apresentadas as ferramentas utilizadas ao longo do desenvolvimento deste projeto.

Finalmente é realizada uma apreciação crítica sobre o trabalho final, destacando os pontos fortes e fracos do mesmo e o trabalho que futuramente poderá ser desenvolvido. A bibliografia pode ser consultada nas últimas páginas deste documento.

2. Especificação e Análise de Requisitos

2.1. Propósito da Base de Dados

No contexto deste projeto faz sentido considerar quatro grandes entidades: os **clientes**, os **jogos**, as **categorias dos jogos** e os **produtores dos jogos**. Os clientes frequentaram a loja online para comprar os mais diversos jogos. Os jogos são os produtos que os clientes irão consumir. As categorias identificam os vários tipos de jogos. Os produtores dos jogos identificam quem produziu o jogo.

2.2. Objetivo da Base de Dados

O objetivo desta base de dados passa por albergar toda a informação necessária para representar o processo de vendas da empresa, para posteriormente auxiliar o gerente da empresa nas tomadas de decisões relativa aos seus utilizadores e aumentar os rendimentos da empresa, atraindo novos utilizadores ou então fazendo com que os clientes comprem ainda mais. Para evitar que o gerente compre jogos que depois não serão vendidos é necessário que o gerente possua um histórico das compras que tenham sido efetuadas e depois analisar se compensa ou não comprar certos jogos.

2.3. Clientes, Proprietário, Produtores e outros *Stackholders*

Clientes

Os clientes são pessoas que estão registadas na base de dados. Após o registo, o cliente poderá ou não efetuar várias compras de diferentes jogos, podendo comprar ou não várias vezes o mesmo jogo em quantidades variadas.

Proprietário

O Proprietário desempenha um papel importante na aplicação porque será ele a tomar as decisões necessárias para aumentar o rendimento da sua empresa, será ele que irá decidir que tipo de jogos a comprar, as quantidades, o tipo de promoções que irá fazer e a que jogos será aplicado essas promoções.

O Proprietário também será responsável em arranjar formas para garantir que os clientes não deixem de efetuar compras na sua empresa e recorram à concorrência.

Produtores

Os produtores desempenham um papel importante visto que são eles que irão fornecer os jogos à empresa. Mediante do número de jogos que serão fornecidos, o gerente poderá ou não negociar com o produtor para tentar reduzir o custo dos jogos, podendo depois fazer melhores descontos aos clientes, aumentando assim o número de compras e consequentemente o número de novos clientes devido às promoções que irá oferecer.

Outros *Stackholders*

Para além das três entidades acima enunciadas existem diversas outras que são indiretamente afetadas pela aplicação, podendo então interferir no seu funcionamento.

O gestor da base de dados desempenha um papel importante na medida que será ele o responsável pela qualidade dos dados apresentada ao gerente da empresa.

2.4. Utilizadores do Produto

Os utilizadores do produto

Os três principais agentes têm a si associadas diferentes responsabilidades, existindo por isso interfaces específicas para cada um deles. Estando os agentes em diferentes estágios de preparação tecnológica e tendo diferentes motivações, é necessário que cada uma delas siga princípios diferentes.

O **responsável da empresa**, necessita de uma interface simples e intuitiva, dando acesso a todas as estatísticas e funcionalidades principais de forma direta, sendo a sua motivação a maximização do lucro e o auxílio na gestão.

Ao **cliente** deve ser oferecido uma interface inteligente e altamente configurável, pois este é um público jovem e com grandes competências tecnológicas.

O **gestor da base de dados** deve ter acesso a toda a informação relativa aos clientes, as compras efetuadas e as principais empresas de comercialização de jogos. Será ele o

responsável pela criação dos relatórios que irá reunir toda esta informação para depois apresentar ao responsável da empresa.

Participação dos utilizadores

Sendo este um produto orientado a um mercado específico, as opiniões e conhecimentos dos diversos elementos envolvidos são fundamentais, devendo o seu desenvolvimento ser orientado para tal.

O **Proprietário da empresa** necessita de perceber as melhores estratégias para atrair clientes.

Os **Clientes** são o principal meio de obtenção de *feedback* geral, como os pontos positivos e negativos. Destes é também possível obter novas ideias de *marketing*, tipo de promoções a fazer, jogos mais comprados, altura do ano onde se compram mais jogos, etc.

O **Gestor da base de dados** vai ser o responsável por apresentar os relatórios ao gerente da empresa de forma a definir que tipo de estratégias irão ser usadas.

2.5. Requisitos Funcionais

De modo a ser criada uma base de dados capaz de guardar as informações dos jogos e dos clientes, foram levantados os seguintes requisitos funcionais para expressar e relacionar a devida informação corretamente:

1. A informação respetiva aos clientes será guardada tal como o seu número de cliente, nome, sexo, data de nascimento, data em que se registou número de telemóvel, cidade, país e email. A cada cliente estarão também associadas as várias compras que o cliente efetuou.

2. A base de dados deverá guardar toda informação relativa às compras de jogos, sendo que está constituída pela data em que o jogo foi comprado, o preço a que foi vendido, se foi feito algum desconto e qual foi, e a quantidade que foi comprada. Cada compra está evidentemente associada a um cliente e ao jogo que desejou adquirir.

3. Uma produtora desenvolve vários jogos, que serão posteriormente vendidos a vários clientes, sendo que estes se encontro categorizados.

4. Um cliente após efetuar uma aquisição, deverá ser possível avaliar o jogo que acabou de comprar, contribuindo assim para a sua classificação global.

5. Os dados relativos a um jogo também deverão ser guardados, mais concretamente, o seu preço base, o nome do jogo, o desconto atual aplicado, a

idade mínima que um utilizador tem de ter para poder comprar o jogo e posteriormente jogar, e por fim a data a que foi lançado.

2.6. Requisitos Operacionais

Para que o nosso sistema de base de dados esteja corretamente implementado, deve ser possível:

1. Consultar a lista de todos os clientes que se registaram na nossa plataforma;
2. Registar novos clientes;
3. Consultar todos os clientes que já efetuaram compras;
4. Identificar quais os picos de volume de compras semanalmente, mensalmente e anualmente;
5. Registo de uma compra de um jogo por um cliente;
6. Registo de um novo jogo para venda e associar/registar o respetivo produtor;
7. Identificar jogos mais vendidos por faixa etária, e qual a faixa etária que mais utiliza a loja online;
8. Identificar quais os jogos mais vendidos semanalmente, mensalmente e anualmente;
9. Consultar quais são os clientes que tem maior volume de compras;
10. Consultar os valores das avaliações dadas pelo cliente sobre o aluguer para medir a qualidade de serviço;
11. Consultar quais cidades e quais países que mais adquirem jogos;
12. Consultar qual a produtora vende mais jogos;
13. Consultar jogos com melhor e pior avaliação;
14. Consultar os diferentes volumes de venda, com promoção e sem promoção.

3. Modelo de Dados Conceptual

3.1. Identificar Entidades

Neste primeiro passo, e com base na análise dos requisitos apresentada no capítulo anterior, vamos proceder à identificação das entidades a representar no modelo conceptual.

Através da análise dos requisitos, identificámos quatro entidades:

- Cliente: representa todos os clientes registados da loja online *Site XXI*. Como anteriormente referido, um cliente para efetuar uma compra precisa de estar devidamente registado na plataforma, pelo que é necessária esta entidade para possibilitar o cumprimento deste requisito;
- Jogo: representa todos os videojogos disponíveis na loja online. Como este é o produto que a loja tem para vender, e como existe inúmeros jogos diferentes para vender, é necessário a criação desta entidade para os podermos caracterizar com detalhe de modo a serem facilmente identificados;
- Produtor: representa todos os produtores dos videojogos à venda na loja virtual.
- Categoria: representa todas as categorias de jogos disponíveis, na qual cada jogo pode ser inserido em uma ou mais.

De seguida, apresentamos o dicionário de dados representativo das entidades com as suas informações detalhadas.

Entidade	**Descrição**	***Aliases***	**Ocorrência**
Cliente	Termo geral que descreve todos os utilizadores registados da loja online *Site XXI*.	Utilizador	Cada utilizador da loja online que se encontra registado.
Jogo	Termo geral que descreve todos os jogos para venda na loja virtual.	Produto	Cada jogo pode ser obtido por qualquer utilizador, desde que esta tenha a idade mínima para poder jogar o mesmo e que ainda estejam disponíveis para venda *keys* do mesmo.

Produtor	Termo geral que descreve todos os produtores dos videojogos presentes na loja online.	Desenvolvedor	Cada editora possui um número variado de videojogos.
Categoria	Termo geral que descreve todas as categorias de videojogos presentes na loja virtual.	Género	Cada categoria de jogos a serem vendidos pelo *Site XXI*.

Tabela 1 - Dicionário de dados das entidades.

3.2. Identificar Relacionamentos entre Entidades

Após a identificação das entidades, é necessário identificar e caracterizar os relacionamentos que existem entre as mesmas. A partir da análise de requisitos, identificámos quatro relacionamentos entre entidades:

- **Compra**:
  i. Relacionamento: relaciona as entidades Cliente e Jogo;
  ii. Descrição: identifica todas as compras de jogos feitas pelos clientes da loja online;
  iii. Cardinalidade: relacionamento com cardinalidade N:N, na qual um cliente pode comprar N jogos, e cada jogo pode ser comprar N vezes por clientes;
  iv. Participação: Cliente (parcial) e Jogo (parcial), pois poderão existir clientes que nunca efetuaram a compra de um jogo, bem como podem existir jogos que até ao momento poderão nunca ter sido comprados.
- **Avalia:**
  i. Relacionamento: relaciona as entidades Cliente e Jogo;
  ii. Descrição: associa as avaliações aos jogos feitas pelos clientes, sendo necessário que um cliente tenha comprado pelo menos uma *key* para poder fazer uma única avaliação ao jogo;
  iii. Cardinalidade: relacionamento com cardinalidade N:N, na qual um cliente pode fazer uma avaliação por cada jogo que adquiriu, e cada jogo pode ser avaliado N vezes por clientes;
  iv. Participação: Cliente (parcial) e Jogo (parcial), pois poderão existir clientes que nunca fizeram avaliações aos jogos que adquiriram, e jogos que não contêm nenhuma avaliação.
- **Produzido:**
  i. Relacionamento: relaciona as entidades Jogo e Produtor;
  ii. Descrição: associa todos os produtores e os seus jogos produzidos que se encontram à venda na loja virtual;
  iii. Cardinalidade: relacionamento com cardinalidade 1:N, na qual um jogo é apenas produzido por um produtor, e um produtor pode ter desenvolvido mais do que um jogo;

iv. Participação: Jogo (total) e Produtor (total), pois a todos os jogos têm de corresponder a um produtor, e todos os produtores têm de ter pelo menos um jogo produzido.

- **Tem:**
  i. Relacionamento: relaciona as entidades Jogo e Categoria;
  ii. Descrição: associa os jogos e as categorias onde estes se inserem;
  iii. Cardinalidade: relacionamento com cardinalidade N:N, na qual um jogo pode estar inserido em mais do que uma categoria, e uma categoria pode conter vários jogos;
  iv. Participação: Jogo (total) e Categoria (parcial), pois todos os jogos têm de ser inseridos em pelo menos uma categoria, no entanto uma categoria pode não possuir algum jogo;

De seguida, encontra-se um diagrama ER parcial descritivo do esquema conceptual até ao momento, contendo as entidades e os relacionamentos entre as mesmas, e um dicionário de dados contendo em forma tabular e resumida a informação sobre os relacionamentos.

