TECNOLOGIA SETÚBAL
ESCOLA SUPERIOR POLITÉCNICO SETÚBAL

Complementos de Bases de Dados

2025/2026

Licenciatura em Engª. Informática

1ª Fase Relatório Técnico

Turma: 4
Horário de Laboratório: 10h30/12h30
Docente: Cláudio Sapateiro
Grupo: 3
Nº202300149, Diogo Sabino
Nº2024151048, Rodrigo Antunes

1. Introdução

Este projeto consiste em melhorar uma base de dados antiga e desatualizada. Recebemos ficheiros em formato "csv" contendo os dados de uma base de dados antiga relativa a uma empresa chamada "AdventureWorks" e baseando-nos nesses dados temos de propor um novo modelo relacional para uma nova base de dados que deverá ser normalizada, otimizada e capaz de carregar todos os dados da base de dados antiga.

2. Especificação de Requisitos

ID	Descrição	Implementado (S/N)
R01	Um Customer tem sempre um e apenas um Gender.	S
R02	Um Gender tem zero ou mais Customers.	S
R03	Um Customer tem sempre uma e apenas uma Occupation.	S
R04	Uma Occupation tem zero ou mais Customers.	S
R05	Um Customer tem sempre uma e apenas uma Education.	S
R06	Uma Education tem zero ou mais Customers.	S
R07	Um Customer tem sempre uma e apenas uma Address.	S
R08	Uma Address tem um ou mais Customers.	S
R09	Um Customer tem zero ou mais SalesOrder.	S
R10	Uma SalesOrder tem sempre um e apenas um Customer.	S
R11	Um Customer sempre um e apenas um User.	S
R12	Um User tem sempre um e apenas um Customer.	S

R13	Um User recebe zero ou mais Emails.	S
R14	Um Email é enviado para um e apenas um User	S
R15	Um StateProvince tem sempre uma e apenas uma Region.	S
R16	Uma Region tem um ou mais StateProvince.	S
R16	Uma Region tem sempre um e apenas um Country.	S
R17	Um Country tem uma ou mais Region.	S
R18	Um Country tem sempre um e apenas um Group.	S
R19	Um Group tem um ou mais Country.	S
R20	Uma SalesOrder tem sempre um ou mais SalesOrderLine.	S
R21	Uma SalesOrderLine tem uma e apenas uma SalesOrder.	S
R22	Uma SalesOrderLine tem sempre um e apenas um Products.	S
R23	Um Products tem zero ou mais SalesOrderLine.	S
R24	Um Products tem zero ou uma Color.	S
R25	Uma Color zero um ou mais Products.	S
R26	Um Products tem sempre uma e apenas uma Category.	S
R27	Uma Category tem zero ou mais Products.	*S*
R28	Um Product tem zero ou uma color.	*S*
R29	Uma color tem zero ou mais produtos.	*S*

R30	O sistema deve permitir adicionar um novo Customer.	*S*
R31	O sistema deve permitir remover um Customer.	*S*
R32	O sistema deve permitir editar um Customer.	*S*
R33	O sistema deve permitir adicionar uma nova subcategoria	*S*
R34	O sistema deve permitir apagar uma categoria, apagando todas as suas subcategorias	*S*
R35	O sistema deve poder devolver um endereço completo	*S*
R36	O sistema deve poder devolver o nome completo com título de um Customer	*S*
R37	O sistema deve poder devolver o valor total de vendas em um dia específico	*S*

3. Modelo Relacional (*Modelo de dados*)

Customer (customerID, firstName, middleName, lastName, birthDate, yearlyIncome, numbersCarsOwned, dateFirstPurchase, title, martialStatus, genderID, occupationID, educationID, addressID, stateProvinceID, userEmail)

- PK (customerID)

- FK (genderID) -> Gender

- FK (occupationID) -> Occupation

- FK (educationID) -> Education

- FK (addressID) -> Address

- FK (userEmail) -> UserSecurity

UserSecurity (userEmail, password, phone, securityQuestion, securityAnswer)

- PK (userEmail)

SentEmails (sentEmailsID, message, timestamp, userEmail)

- PK (sentEmailsID)

- FK (userEmail)