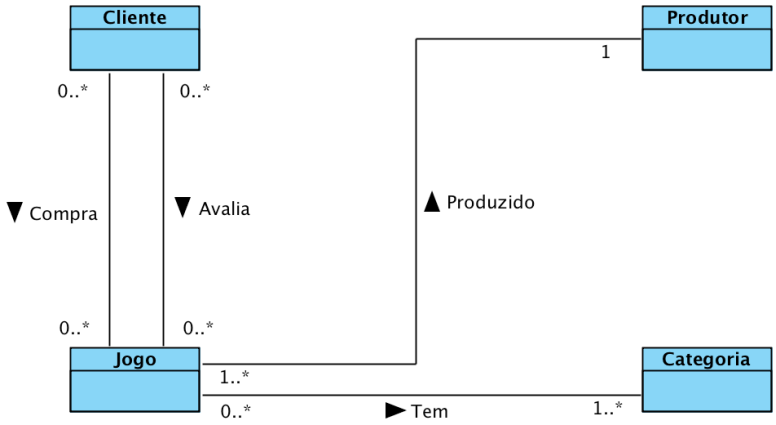


Figura 1 - Diagrama ER de relacionamentos entre entidades.

Entidade	Multiplicidade	Relacionamento	Multiplicidade	Entidade
Cliente	(0,N)	Compra	(0,N)	Jogo
Cliente	(0,N)	Avalia	(0,N)	Jogo
Jogo	(1,N)	Produzido	(1,1)	Produtor
Jogo	(0,N)	Tem	(1,N)	Categoria

Tabela 2 - Dicionário de dados dos relacionamentos.

3.3. Identificar e Associar Atributos com Entidades ou Relacionamentos

Depois de identificadas as entidades e os relacionamentos, é necessário identificar os atributos de fazem parte de cada, caracterizando o seu nome, descrição, tipo e tamanho de dados, se pode ser nulo, multivalor, derivado ou se possui algum valor *default*. De seguida são apresentados os dicionários de dados com todos os atributos identificados para as entidades e os relacionamentos, respetivamente.

Entidade	**Atributos**	**Descrição**	**Tipo e Tamanho**	***Nulls***	**Multivalor**	**Derivado**	**Valor *Default***
Cliente	nome	Nome do cliente	100 caracteres variáveis	Não	Não	Não	Não
	email	Email do cliente	50 caracteres variáveis	Não	Não	Não	Não
	data_nascimento	Data de nascimento do cliente	Data	Não	Não	Não	Não
	sexo	Sexo do cliente	1 caracter variável	Não	Não	Não	Não
	nr_telemovel	Número de telemóvel do cliente	20 caracteres variáveis	Sim	Não	Não	Não
	cidade	Cidade onde o cliente reside	35 caracteres variáveis	Não	Não	Não	Não
	pais	País onde o cliente reside	35 caracteres variáveis	Não	Não	Não	Não
	data_registo	Data de registo da conta do cliente	Data	Não	Não	Não	Não
Jogo	nome	Nome do jogo	50 caracteres variáveis	Não	Não	Não	Não
	data_lancamento	Data de lançamento do jogo	Data	Não	Não	Não	Não
	idade_minima	Idade mínima para poder jogar o jogo	Número inteiro	Não	Não	Não	Não
	stock	Quantidade de *keys* do jogo disponíveis para venda	Número inteiro	Não	Não	Não	Não
	preco	Preço atual de cada *key* do jogo	Número decimal	Não	Não	Não	Não
	desconto	Desconto atual aplicado a cada	Número decimal	Não	Não	Não	Não

		key do jogo					
Produtor	nome	Nome do produtor de jogos	50 caracteres variáveis	Não	Não	Não	Não
Categoria	nome	Nome de uma categoria de jogos	25 caracteres variáveis	Não	Não	Não	Não

Tabela 3 - Dicionário de dados dos atributos das entidades.

Relacionamento	**Atributos**	**Descrição**	**Tipo e Tamanho**	**Nulls**	**Multivalor**	**Derivado**	**Valor Default**
Compra	quantidade	Quantidade comprada por um cliente de um dado jogo	Número inteiro	Não	Não	Não	Não
	preco	Preço por quantidade aplicado a cada *key* do jogo	Número decimal	Não	Não	Não	Não
	desconto	Desconto por quantidade aplicado a cada *key* do jogo	Número decimal	Não	Não	Não	Não
	data_compra	Data em que a compra do jogo foi efetuada	Data	Não	Não	Não	Não
Avalia	avaliação	Avaliação feita por um cliente sobre um jogo	Número inteiro	Não	Não	Não	Não
	descricao	Descrição justificativa da avaliação feita	Número indefinido de caracteres	Sim	Não	Não	Não
	data_avaliacao	Data em que a avaliação foi feita	Data	Não	Não	Não	Não

Tabela 4 - Dicionário de dados dos atributos dos relacionamentos.

3.4. Determinar o Domínio dos Atributos

Após a identificação e caracterização dos atributos referentes às entidades e aos relacionamentos, é necessário identificar o seu domínio, ou seja, o conjunto de valores que formam o valor final de um ou mais atributos. Posteriormente, estão tabelados os domínios de cada atributo de cada entidade e relacionamento, respetivamente.

Entidade	**Atributos**	**Domínio**
Cliente	nome	Qualquer sequência de caracteres
	email	*E-mail* válido
	data_nascimento	Data de nascimento válida
	sexo	Qualquer caracter
	nr_telemovel	Número de telemóvel válido
	cidade	Qual quer sequência de caracteres que represente uma cidade
	pais	Qualquer sequência de caracteres que represente um país
	data_registo	Data de registo válida
Jogo	nome	Qualquer sequência de caracteres
	data_lancamento	Data de lançamento válida
	idade_minima	Número inteiro maior que 0 e menor ou igual a 18
	stock	Número inteiro maior do que 0
	preco	Número decimal maior ou igual a 0
	desconto	Número decimal maior ou igual a 0 e menor ou igual a 100
Produtor	nome	Qualquer sequência de caracteres
Categoria	nome	Qualquer sequência de caracteres

Tabela 5 - Dicionário de dados com os domínios dos atributos das entidades.

Relacionamento	**Atributos**	**Domínio**
Compra	quantidade	Número inteiro maior que 0
	preco	Número decimal (2 casas decimais) maior que 0
	desconto	Número decimal (2 casas decimais) maior ou igual a 0 e menor ou igual a 100
	data_compra	Data de compra válida
Avalia	avaliacao	Número inteiro entre 1 e 5
	descricao	Qualquer sequência de caracteres
	data_avaliacao	Data de avaliação válida

Tabela 6 - Dicionário de dados com os domínios dos atributos dos relacionamentos.

3.5. Determinar Chaves Candidatas, Primárias e Alternativas

Nesta etapa é necessário determinar quais as chaves candidatas do conjunto de atributos das nossas entidades, para desta forma definir a chave primária de cada uma, e também as chaves alternativas.

Através da análise dos atributos de cada entidade, foram encontradas as seguintes chaves candidatas:

- Cliente: para a entidade Cliente apenas identificámos o atributo email como chave candidata, visto que o e-mail é único para cada cliente. Todos os outros atributos não são suficientes para identificar unicamente um cliente, como, por exemplo, o caso do nome pois pode haver duas pessoas com o mesmo nome. No entanto, optámos por criar a chave `id_cliente` para ser chave primária, sob a forma de um número inteiro, pois normalmente uma chave numérica é mais eficiente que uma chave sob a forma de uma *string*, como o caso do e-mail. O atributo email passou assim a ser chave alternativa;
- Jogo: para a entidade Jogo apenas identificámos o atributo nome como chave candidata, visto que o nome de um jogo é único. Todos os outros atributos não são suficientes para identificar unicamente um jogo pois, no caso da data de lançamento, vários jogos podem ser lançados no mesmo dia. Consequentemente, e pelas mesmas razões acima descritas, decidimos criar a chave `id_jogo` para ser a chave primária, sob a forma de um número inteiro, deixando para chave alternativa o nome do jogo;
- Produtor: para a entidade Produtor identificámos o atributo nome como chave candidata, pois um produtor tem uma marca registada, ou seja, um nome que apenas o identifica. Porém, e pelas mesmas razões descritas para a entidade Cliente e Jogo, decidimos criar a chave `id_produtor` para chave primária, ficando o atributo nome a ser a chave alternativa;
- Categoria: para a entidade Categoria identificámos o atributo nome como chave candidata, pois o nome da categoria é único. Contudo, pelas mesmas razões descritas para as entidades anteriores, optámos por criar a chave `id_categoria` para chave primária, passando para chave alternativa o atributo nome da categoria.

Seguidamente é apresentado o dicionário de dados das chaves primárias e alternativas definidas para cada entidade.

Entidade	Chave primária	Chaves Alternativas
Cliente	id_cliente	email
Jogo	id_jogo	nome
Produtor	id_produtor	nome
Categoria	id_categoria	nome

Tabela 7 - Dicionário de dados das chaves primárias e alternativas das entidades.

3.6. Verificar Existência de Redundâncias

Neste passo, é necessário analisa o modelo conceptual com o objetivo de verificar a existência de redundâncias e, em caso afirmativo, removê-las. Para tal, teremos de analisar

- Voltar a examinar os relacionamentos de 1:1;
- Remover relações redundantes;
- Verificar se modificações temporais podem causar redundâncias.

Em relação ao primeiro ponto, o nosso modelo não possui relacionamentos de 1:1 entre entidades, pelo que não existem redundâncias neste ponto.

Seguidamente, e já em relação ao segundo ponto, um relacionamento é redundante quando é possível obter a mesma informação à custa de outros relacionamentos. À primeira vista, é percetível que possa haver redundância pois existem dois relacionamentos entre as entidades Cliente e Jogo. Mas, como cada relacionamento tem um significado diferente, um regista as compras de jogos efetuadas pelos clientes e o outro regista as avaliações feitas pelos clientes aos jogos, é possível chegar à conclusão que não existe algum tipo de redundância.

Por fim, e em relação ao terceiro ponto, temos de ter em consideração que mudanças temporais, ou seja, à medida que passa o tempo, certos relacionamentos entre entidades possam tornar-se redundantes. No entanto, após verificação a verificação do modelo tendo em contas as circunstâncias acima descritas, não encontramos redundâncias, pelo que o modelo se encontra válido até este ponto.

3.7. Validar o Modelo de Dados Conceptual através das Transações do Utilizador

Neste momento está definido um modelo de dados conceptual que representa os requisitos impostos pela loja online *Site XXI*. Assim, justifica-se então a verificação de todo o modelo de forma a garantir que este consegue dar resposta a todas as necessidades imposta pelo utilizador.

De seguida, apresenta-se as principais três transações consideradas, a sua definição e descrição, e um mapa de transações que valida o nosso modelo:

a) Registo de uma compra de um jogo por um cliente: Inserção de uma compra efetuada por um dado cliente de um dado jogo. Neste caso, é utilizado o relacionamento Compra entre as entidades Cliente e Jogo para realizar o registo da compra;

b) Registo de um novo jogo para venda e associar/registar o respetivo produtor: Inserção de um novo jogo para venda mantido na entidade Jogo e associar o respetivo produtor mantido na entidade Produtor. Caso não exista o produtor, é necessário proceder à inserção deste. Neste caso, é utlizado o relacionamento Produzido entre as entidades Jogo e Produtor para realizar a associação dos mesmos.

c) Registo de cliente novo: Inserção de um novo cliente mantido na entidade Cliente.

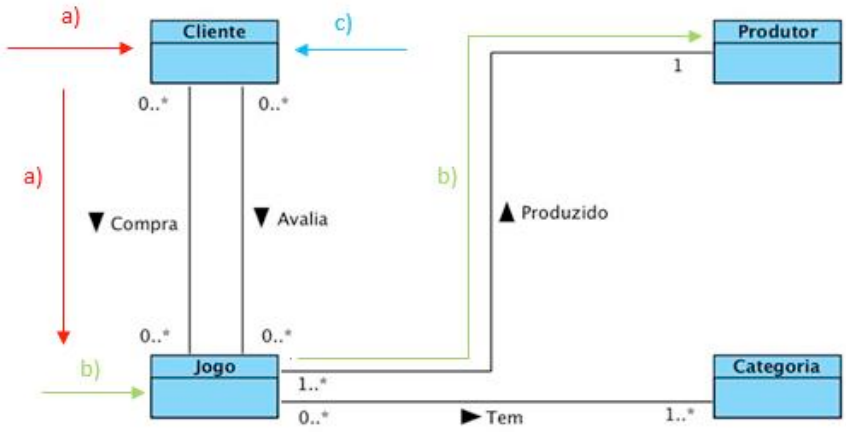


Figura 2 - Mapa de transações do modelo conceptual.

3.8. Revisão do Modelo de Dados Conceptual com os Utilizadores

Nesta última fase, estamos aptos a representar o esquema conceptual final de forma a procedermos a uma posterior avaliação do mesmo com os utilizadores finais do sistema, verificando se este corresponde ao que foi pedido e cumpre todos os requisitos especificados.

De seguida, é apresentado o esquema conceptual do projeto:

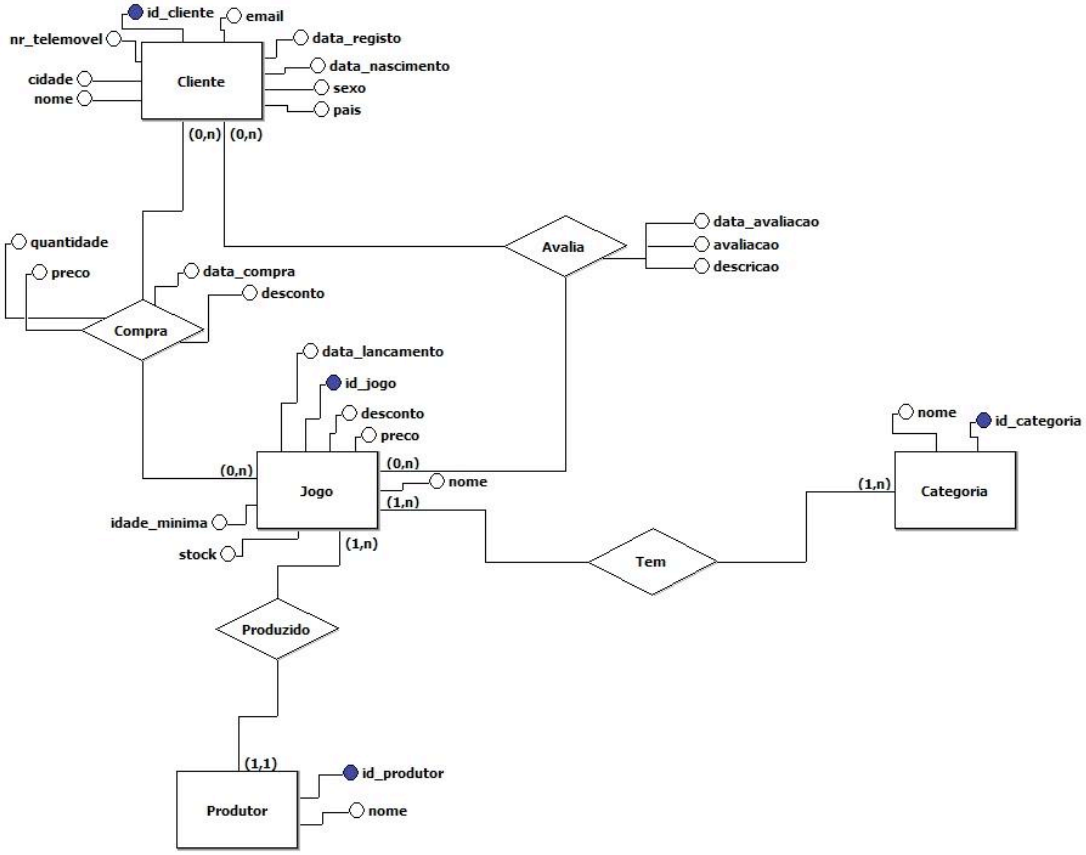


Figura 3 - Esquema conceptual do projeto desenvolvido no brModelo.

Na revisão do dicionário de entidades com os futuros utilizadores do sistema, podemos constatar que foram identificadas todas as entidades importantes no modelo e que correspondem às necessidades da empresa. Em relação ao dicionário de relacionamentos, foi justificada a forma como as entidades foram associadas e os utilizadores verificaram que estes relacionamentos definidos correspondem à realidade do problema. Posteriormente, foi verificada e validada a informação associada a cada entidade e validar o domínio de atributo, constatando que estes correspondem aos requisitos do problema. Por fim, foram analisadas as transações que o modelo teria de cumprir, constatando com os futuros utilizadores do sistema que este cumpre todas as transações impostas pelos requisitos, e também que o esquema conceptual representa de forma correta o futuro sistema a construir, pelo que foi aprovado pelo cliente o modelo de dados conceptual como sendo uma solução que dá resposta a todas as necessidades da empresa.

4. Modelo de Dados Lógico

4.1. Derivar Relações para o Modelo de Dados Lógico

Neste passo inicial vamos procurar derivar as relações que irão fazer parte do nosso modelo de dados lógico, através das entidades e relações anteriormente identificadas no modelo de dados conceptual.

Numa primeira fase, procurámos derivar as relações que representam as entidades. Para tal, por cada entidade forte, é necessário criar uma relação que inclua todos os atributos simples dessa mesma entidade, podendo posteriormente estas relações virem a ser alteradas devido ao tratamento dos relacionamentos, atributos multivalor, entre outros.

Já numa segunda fase, é necessário proceder à tradução dos relacionamentos identificados anteriormente para as respetivas relações do modelo de dados lógico, identificando as alterações que estas possam provocar nas relações já estabelecidas para as entidades.

Primeiramente, optámos por traduzir os relacionamentos 1:N, na qual apenas foi identificado um, o relacionamento Produzido, que relaciona cada jogo ao ser produtor. Neste caso, à relação Jogo é acrescentada a chave primária da entidade Produtor, como chave estrangeira.

Posteriormente, optámos por derivar as relações dos relacionamentos N:N, na qual existem três: Compra, que identifica cada compra de um jogo feita pelos clientes; Avalia, que relaciona cada avaliação feita pelos clientes aos jogos; Tem, que relaciona cada jogo às categorias onde se insere.

Para os relacionamentos Avalia e Tem, apenas é necessário a criação de uma relação para cada, cuja chave primária irá ser uma chave composta pelas chaves primárias das entidades que cada uma relaciona, contendo também, no caso do relacionamento Avalia, o conjunto de atributos do relacionamento identificados anteriormente. Assim sendo, as relações criadas foram Avaliacao e JogoCategoria, para os relacionamentos Avalia e Tem respetivamente.

Já para o relacionamento Compra, é necessário a criação de uma relação, ao qual optámos pelo mesmo nome, com os seus atributos anteriormente identificados. Já em relação à sua chave primária, esta é composta apenas por um `id_compra`, que foi definido devido às

chaves estrangeiras `id_cliente` e `id_jogo` não serem suficientes para serem a chave primária, pois um mesmo cliente não poderia comprar o mesmo jogo duas vezes distintas.

Por fim, apresentamos de seguida todas as relações definidas usando a notação DBDL:

```
Cliente (id_cliente, nome, email, data_nascimento, sexo, nr_telemovel, cidade, pais, data_registo)
        Primary Key id_cliente
        Alternate Key email

Jogo (id_jogo, nome, data_lancamento, idade_minima, stock, preco, desconto, id_produtor)
        Primary Key id_jogo
        Alternate Key nome
        Foreign Key id_produtor references Produtor(id_produtor)

Produtor (id_produtor, nome)
        Primary Key id_produtor
        Alternate Key nome

Categoria (id_categoria, nome)
        Primary Key id_categoria
        Alternate Key nome

Avaliacao (id_cliente, id_jogo, avaliacao, descricao, data_avaliacao)
        Primary Key id_cliente, id_jogo
        Foreign Key id_cliente references Cliente(id_cliente)
        Foreign Key id_jogo references Jogo(id_jogo)

JogoCategoria (id_jogo, id_categoria)
        Primary Key id_jogo, id_categoria
        Foreign Key id_jogo references Jogo(id_jogo)
        Foreign Key id_categoria references Categoria(id_categoria)

Compra (id_compra, id_cliente, id_jogo, quantidade, preco, desconto, data_compra)
        Primary Key id_compra
        Alternate Key id_compra, id_cliente, id_jogo
        Foreign Key id_cliente references Cliente(id_cliente)
        Foreign Key id_jogo references Jogo(id_jogo)
```

4.2. Validar Relações através da Normalização

Após derivadas as relações que fazem parte do modelo de dados lógico, está na altura de as validar, aplicando-lhes as primeiras três formas normais.

Uma relação está na 1ª Forma Normal se cada atributo contém apenas valores atómicos e se não há conjuntos de atributos repetidos descrevendo a mesma característica. Analisando todas as nossas relações, o único aspeto que poderia levantar questões poderia ser o atributo telemóvel da relação Cliente, só que como o atributo é simples, ou seja, é constituído por apenas um número de telemóvel, encontra-se de acordo com a 1NF. Podemos então afirmar que as nossas relações estão de acordo com a 1NF.

Já em relação à 2ª Forma Normal, esta exige que uma relação esteja na 1NF e que os atributos que não fazem parte da chave primária dependam da totalidade da mesma. Resumidamente, depois de identificada a chave da tabela, se esta for composta apenas por um atributo ou por todos os atributos da relação, a relação encontra-se na 2NF, caso contrário é necessário verificar se todos os atributos que não são chave dependem da totalidade da mesma, como desde já foi referido. Neste caso, apenas a relação Avaliacao teve de ser verificada, e constatámos esta se encontra na 2NF, pois os atributos `avaliacao`, `descricao` e `data_avaliacao` correspondem a uma avaliação feita por um cliente sobre um jogo que comprou, pelo que são dependentes da totalidade da chave.

Por fim, temos de validar as relações através da 3ª Forma Normal, que nos diz que uma relação está na 3NF se está na 2NF e nenhum dos atributos não chave depende de outro também não chave. Após a verificação desta forma normal a cada relação do modelo de dados lógico, constatámos que a relação Cliente não está na 3NF, pois existe uma dependência entre cidade e país, visto que através do nome de uma cidade, facilmente sabemos a que país pertence. Então, de forma a respeitar a 3NF, criámos duas novas relações, denominadas Cidade e Pais, na qual a primeira irá conter um identificador numérico (chave primária), o nome da cidade e o identificador do país a que pertence, e a segunda irá conter um identificador número (chave primária) e o nome do país respetivo. A relação Cliente também sofreu modificações, contendo agora apenas a chave primária da cidade a que o cliente pertence, ao invés de conter o nome da cidade e do país.