Address (addressID, addressLine1, postalCode, city)

- PK (addressID)

- FK (cityID) -> City

Education (educationID, educationName, customerID)

- PK (educationID)

- FK (customerID) -> Customer

Occupation (occupationID, occupationName, customerID)

- PK (occupationID)

- FK (customerID) -> Customer

Gender (genderID, genderName, customerID)

- PK (genderID)

- FK (customerID) -> Customer

City (cityID, cityName, stateProvinceID)

- PK (cityID)

- FK (stateProvinceID)

StateProvince (stateProvinceID, stateProvinceName, regionID)

- PK (stateProvinceID)

- FK (regionID) -> Region

Region (regionID, regionName, countryID)

- PK (regionID)

- FK (countryID) -> Country

Country (countryID, countryName, groupID)

- PK (countryID)

- FK (groupID) -> Group

Group (groupID, groupName)

- PK (groupID)

SalesOrder (salesOrderID, orderDate, dueDate, shipDate, freight, taxAmt, totalSalesAmt, customerID, currencyID)

- PK (salesOrderID)

- FK (customerID) -> Customer

- FK (currencyID) -> Currency

SalesOrderLine (salesOrderLineNumber, unitPrice, salesOrderID, productID)

- PK (salesOrderID, salesOrderLineNumber)

- FK (salesOrderNumber) -> SalesOrder

- FK (productID) -> Product

Currency (currencyID, currencyName, curencyAlternateKey)

- PK (currencyID)

Products (productID, productName, modelName, sizeRange, description, class, style, weight, weightUnitMeasureCode, standardCost, listPrice, dealerPrice, safetyStockLevel, daysToManufacture, finishedGoodFlag, productLine, colorID, categoryID)

- PK (productID)

- FK (colorID) -> Color

- FK (categoryID) -> Category

Category (categoryID, categoryName, parentCategoryID)

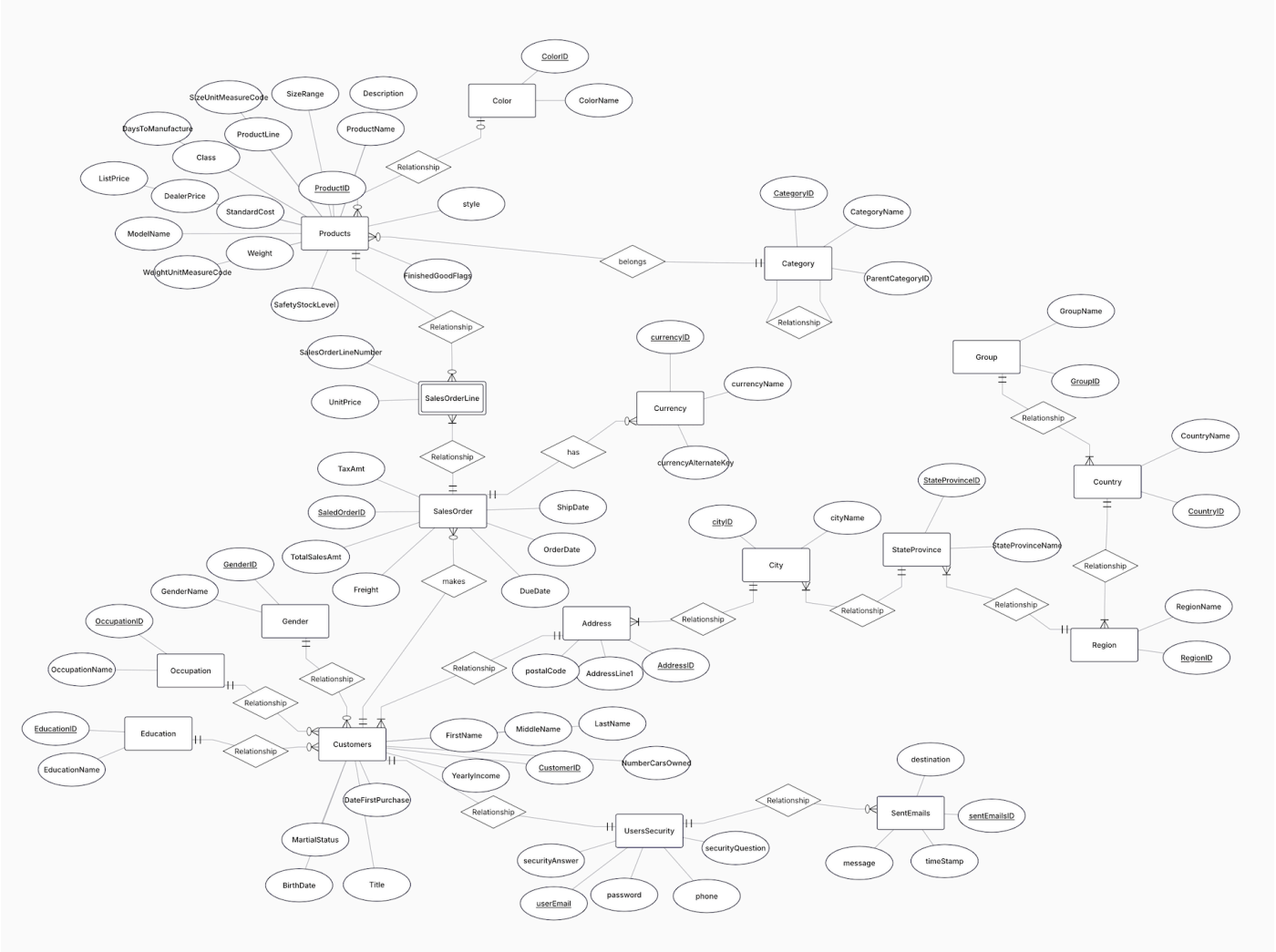
- PK (categoryID)