Depois de validado o modelo segundo as primeiras três formas normais e considerando as modificações referidas anteriormente, o modelo atual (na notação DBDL) é o seguinte:

Cliente (id_cliente, nome, email, data_nascimento, sexo, nr_telemovel, data_registo, id_cidade)
    Primary Key id_cliente
    Alternate Key email
    Foreign Key id_cidade **references** Cidade(id_cidade)

Cidade (id_cidade, nome, id_pais)

```
        Primary Key id_cidade
        Alternate Key id_cidade, id_pais
        Foreign Key id_pais references Pais(id_pais)

Pais (id_pais, nome)
        Primary Key id_pais

Jogo (id_jogo, nome, data_lancamento, idade_minima, stock, preco, desconto,
id_produtor)
        Primary Key id_jogo
        Alternate Key nome
        Foreign Key id_produtor references Produtor(id_produtor)

Produtor (id_produtor, nome)
        Primary Key id_produtor
        Alternate Key nome

Categoria (id_categoria, nome)
        Primary Key id_categoria
        Alternate Key nome

Avaliacao (id_cliente, id_jogo, avaliacao, descricao, data_avaliacao)
        Primary Key id_cliente, id_jogo
        Foreign Key id_cliente references Cliente(id_cliente)
        Foreign Key id_jogo references Jogo(id_jogo)

JogoCategoria (id_jogo, id_categoria)
        Primary Key id_jogo, id_categoria
        Foreign Key id_jogo references Jogo(id_jogo)
        Foreign Key id_categoria references Categoria(id_categoria)

Compra (id_compra, id_cliente, id_jogo, quantidade, preco, desconto,
data_compra)
        Primary Key id_compra
        Alternate Key id_compra, id_cliente, id_jogo
        Foreign Key id_cliente references Cliente(id_cliente)
        Foreign Key id_jogo references Jogo(id_jogo)
```

4.3. Validar Relações através das Transações do Utilizador

Neste momento está definido um modelo de dados lógico que representa os requisitos impostos pela loja online *Site XXI*. Assim, justifica-se então a verificação de todo o modelo de

forma a garantir que este consegue dar resposta a todas as necessidades imposta pelo utilizador. Denota-se que este passo foi realizado aquando da modelação de dados conceptual, tendo sido validado.

De seguida, apresenta-se as principais três transações consideradas na modelação conceptual e que agora

a) Registo de uma compra de um jogo por um cliente: Inserção de uma compra efetuada por um dado cliente de um dado jogo. Neste caso, é utilizado o relacionamento Compra entre as entidades Cliente e Jogo para realizar o registo da compra;
b) Registo de um novo jogo para venda e associar/registar o respetivo produtor: Inserção de um novo jogo para venda mantido na entidade Jogo e associar o respetivo produtor mantido na entidade Produtor. Caso não exista o produtor, é necessário proceder à inserção deste. Neste caso, é utlizado o relacionamento Produzido entre as entidades Jogo e Produtor para realizar a associação dos mesmos.
c) Registo de cliente novo: Inserção de um novo cliente mantido na entidade Cliente.

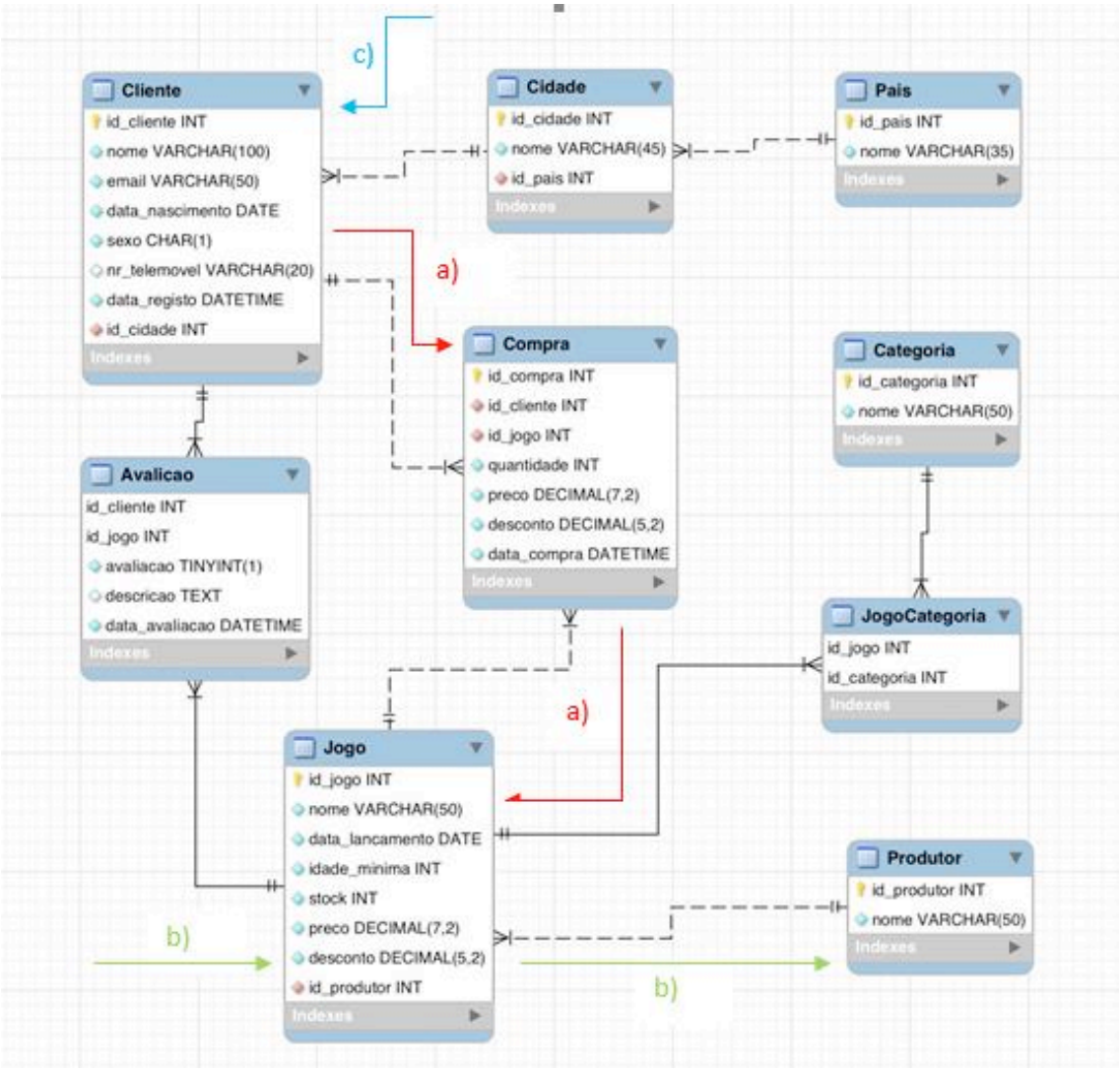


Figura 4 - Mapa de transações do modelo lógico.

4.4. Verificar Restrições de Integridade

De forma a garantir que o modelo de dados lógico representa de forma correta os requisitos impostos pela empresa *Site XXI*, necessitamos de garantir que todas as restrições de integridade estão representadas neste modelo de dados e se são de facto respeitadas.

Inicialmente, é necessário verificar os atributos que têm de conter um valor válido, ou seja, se estes não podem conter valores nulos, e também é necessário identificar e validar o domínio de cada atributo do modelo, de forma a que estes contenham sempre valores consistentes e válidos. Estas restrições estão identificadas nos capítulos 3 e 4 da modelação de dados conceptual, respetivamente.

De seguida, é necessário verificar as multiplicidades definidas para cada relacionamento entre entidades, de forma a que o tipo de relacionamento definido entre duas entidades esteja de acordo com as especificações da empresa, e também é necessário verificar a integridade das entidades, ou seja, se as suas chaves primárias não podem ser nulas. Estas restrições estão documentadas nos capítulos 2 e 5 da modelação de dados conceptual, respetivamente.

Posteriormente, é necessário definir as restrições de integridade para cada chave estrangeira de cada relação, de forma a garantirmos a consistência da nossa base de dados. Para tal, optámos pela aplicação do mesmo método de restrição de integridade para todas as chaves estrangeiras das nossas relações, que se caracteriza por no caso de ocorrer um *update* este refletir-se nos registos das relações em causa, e no caso de ocorrer um *delete* não deixar que alguma ação aconteça, não realizando a operação.

Depois de identificadas as restrições de integridade, é necessário definir as regras de negócio do modelo. Para tal, vão ser identificadas restrições gerais aos atributos de cada relação, isto caso existam. De seguida, apresentam-se as restrições definidas para cada relação:

- Cliente: o `email` é único; a `data_nascimento` tem de ser válida, ou seja, anterior à data atual da transação;
- Cidade: o `nome` é único;
- Pais: o `nome` é único;
- Jogo: o `nome` é único; a `data_lancamento` tem de ser válida, ou seja, anterior à data atual da transação; o `stock` é maior ou igual a 0; o `desconto` varia entre 0 e 100 inclusive; o `preco` é maior ou igual a 0; a `idade_minima` varia entre 0 e 18, inclusive;
- Produtor: o `nome` é único;
- Categoria: o `nome` é único;
- Avaliacao: a `avalicao` varia entre 1 e 5;
- Compra: a `quantidade` é maior do que 0; o `preco` é maior ou igual que 0; o `desconto` varia entre 0 e 100.

Por conseguinte, após a validação do modelo segundo as primeiras três formas normais e considerando as modificações referidas anteriormente, o modelo atual (na notação DBDL) é o seguinte:

```
Cliente (id_cliente, nome, email, data_nascimento, sexo, nr_telemovel, data_registo, id_cidade)
        Primary Key id_cliente
        Alternate Key email
        Foreign Key id_cidade references Cidade(id_cidade)
                    ON UPDATE CASCADE ON DELETE NO ACTION

Cidade (id_cidade, nome, id_pais)
        Primary Key id_cidade
        Alternate Key id_cidade, id_pais
        Foreign Key id_pais references Pais(id_pais)
                    ON UPDATE CASCADE ON DELETE NO ACTION

Pais (id_pais, nome)
        Primary Key id_pais

Jogo (id_jogo, nome, data_lancamento, idade_minima, stock, preco, desconto, id_produtor)
        Primary Key id_jogo
        Alternate Key nome
        Foreign Key id_produtor references Produtor(id_produtor)
                    ON UPDATE CASCADE ON DELETE NO ACTION

Produtor (id_produtor, nome)
        Primary Key id_produtor
        Alternate Key nome

Categoria (id_categoria, nome)
        Primary Key id_categoria
        Alternate Key nome

Avaliacao (id_cliente, id_jogo, avaliacao, descricao, data_avaliacao)
        Primary Key id_cliente, id_jogo
        Foreign Key id_cliente references Cliente(id_cliente)
                    ON UPDATE CASCADE ON DELETE NO ACTION
        Foreign Key id_jogo references Jogo(id_jogo)
                    ON UPDATE CASCADE ON DELETE NO ACTION

JogoCategoria (id_jogo, id_categoria)
        Primary Key id_jogo, id_categoria
        Foreign Key id_jogo references Jogo(id_jogo)
```

```
                ON UPDATE CASCADE ON DELETE NO ACTION
    Foreign Key id_categoria references Categoria(id_categoria)
                ON UPDATE CASCADE ON DELETE NO ACTION

Compra (id_compra, id_cliente, id_jogo, quantidade, preco, desconto,
data_compra)
    Primary Key id_compra
    Alternate Key id_compra, id_cliente, id_jogo
    Foreign Key id_cliente references Cliente(id_cliente)
                ON UPDATE CASCADE ON DELETE NO ACTION
    Foreign Key id_jogo references Jogo(id_jogo)
                ON UPDATE CASCADE ON DELETE NO ACTION
```

4.5. Tamanho Inicial e Análise do Crescimento Futuro

O objetivo deste tópico é apresentar uma estimativa inicial do tamanho da base de dados e analisar o seu crescimento futuro, tendo em conta as implicações que este crescimento possa provocar no modelo de dados da base de dados, devido à necessidade de inserção de novos atributos, entidades ou relacionamentos.