- FK (parentCategory) -> Category

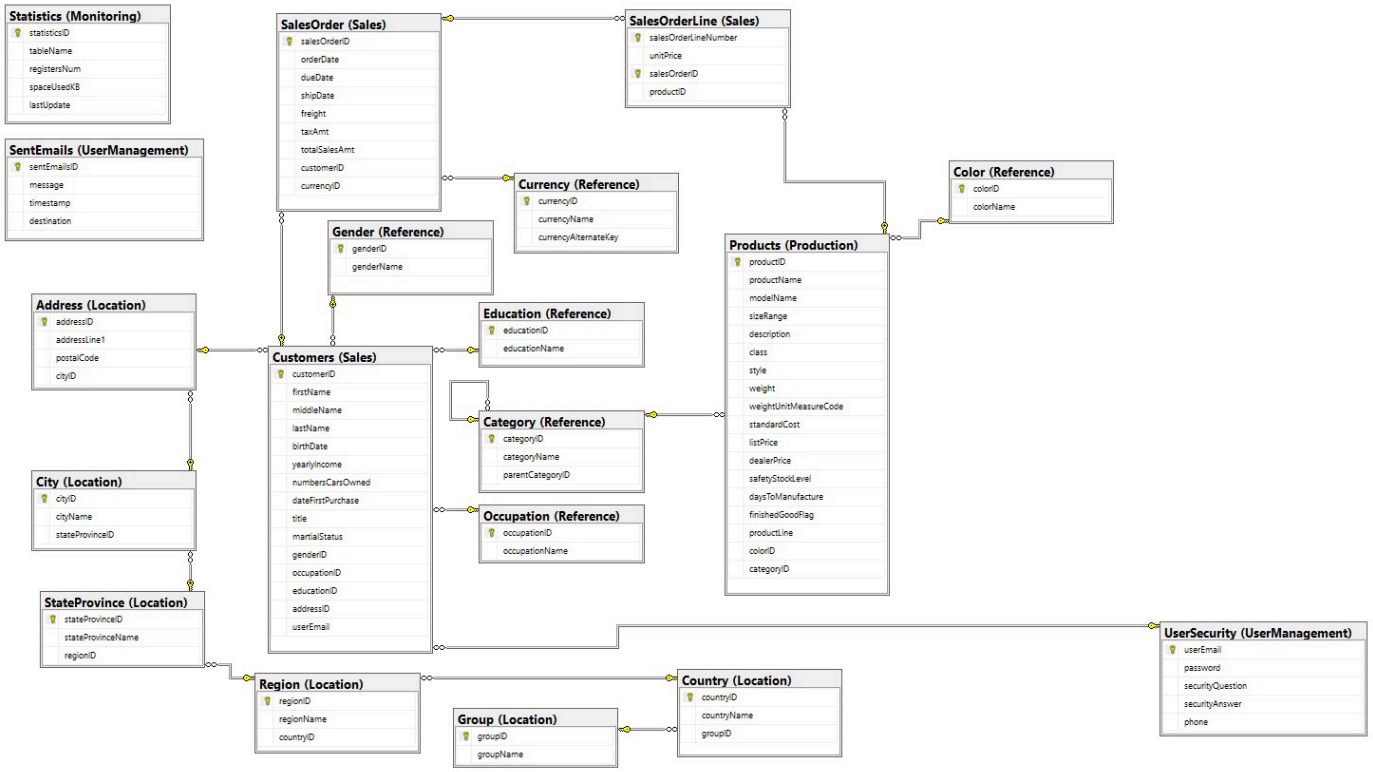
Color (colorID, colorName)