Inicialmente, e através da análise de requisitos, é estimado que ao fim do primeiro mês a base de dados terá: cerca de 100 clientes registados; cerca de 50 jogos diferentes à venda, com *stock* variável inicial de 5000 *keys*, produzidos por 30 diferentes produtores; cerca de 15 categorias de jogos diferentes; cerca de 250 compras feitas e cerca de 30 avaliações feitas por clientes.

Segundo uma estimativa feita pela empresa, estes esperam que existe um crescimento diário de 10 clientes e que as vendas de jogos aumentem em cerca de 5% ao mês. É previsto também que, em média, sejam adicionados cerca de 3 novos produtos por mês para venda, sendo que a reposição de stock é feita de acordo com a quantidade em stock atual de *keys* de cada produto.

Posto isto, e visto que este tipo de negócio é um pouco imprevisível, pois é um negócio que se insere na venda de jogos que é um mercado que tem muita evidência nos últimos anos, e as previsões podem ser muito flexíveis, e por isso o crescimento pode aumentar em grande escala comparado ao que foi projetado. Para tal, é necessário que o modelo de dados esteja preparado para responder a um grande aumento, e também tem de estar preparado para a definição de novos requisitos por parte do utilizador.

O modelo neste momento apenas está preparado para um sistema simples de vendas, na qual apenas se vende o jogo individualmente, podendo, no entanto, o cliente comprar várias *keys* do mesmo jogo ao mesmo tempo. No entanto, poderá ser necessária a inclusão de um sistema de vendas um pouco diferente, na qual se pode agora vender um *bundle* de jogos, de forma a diversificar os conteúdos à venda na loja online. Neste caso, o nosso modelo de dados

necessita de ser alterado, alterações estas que podem ser inseridas com alguma facilidade, mas que, no entanto, obriga a rever o modelo de dados definido.

Outras alterações poderão ser inseridas posteriormente, nomeadamente adicionar mais informações sobre os produtores, de forma a ter os contactos dos mesmos e a possuir um modelo de negócio em conjunto com estes. Assim sendo, o nosso modelo de dados poderá sofrer algumas alterações futuras dependendo das necessidades da empresa, que implicarão revisão no modelo, algo que não é muito favorável ao utilizador do sistema.

4.6. Revisão do Modelo de Dados Lógico com os Utilizadores

Nesta última fase da modelação lógica, é necessário proceder à revisão do modelo junto do utilizador com o objetivo de o validar segundo as especificações do mesmo.

Inicialmente, foi apresentado a tradução do modelo conceptual para o modelo lógico. Recorrendo ao dicionário de dados lógico, foi possível verificar o modelo a nível da consistência dos seus dados e verificar se este cumpre as transações definidas pelo cliente, na qual se constatou que este representa a realidade pretendida.

De seguida, foram especificadas as restrições ao nosso modelo, desde as restrições ao nível dos atributos, restrições de integridade e também as restrições de negócio. As restrições definidas foram prontamente validadas pelo cliente, que não expressou qualquer necessidade de retificações ao modelo.

5. Modelo de Dados Físico

5.1. Traduzir o Modelo de Dados Lógico para o SGBD

Neste passo inicial, vamos proceder à tradução das relações identificadas no modelo de dados lógico para a respetiva representação física, de modo a que as relações e as suas restrições possam ser suportadas pelo SGBD escolhido. Para tal, serão realizados apenas dois dos três passos, sendo o primeiro desenhar as relações base e o segundo desenho as restrições gerais (de negócio). O terceiro passo seria desenhar as representações dos dados derivados, mas como não possuímos dados derivados, omitimos esse passo.

5.1.1 Desenho das Relações Base

Inicialmente, é preciso colocar à disposição toda a informação sobre as relações apresentadas no modelo lógico de dados, desde a informação que diga respeito a domínios, valores por defeito, valores nulos, entre outros. De seguida, serão apresentadas todas as relações representadas na notação DBDL:

```
Cliente (
id_cliente          INT             NOT NULL AUTO_INCREMENT,
nome                VARCHAR(100)    NOT NULL,
email               VARCHAR(50)     NOT NULL,
data_nascimento     DATE            NOT NULL,
sexo                CHAR(1)         NOT NULL,
nr_telemovel        VARCHAR(20),
data_registo        DATETIME        NOT NULL,
id_cidade           INT             NOT NULL,
PRIMARY KEY (id_cliente),
FOREIGN KEY (id_cidade) REFERENCES Cidade(id_cidade)
                    ON UPDATE CASCADE ON DELETE NO ACTION
);

Cidade (
id_cidade           INT             NOT NULL AUTO_INCREMENT,
```

```
nome                VARCHAR(35)         NOT NULL,
id_pais             INT                 NOT NULL,
PRIMARY KEY (id_cidade),
FOREIGN KEY (id_pais) REFERENCES Pais(id_pais)
                    ON UPDATE CASCADE ON DELETE NO ACTION
);

Pais (
id_pais             INT                 NOT NULL AUTO_INCREMENT,
nome                VARCHAR(35)         NOT NULL,
PRIMARY KEY (id_pais)
);

Jogo (
id_jogo             INT                 NOT NULL AUTO_INCREMENT,
nome                VARCHAR(50)         NOT NULL,
data_lancamento     DATE                NOT NULL,
idade_minima        INT                 NOT NULL,
stock               INT                 NOT NULL,
preco               DECIMAL(7,2)        NOT NULL,
desconto            DECIMAL(5,2)        NOT NULL,
id_produtor         INT                 NOT NULL,
PRIMARY KEY (id_jogo),
FOREIGN KEY (id_produtor) REFERENCES Produtor(id_produtor)
                    ON UPDATE CASCADE ON DELETE NO ACTION
);

Produtor (
id_produtor         INT                 NOT NULL AUTO_INCREMENT,
nome                VARCHAR(50)         NOT NULL,
PRIMARY KEY (id_produtor)
);

Categoria (
id_categoria        INT                 NOT NULL AUTO_INCREMENT,
nome                VARCHAR(25)         NOT NULL,
PRIMARY KEY (id_categoria)
);

Avaliacao (
```

```
id_cliente          INT             NOT NULL,
id_jogo             INT             NOT NULL,
avaliacao           TINYINT(1)      NOT NULL,
descricao           TEXT,
data_avaliacao      DATETIME        NOT NULL,
PRIMARY KEY (id_cliente, id_jogo),
FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente)
                    ON UPDATE CASCADE ON DELETE NO ACTION,
FOREIGN KEY (id_jogo) REFERENCES Jogo(id_jogo)
                    ON UPDATE CASCADE ON DELETE NO ACTION
);

JogoCategoria (
id_jogo             INT             NOT NULL,
id_categoria        INT             NOT NULL,
PRIMARY KEY (id_jogo, id_categoria),
FOREIGN KEY (id_jogo) REFERENCES Jogo(id_jogo)
                    ON UPDATE CASCADE ON DELETE NO ACTION,
FOREIGN KEY (id_categoria) REFERENCES Categoria(id_categoria)
                    ON UPDATE CASCADE ON DELETE NO ACTION
);

Compra (
id_compra           INT             NOT NULL AUTO_INCREMENT,
id_cliente          INT             NOT NULL,
id_jogo             INT             NOT NULL,
quantidade          INT             NOT NULL,
preco               DECIMAL(7,2)    NOT NULL,
desconto            DECIMAL(5,2)    NOT NULL,
data_compra         DATETIME        NOT NULL,
PRIMARY KEY (id_compra),
FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente)
                    ON UPDATE CASCADE ON DELETE NO ACTION
FOREIGN KEY (id_jogo) REFERENCES Jogo(id_jogo)
                    ON UPDATE CASCADE ON DELETE NO ACTION
);
```

5.1.2 Desenho das Restrições

Depois de desenhadas as relações base do nosso modelo, é agora necessário definir as restrições gerais de forma a serem implementadas no SGDB. Para cada uma das restrições de negócio, será apresentada uma alternativa implementada em SQL standard, podendo, no entanto, serem utilizados outros mecanismos para a implementação de uma mesma restrição. Certas implementações das restrições de negócio poderão não funcionar em certos SGBDs, pelo que será necessário recorrer a outros mecanismo para as implementar.

De seguida, são apresentadas todas as restrições de negócio e consequentes implementações:

- Cliente:
    1. O `email` de um cliente tem de ser único: acrescentar a cláusula `UNIQUE` à definição do atributo na relação;
    2. A `data_nascimento` de um cliente tem de ser anterior à data atual: acrescentar a seguinte cláusula à definição do atributo na relação:
       ```
       CHECK ((datediff(curdate(),data_nascimento))<0)
       ```
    3. O `sexo` de um cliente apenas pode ser composto por 'M' ou 'F': acrescentar a seguinte cláusula à definição do atributo na relação:
       ```
       CHECK(sexo='M' OR sexo='F')
       ```

- Cidade:
    1. O `nome` de uma cidade é único: acrescentar a cláusula `UNIQUE` à definição do atributo na relação.
- Pais:
    1. O `nome` de um país é único: acrescentar a cláusula `UNIQUE` à definição do atributo na relação.
- Jogo:
    1. O `nome` de um jogo é único: acrescentar a cláusula `UNIQUE` à definição do atributo na relação;
    2. A `data_lancamento` de um jogo tem de ser anterior à data atual: acrescentar a seguinte cláusula à definição do atributo na relação:
       ```
       CHECK((datediff(curdate(),data_ lancamento))<0)
       ```
    3. O `stock` de um jogo é maior ou igual a 0: acrescentar a cláusula `UNSIGNED` à definição do atributo na relação;
    4. O `desconto` de um jogo é maior ou igual a 0 e menor ou igual a 100: acrescentar as cláusulas `UNSIGNED` e `CHECK(desconto<=100.00)` à definição do atributo na relação;
    5. O `preco` de um jogo é maior ou igual a 0: acrescentar a cláusula `UNSIGNED` à definição do atributo na relação;

6. A `idade_minima` de um jogo varia entre 0 e 18: acrescentar as cláusulas `UNSIGNED` e `CHECK(idade_minima <=18)` à definição do atributo na relação.

- Produtor:
    1. O `nome` de um produtor é único: acrescentar a cláusula `UNIQUE` à definição do atributo na relação.
- Categoria:
    1. O `nome` de uma categoria é único: acrescentar a cláusula `UNIQUE` à definição do atributo na relação.
- Avaliacao:
    1. A `avaliacao` de uma categoria é única: acrescentar a cláusula `CHECK(avaliacao>=1 AND avaliacao<=5)` à definição do atributo na relação.
- Compra:
    - A `quantidade` comprada é maior do que 0: acrescentar as cláusulas `UNSIGNED` e `CHECK(quantidade>0)` à definição do atributo na relação;
    - O `preco` de um jogo é maior ou igual a 0: acrescentar a cláusula `UNSIGNED` à definição do atributo na relação;
    - O `desconto` de um jogo é maior ou igual a 0 e menor ou igual a 100: acrescentar as cláusulas `UNSIGNED` e `CHECK(desconto<=100.00)` à definição do atributo na relação.

5.2. Análise das Transações

Nesta secção iremos analisar mais detalhadamente as transações realizadas, mais precisamente as seguintes transações:

(A) Inserir a compra de um jogo feita por um cliente e atualizar o stock disponível do jogo;
(B) Inserir um jogo e adicionar/associar um produtor;
(C) Registo de um novo cliente;

Comprar um jogo por parte de um cliente

A entidade jogo é uma das entidades chaves neste universo. Deste modo, sempre que é registada uma compra é inserido um registo na tabela Compras. Como é óbvio, a compra resulta numa diminuição do stock do jogo, logo, é necessário que se atualize a tabela Jogo removendo a quantidade comprada.

Inserir um jogo e associar/criar respetiva produtora

Como um jogo tem sempre uma produtora associada, então é natural que sempre que inserimos um jogo temos de inserir na tabela JogoCategoria a relação existente entre o jogo e

a produtora, se essa mesmo produtora for nova, ou seja, se nunca tiver sido comprado nenhum jogo pertencente a essa mesmo produtora, esta será inserida.

Registo de um novo cliente

Existe, portanto, a necessidade de registar clientes, para isso é necessário inserir esse mesmo cliente na nossa BD, e atualizar a sua informação. Nomeadamente o email, que tem de ser obrigatoriamente único, a data de nascimento tem de ser válida e o número de telemóvel tem de ser único.

I=INSERT; R=READ; U=UPDATE; D=DELETE

TRANSACTION/	**(A)**				**(B)**				**(C)**			
RELATION	**I**	**R**	**U**	**D**	**I**	**R**	**U**	**D**	**I**	**R**	**U**	**D**
UTILIZADOR		X							X	X		
PAÍS												
CIDADE												
PRODUTOR					X	X						
JOGO		X	X		X							
CATEGORIA												
COMPRAS	X											
AVALIAÇÃO												

Tabela 8 - *Cross-referencing* entre transações e relações.

Determinar frequência de informação

Com base nos requisitos definidos, foi estimado que existem cerca de 100 clientes, que na loja havia 50 jogos diferentes, esses jogos pertencem a 30 produtoras distintas, no total existem um total de 5000 jogos, com uma média de 250 compras por mês. Através da tabela 1 para as transações (A), (B) e (C), verificámos que, devido à importância da relação Jogos, esta será sem dúvida a mais acedida, pelo que precisa de ser o mais eficiente possível. Para isso é importante analisar a média e o máximo número de transações por dia. Mas também verificar em que janela temporal ocorre o pico de transações, sendo que, no nosso caso em particular, ocorre entre as 20:00 e 22:00 durante a semana, e entre as 14:00 e 18:00 durante o fim-de-semana.

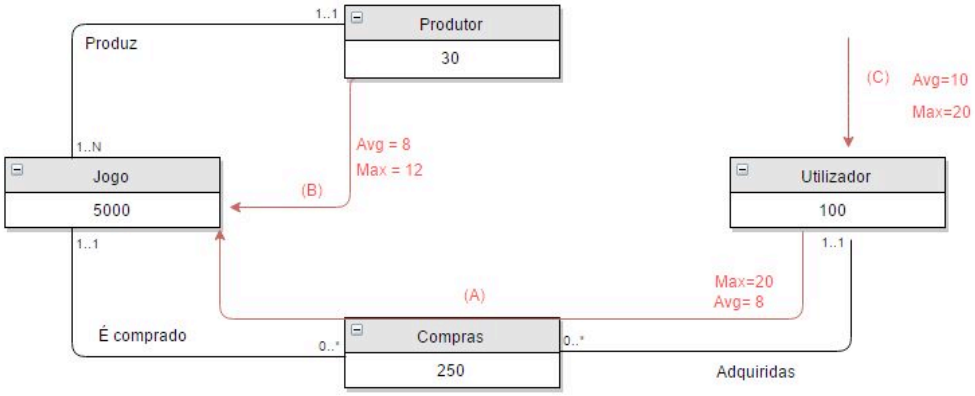


Figura 5 - Mapa de transações dos acessos às tabelas.

Analisar o uso dos dados

Tendo identificado as transações importantes, analisamos cada uma delas em mais detalhe. Para cada transação, devemos determinar:

- As relações e atributos acedidos pela transação e o tipo de acesso que é efetuado, ou seja, se é uma inserção, atualização, exclusão ou recuperação (também conhecido como uma consulta) transação.
- Os atributos utilizados em qualquer predicado (em SQL, os predicados são as condições especificado na cláusula WHERE).
- Para uma consulta, os atributos que estão envolvidos na associação de duas ou mais relações.
- A frequência esperada na qual a transação será executada.
- Os objetivos de desempenho para a transação.

Analise de transações para a ***transação (A):***

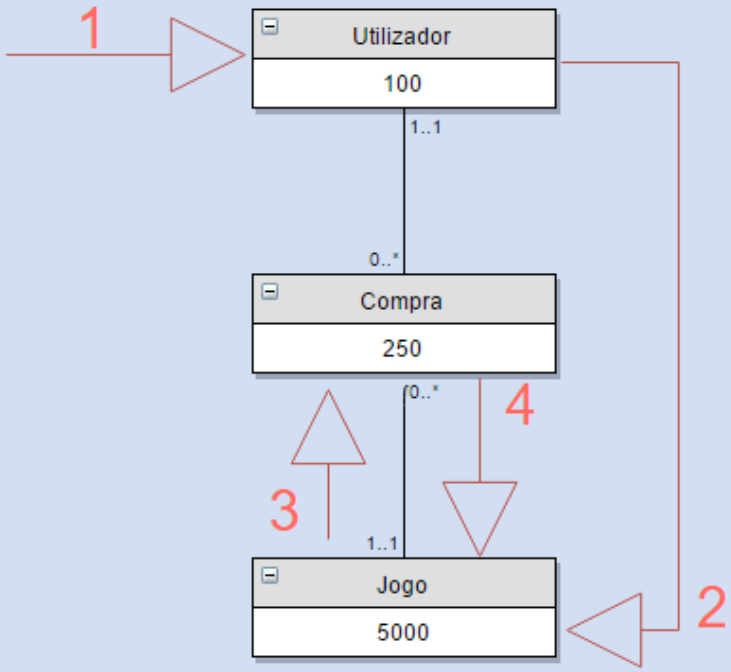
Transação	**(A)** Inserir a compra de um jogo feita por um cliente e atualizar a quantidade disponível do jogo;

Volume de Transações

Media: **8** por Dia

Pico de transações 20 por Dia (principal pico de atividade das 20:00 ás 22:00 Durante a semana, e 14:00 ás 16:00 no fim-de-sema**na**)

Mapa de Transações



Acesso	Entidade	Tipo de Acesso
1	Utilizador	READ
2	Jogo	READ

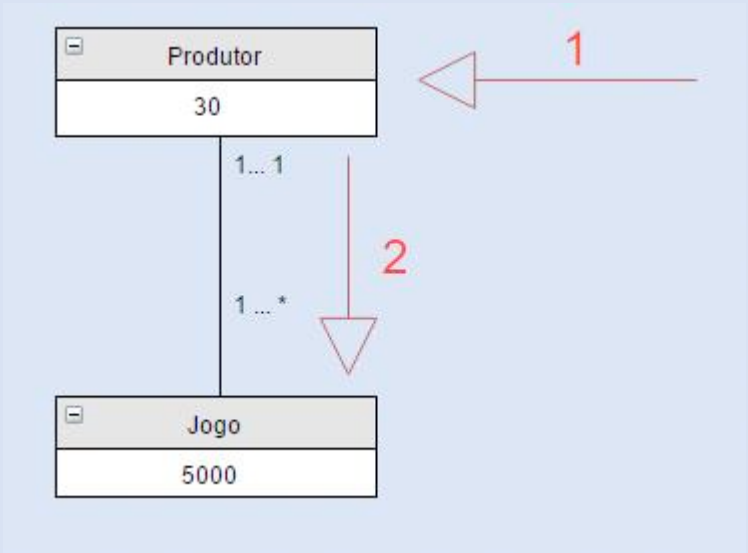
3	Compra	INSERT
4	Jogo	UPDATE

Analise de transações para a ***transação (B):***

Transação (B) Inserir um jogo e adicionar/associar um produtor;

Volume de transações	
Media:	**8** por Dia Pico de transações
Pico de transações:	**12** por Dia (principal pico de atividade das 20:00 ás 22:00 Durante a semana, e 14:00 ás 16:00 no fim-de-semana)

Mapa de Transações

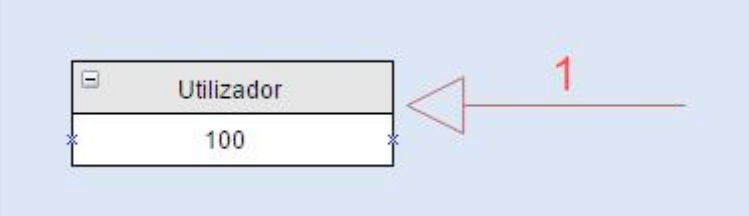


Acesso	**Entidade**	**Tipo de Acesso**
1	Produtor	READ, INSERT
2	Jogo	READ

Analise de transações para a ***transação (C):***

Transação	**(C)** Registar de um novo cliente;
Volume de transações	
Media:	**10** por Dia Pico de transações
Pico de transações:	**20** por Dia (principal pico de atividade das 20:00 ás 22:00 Durante a semana, e 14:00 ás 16:00 no fim-de-semana)

Mapa de Transações



Acesso	Entidade	Tipo de Acesso
1	Utilizador	READ, INSERT

5.3. Escolha dos Índices

Um índice é uma estrutura de dados que permite ao SGBD localizar registos num ficheiro mais rapidamente e, consequentemente, procurar melhorar o tempo de respostas a *queries* do utilizador.

Os SGBDs criam os índices para as chaves primárias por defeito, deixando para nós a decisão da criação de novos índices secundários. Estes índices secundários são relativos a pesquisas muito frequentes sobre certos atributos.

Posteriormente, observamos as relações estabelecidas e verificámos a existência de atributos ao qual vão ser frequentemente usados para pesquisas nas relações. Para esses atributos identificados, decidimos criar os seguintes índices para as relações:

- Cliente: para o cliente identificamos que os atributos `email` e `id_cidade` vão ser alvo de várias pesquisas, pelo que optámos por criar o seguinte índice:

```
CREATE INDEX fk_email_idx ON Cliente(email)
CREATE INDEX fk_id_cidade _idx ON Cliente(id_cidade)
```

- Cidade: para a cidade identificamos que o atributo `id_pais` vai ser alvo de várias pesquisas, pelo que optámos por criar o seguinte índice:

```
CREATE INDEX fk_id_pais_idx ON Cliente(id_pais)
```

- Jogo: para o jogo identificamos que os atributos `nome` e `id_produtor` vão ser alvo de várias pesquisas, pelo que optámos por criar o seguinte índice:

```
CREATE INDEX fk_nome_idx ON Jogo(nome)
CREATE INDEX fk_id_produtor_idx ON Jogo(id_produtor)
```

- Produtor: para o produtor identificamos que o atributo `nome` vai ser alvo de várias pesquisas, pelo que optámos por criar o seguinte índice:

```
CREATE INDEX fk_nome_idx ON Produtor(nome)
```

- Categoria: para a categoria identificamos que o atributo `nome` vai ser alvo de várias pesquisas, pelo que optámos por criar o seguinte índice:

```
CREATE INDEX fk_nome_idx ON Categoria(nome)
```

- Compra: para a compra identificamos que os atributos `id_cliente` e `id_jogo` vão ser alvos de várias pesquisas, pelo que optámos por criar os seguintes índices:

```
CREATE INDEX fk_id_cliente_idx ON Compra(id_cliente)
CREATE INDEX fk_id_jogo_idx ON Compra(id_jogo)
```

5.4. Estimativa dos Requisitos do Espaço em Disco

Para a estimativa dos requisitos de espaço em disco, baseamo-nos no tamanho dos tipos para o motor de armazenamento InnoDB do MySQL. Posto isto, precisamos de saber o espaço necessário para representar cada tipo de domínio utilizado na base de dados. O InnoDB utiliza o "COMPACT Row Format" por defeito, e os tamanhos para cada tipo são os seguintes:

- INT 4 bytes
- DECIMAL (7,2) 4 bytes
- DECIMAL(5,2) 3 bytes
- DATE 3 bytes
- VARCHAR(N) N+2 bytes
- CHAR(1) 1 bytes
- DATETIME 8 bytes
- TINYINT(1) 1 byte
- TEXT L+2 bytes, L<2^16 = 65535 bytes

Após alguns cálculos, concluímos que o espaço necessário para cada registo nas tabelas são os seguintes:

- Cliente 196 bytes
- Categoria 31 bytes
- Cidade 45 bytes
- Jogo 78 bytes

- Pais 41 bytes
- Produtor 56 bytes
- *Avaliação 65 552 bytes
- Compra 31 bytes
- JogoCategoria 8 bytes

*cálculo efetuado tendo em conta uso de todos os caracteres, ou seja, utilização do tamanho máximo possível do tipo TEXT.

Tendo em conta os cálculos feitos em cima para cada uma das tabelas, segue-se o calculo para cada registo numa tabela a multiplicar pelo número de registos totais:

- Cliente 196*100 = 196 00 bytes
- Categoria 31*10 = 310 bytes
- Cidade 45*20 = 900 bytes
- Jogo 78*50 = 3 900 bytes
- Pais 41*5 = 205 bytes
- Produtor 56*30 = 1 680 bytes
- *Avaliação 65 552*30 = 1 966 560 bytes
- Compra 31*250 = 7 750 bytes
- JogoCategoria 8*75 = 600 bytes

Somando os valores a cima representados estima-se que o tamanho inicial necessário ronde 2 001 505 *bytes*, ou seja, aproximadamente 2MB (*megabytes*).

5.5. Definição das Vistas dos Utilizadores e Regras de Acesso

Uma das questões mais importantes no desenvolvimento de uma base de dados é a restrição das vistas dos diferentes utilizadores. Foram definidos dois perfis de utilizadores da nossa base de dados:

- Administradores: tem acesso a toda a base de dados e pode executar todas as ações que entender, podendo também apresentar conclusões sobre os dados apresentados para depois apresentar ao gerente da loja;

- Cliente: tem praticamente acesso a toda a base de dados, desde aos dados dos utilizadores, dados descritivos dos produtos que incluem os produtores e as categorias. No entanto, estes não têm acesso ao stock dos jogos da empresa.

Relativamente aos mecanismos de segurança, é necessário definir para cada tipo de utilizador os seus determinados privilégios, fazendo com que estes apenas acedam ao estritamente necessário.

No caso do Administrador da base de dados, este deve possuir todos as permissões para adicionar, remover e modificar. Visto que este tem de administrar a base de dados, logo tem de ser capaz de executar qualquer das ações anteriormente descritas.

Já em relação ao Cliente, este poderá aceder a praticamente todos os dados do sistema operacional. Poderá aceder a certas informações sobre os utilizadores, aceder ao seu registo de compras e avaliações. No entanto, este não poderá aceder aos dados do stock atual da empresa, bem como proceder a alguma modificação ou remoção que não estejam relacionadas com a sua conta de cliente.

No anexo III encontram-se definidas algumas vistas para os utilizadores do sistema de base de dados.

6. Análise da Qualidade dos Dados

Ao longo deste capítulo vamos analisar a qualidade dos dados presentes no sistema de base de dados implementado anteriormente. Para tal, optámos por utilizar a ferramenta *Data Cleaner* que possui um vasto conjunto de funcionalidades que nos permitem analisar as restrições de integridade, regras de negócio, conteúdo dos dados, entre outros.

Inicialmente, optámos por analisar a integridade referencial de todas as chaves estrangeiras definidas, e pudemos constatar que o nosso modelo obedece a estas restrições. De seguida, apresentámos o resultado de um teste feito à chave estrangeira `id_jogo` na tabela Compra relativamente à chave primária da tabela Jogo:

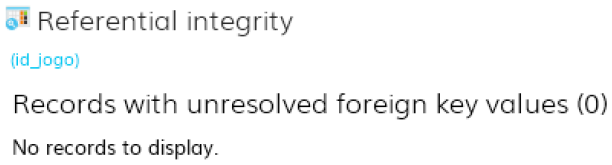


Figura 6 - Resultado do teste da restrição de integridade.

De seguida, optámos por analisar as regras de negócio definidas para os nossos atributos, entre as quais testámos a restrição de valor único, e concluímos que o nosso modelo obedece a estas restrições. De seguida, apresentámos o resultado do teste feito ao atributo `email` da tabela Cliente:

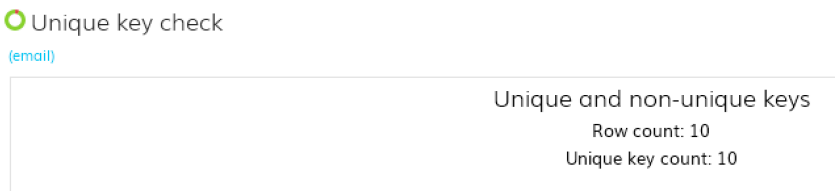


Figura 7 - Resultado do teste da restrição de valor único.

Por fim, analisámos ao pormenor o conteúdo das nossas tabelas, nomeadamente as tabelas Pais e Cidade, de forma a ver se encontrávamos possíveis representações diferentes

para um mesmo país e cidade, respetivamente. Os resultados destes testes foram os seguintes:

Value	COUNT(*)
Andorra	1
Espanha	1
Estados Unidos da América	1
Franca	1
Inglaterra	1
Portugal	1
Suica	1
USA	1

Value	COUNT(*)
Barcelona	1
Braga	1
Fafe	1
Lisboa	1
Lousada	1
Madrid	1
Paris	1
Porto	1
VNF	1
Vila Nova de Famalicão	1

Figura 8 - Resultado do teste da análise do conteúdo das tabelas Pais e Cidade.

Através da análise destes resultados, pudemos verificar que existem duas representações para um mesmo país (“Estados Unidos da América” e “USA”) e para uma mesma cidade (“Vila Nova de Famalicão” e “VNF”). Para este tipo de ocorrências, é necessário uniformizar os dados para prevenir possíveis futuros erros.

7. Ferramentas Utilizadas

Para a elaboração deste documento foram utilizadas várias ferramentas:

- Microsoft Word 2016: elaboração deste documento;
- brModelo: desenho do modelo conceptual;
- VisualParadigm: desenho dos modelos ER;
- MySQL Workbench: desenvolvimento do modelo de dados lógico e físico;
- Draw.io: desenho dos mapas de transações;
- DataCleaner: análise da qualidade dos dados.

8. Conclusões e Trabalho Futuro

O desenvolvimento de uma base de dados é um processo longo e complicado, sendo que necessita de ser feito de forma correta, caso contrário pode provocar elevados custos a uma organização. De forma a prevenir futuros erros de implementação, seguimos uma metodologia de forma a sistematizar todo o processo de desenvolvimento do sistema de base de dados, que se revelou fulcral para o sucesso da implementação do sistema.

Depois de aplicada toda a metodologia, e já com o modelo implementado, pudemos constatar que o processo terminou com sucesso, estando esta implementada de forma a cumprir todos os requisitos imposto pela loja online. No entanto, achámos que o sistema de base de dados poderia ser melhorado de forma a conter outro tipo de funcionalidades, como vendas de *bundles* de jogos ou um melhor serviço de gestão de stock, mas que não são necessárias para o momento, mas que poderão vir a ser implementadas num futuro próximo.

Por fim, analisámos a qualidade dos dados que o sistema de base de dados contém, ao qual encontrámos anomalias que devem ser tratadas de forma correta. Estes erros podem ser preocupantes se não forem detetados e devidamente corrigidos, isto porque podem influenciar futuras análises de dados de forma negativa, que podem também acabar por afetar negativamente o negócio da loja online.

Bibliografia

M.Connolly, T. & E.Begg, C., 2015. *Database Systems A Pratical Approach to Design, Implementation, and Management.* 6th Edition ed. England: Person Education Limited.

Lista de Siglas e Acrónimos

BD Base de Dados

SO Sistema Operacional

BI *Business Intelligence*

DBDL Database Design Language

SGBD Sistema de Gestão de Base de Dados

1NF 1ª Forma Normal

2NF 2ª Forma Normal

3NF 3ª Forma Normal

Anexos

I. *Script* completo da criação do esquema físico

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET                        @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET                                              @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema sitexxi
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema sitexxi
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `sitexxi` DEFAULT CHARACTER SET utf8 ;
USE `sitexxi` ;

-- -----------------------------------------------------
-- Table `sitexxi`.`Pais`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sitexxi`.`Pais` (
  `id_pais` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(35) NOT NULL,
  PRIMARY KEY (`id_pais`),
  UNIQUE INDEX `nome_UNIQUE` (`nome` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `sitexxi`.`Cidade`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sitexxi`.`Cidade` (
```

```
  `id_cidade` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NOT NULL,
  `id_pais` INT NOT NULL,
  PRIMARY KEY (`id_cidade`),
  INDEX `fk_id_pais_idx` (`id_pais` ASC),
  UNIQUE INDEX `nome_UNIQUE` (`nome` ASC),
  CONSTRAINT `fk_Cidade_Pais`
    FOREIGN KEY (`id_pais`)
    REFERENCES `sitexxi`.`Pais` (`id_pais`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `sitexxi`.`Cliente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sitexxi`.`Cliente` (
  `id_cliente` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(100) NOT NULL,
  `email` VARCHAR(50) NOT NULL,
  `data_nascimento` DATE NOT NULL,
  `sexo` CHAR(1) NOT NULL,
  `nr_telemovel` VARCHAR(20) NULL,
  `data_registo` DATETIME NOT NULL,
  `id_cidade` INT NOT NULL,
  PRIMARY KEY (`id_cliente`),
  INDEX `fk_id_cidade_idx` (`id_cidade` ASC),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC),
  INDEX `fk_email_idx` (`email` ASC),
  CONSTRAINT `fk_Cliente_Cidade1`
    FOREIGN KEY (`id_cidade`)
    REFERENCES `sitexxi`.`Cidade` (`id_cidade`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `sitexxi`.`Produtor`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `sitexxi`.`Produtor` (
  `id_produtor` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`id_produtor`),
  UNIQUE INDEX `nome_UNIQUE` (`nome` ASC),
  INDEX `fk_nome_idx` (`nome` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `sitexxi`.`Jogo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sitexxi`.`Jogo` (
  `id_jogo` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(50) NOT NULL,
  `data_lancamento` DATE NOT NULL,
  `idade_minima` INT UNSIGNED NOT NULL,
  `stock` INT UNSIGNED NOT NULL,
  `preco` DECIMAL(7,2) UNSIGNED NOT NULL,
  `desconto` DECIMAL(5,2) UNSIGNED NOT NULL,
  `id_produtor` INT NOT NULL,
  PRIMARY KEY (`id_jogo`),
  UNIQUE INDEX `nome_UNIQUE` (`nome` ASC),
  INDEX `fk_id_produtor_idx` (`id_produtor` ASC),
  INDEX `fk_id_nome_idx` (`nome` ASC),
  CONSTRAINT `fk_Jogo_Produtor1`
    FOREIGN KEY (`id_produtor`)
    REFERENCES `sitexxi`.`Produtor` (`id_produtor`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `sitexxi`.`Categoria`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sitexxi`.`Categoria` (
  `id_categoria` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`id_categoria`),
  UNIQUE INDEX `nome_UNIQUE` (`nome` ASC),
```

```
  INDEX `fk_nome_idx` (`nome` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `sitexxi`.`Avaliacao`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sitexxi`.`Avaliacao` (
  `id_cliente` INT NOT NULL,
  `id_jogo` INT NOT NULL,
  `avaliacao` TINYINT(1) NOT NULL,
  `descricao` TEXT NULL,
  `data_avaliacao` DATETIME NOT NULL,
  PRIMARY KEY (`id_cliente`, `id_jogo`),
  INDEX `fk_id_jogo_idx` (`id_jogo` ASC),
  INDEX `fk_id_cliente_idx` (`id_cliente` ASC),
  CONSTRAINT `fk_Cliente_has_Jogo_Cliente1`
    FOREIGN KEY (`id_cliente`)
    REFERENCES `sitexxi`.`Cliente` (`id_cliente`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Cliente_has_Jogo_Jogo1`
    FOREIGN KEY (`id_jogo`)
    REFERENCES `sitexxi`.`Jogo` (`id_jogo`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `sitexxi`.`Compra`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sitexxi`.`Compra` (
  `id_compra` INT NOT NULL AUTO_INCREMENT,
  `id_cliente` INT NOT NULL,
  `id_jogo` INT NOT NULL,
  `quantidade` INT UNSIGNED NOT NULL,
  `preco` DECIMAL(7,2) UNSIGNED NOT NULL,
  `desconto` DECIMAL(5,2) UNSIGNED NOT NULL,
  `data_compra` DATETIME NOT NULL,
  PRIMARY KEY (`id_compra`),
```

```
  INDEX `fk_id_jogo_idx` (`id_jogo` ASC),
  INDEX `fk_id_cliente_idx` (`id_cliente` ASC),
  CONSTRAINT `fk_Cliente_has_Jogo_Cliente2`
    FOREIGN KEY (`id_cliente`)
    REFERENCES `sitexxi`.`Cliente` (`id_cliente`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Cliente_has_Jogo_Jogo2`
    FOREIGN KEY (`id_jogo`)
    REFERENCES `sitexxi`.`Jogo` (`id_jogo`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `sitexxi`.`JogoCategoria`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sitexxi`.`JogoCategoria` (
  `id_jogo` INT NOT NULL,
  `id_categoria` INT NOT NULL,
  PRIMARY KEY (`id_jogo`, `id_categoria`),
  INDEX `fk_id_categoria_idx` (`id_categoria` ASC),
  INDEX `fk_id_jogo_idx` (`id_jogo` ASC),
  CONSTRAINT `fk_Jogo_has_Categoria_Jogo1`
    FOREIGN KEY (`id_jogo`)
    REFERENCES `sitexxi`.`Jogo` (`id_jogo`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Jogo_has_Categoria_Categoria1`
    FOREIGN KEY (`id_categoria`)
    REFERENCES `sitexxi`.`Categoria` (`id_categoria`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

II. *Script* completo do povoamento do sistema operacional “sitexxi”

```
USE sitexxi;

INSERT INTO `sitexxi`.`Pais`
(`nome`)
VALUES
('Portugal'),
('Franca'),
('Inglaterra'),
('Espanha'),
('Suica'),
('Andorra');

INSERT INTO `sitexxi`.`Cidade`
(`nome`,`id_pais`)
VALUES
('Braga',1),
('Lisboa',1),
('Porto',1),
('Fafe',1),
('Paris',2),
('Barcelona',4),
('Madrid',4),
('Lousada',1);

INSERT INTO `sitexxi`.`Cliente`
(`nome`,`email`,`data_nascimento`,`sexo`,`nr_telemovel`,`data_registo`
,`id_cidade`)
VALUES
('Gil  Goncalves','gidl@mail.com','1992-09-1','M','123456709','2011-04-
2',4),
```

```
('Jose    Pedro','jose@mail.com','1993-03-12','M','234567890','2012-01-
2',9),
('Bruno   Ribeiro','bruno@mail.com','1991-01-14','M','345678901','2012-
05-2',1),
('Luis    Pedro','luis@mail.com','1990-04-15','M','456789012','2009-04-
15',1),
('Celia                          Figueiredo','celia@mail.com','1993-12-
24','F','567890123','2012-01-2',2),
('Marcia       Costa','marcia@mail.com','1992-02-12','F',NULL,'2011-12-
2',3),
('Daniel    Rodrigues','daniel@mail.com','1990-09-1','M',NULL,'2010-04-
2',4),
('Ricardo  Lopes','ricardo@mail.com','1990-09-1','M','678901234','2009-
04-2',5),
('Carlos Faria','carlos@mail.com','1993-12-1','M',NULL,'2012-11-2',6);

INSERT INTO `sitexxi`.`Produtor`
(`nome`)
VALUES
('Capcom'),
('Square Enix'),
('Konami'),
('Ubisoft'),
('SEGA'),
('Activision Blizzard'),
('Nintendo'),
('Sony'),
('Microsoft'),
('Electronic Arts');

INSERT INTO `sitexxi`.`Categoria`
(`nome`)
VALUES
('Acao'),
('Aventura'),
('Estrategia'),
('RPG'),
('Desporto'),
('Corrida'),
('Quebra-Cabeca'),
('Simulacao');
```

```
INSERT INTO `sitexxi`.`Jogo`
(`nome`,`data_lancamento`,`idade_minima`,`stock`,`preco`,`desconto`,`i
d_produtor`)
VALUES
('GTA','2014-02-12',18,100,60,5,4),
('Max Payne ','2014-01-12',18,100,65,0,10),
('Hitman','2013-02-12',18,100,50,80,1),
('Gears of War','2014-02-12',18,10,30,10,10),
('Monkey Island','2010-02-12',12,50,20,20,2),
('Tomb Raider','2015-02-12',18,30,30,60,3),
('NASCAR','2014-02-12',18,100,15.3,54,4),
('GTR','2015-10-12',18,10,13,0,5),
('Need For Speed','2013-12-12',18,60,30,28,6),
('Alone in The Dark','2014-02-12',18,100,30,39,7);

INSERT INTO `sitexxi`.`JogoCategoria`
(`id_jogo`,`id_categoria`)
VALUES
(1,1),
(1,2),
(2,3),
(2,2),
(3,4),
(3,1),
(4,4),
(4,1),
(5,2),
(5,1),
(6,3),
(6,4),
(10,1);

INSERT INTO `sitexxi`.`Compra`
(`id_cliente`,`id_jogo`,`quantidade`,`preco`,`desconto`,`data_compra`)
VALUES
(1,1,1,30,50,'2014-05-10'),
(2,1,3,39,40,'2014-06-12'),
(2,2,1,65,0,'2014-01-12'),
(3,2,1,25,50,'2013-05-12'),
(1,3,1,60,0,'2014-02-12'),
```

```
(4,3,1,15,50,'2014-06-01'),
(4,4,3,30,0,'2014-02-13'),
(1,4,1,58,20,'2014-03-01'),
(1,5,1,12,8,'2015-03-12'),
(5,5,1,10,50,'2013-05-12'),
(6,6,3,30,0,'2015-02-12'),
(2,6,4,19.5,0,'2014-06-17'),
(7,7,1,15.3,0,'2014-02-12'),
(7,5,1,4,0,'2014-03-12'),
(8,8,2,13,0,'2015-10-12'),
(8,10,3,30,0,'2014-02-12'),
(9,9,1,15,0,'2013-12-25');

INSERT INTO `sitexxi`.`Avaliacao`
(`id_cliente`,`id_jogo`,`avaliacao`,`descricao`,`data_avaliacao`)
VALUES
(1,1,3,'Gostei muito','2015-05-10'),
(1,2,5,'Melhor jogo de sempre','2015-04-11'),
(2,3,1,'Adorei','2015-01-15'),
(8,8,5,NULL,'2015-07-25'),
(9,9,3,NULL,'2015-09-18'),
(8,10,1,NULL,'2015-10-19'),
(3,4,5,NULL,'2015-05-23'),
(4,1,1,'Recomendo','2015-05-28');
```

III. Definição de algumas vistas dos utilizadores

Vista dos clientes sobre os rstantes clientes

```
CREATE VIEW verPerfilCliente AS
       SELECT nome  AS 'Nome do Cliente',
              Email  AS 'Email do Cliente',
              sexo   AS 'Sexo do Cliente'
       FROM Cliente
```

Vista dos clientes sobre os jogos

```
CREATE VIEW verJogo AS
        SELECT  J.nome AS 'Nome do Jogo',
                P.nome AS 'Nome do Produtor do Jogo',
                J.preco AS 'Preço do Jogo',
                J.desconto AS 'Desconto do Jogo',
                J.data_lancamento AS 'Data de Lançamento',
                J.idade_minima AS 'Idade permitida para jogar',
                J.stock AS 'Quantidade disponível do Jogo',
                AVG(A.avaliacao) AS 'Avaliação do Jogo'
       FROM Jogo AS J  INNER JOIN Produtor AS P
                ON J.id_produtor=P.id_produtor
                INNER JOIN Avaliacao AS A ON A.id_jogo=J.id_jogo
                        GROUP BY A.id_jogo
                        ORDER BY AVG(A.avaliacao) DESC;
```

Vista do gestor sobre as vendas de cada jogo

```
CREATE VIEW verCompras AS
SELECT  J.Nome AS 'Nome do Jogo',
        sum(C.preco*C.quantidade) AS 'Dinheiro Feito com o Jogo'
       FROM Jogo AS J INNER JOIN Compra AS C on J.id_jogo= C.id_jogo
       GROUP BY J.id_jogo
       ORDER BY sum(C.preco*C.quantidade) DESC;
```

Vista do gestor sobre as informações do cliente

```
CREATE VIEW infoCliente AS
SELECT  Nome AS 'Nome',
        FLOOR(DATEDIFF(DATE(NOW()),data_nascimento)/365) as 'Idade',
        email AS 'Email',
        nr_telemovel AS 'Nr Telemóvel',
        data_registo AS 'Data de Registo',
        cidade AS 'Cidade',
        nome AS 'Pais'
        FROM Cliente AS C INNER JOIM Cidade AS Ci
                ON C.id_cidade=Ci.id_cidade
                INNER JOIN Pais AS P on P.id_pais=Ci.id_pais;
```