- PK (colorID)

3.1 Diagrama do Modelo Entidade Relação



3.2 Diagrama do Modelo Relacional



4. Definição do Layout

4.1 Identificação do espaço ocupado por tabela

Nome Tabela	Dimensão do Registo	Nº de Registos (inicial/final)
SentEmails	371 bytes	0
Statistics	276 bytes	0
Gender	63 bytes	2
Group	53 bytes	3
Occupation	53 bytes	5
Education	53 bytes	5
Country	57 bytes	6
Color	23 bytes	9
Region	60 bytes	10
Category	60 bytes	41
StateProvince	60 bytes	53
Currency	59 bytes	105
City	60 bytes	285
Products	686 bytes	387

Customers	471 bytes	18484
UserSecurity	740 bytes	18484
Address	72 bytes	20068
SalesOrder	40 bytes	27659
SalesOrderLine	14 bytes	60389

4.2 Especificação dos FIlegroups

Nome Filegroup	Tabelas associadas	Parâmetros
PRIMARY	Group, Country, Region, StateProvince, Occupation, Education, Gender, Category, Color, Currency	SIZE = 2MB MAXSIZE = 20MB FILEGROWTH = 1MB
AdventureSecondaryFG	City, Address, UserSecurity, Products, Customers, SalesOrder, SalesOrderLine	SIZE = 10MB MAXSIZE = 100MB FILEGROWTH = 10MB
LogFileGroup	SentEmails, Statistics	SIZE = 15MB MAXSIZE = 500MB FILEGROWTH = 50MB

4.3 Schemas

Nome	Descrição
Location	Contém toda a informação geográfica.
UserManagement	Gera e controla dados de utilizadores e segurança.
Reference	Guarda tabelas de referência e listas de valores fixos.
Production	Armazena os dados de produtos, características técnicas e preços.
Sales	Regista os clientes e todas as encomendas.
Monitoring	Guarda estatísticas e métricas sobre as tabelas e o uso da base de dados.

5. Verificação da migração de dados

5.1 Consultas sobre a base de dados original

Consulta específica de Categorias e Subcategorias

```
SELECT COUNT(DISTINCT EnglishProductCategoryName) AS OldCategoryCount
FROM AdventureWorksLegacy.dbo.Products;
SELECT COUNT(*) AS OldSubcategoryCount
FROM AdventureWorksLegacy.dbo.ProductSubCategory;
```

Customers

```
SELECT COUNT(*) AS OldCount FROM AdventureWorksLegacy.dbo.Customer;
```

Gender

```
SELECT COUNT(*) AS OldGenderCount FROM AdventureWorksLegacy.dbo.Customer WHERE Gender IS NOT NULL;
```

Education

```
SELECT COUNT(DISTINCT Education) AS OldEducationCount FROM AdventureWorksLegacy.dbo.Customer WHERE
Education IS NOT NULL;
```

Occupation

```
SELECT COUNT(DISTINCT Occupation) AS OldOccupationCount FROM AdventureWorksLegacy.dbo.Customer WHERE
Occupation IS NOT NULL;
```

Address

```
SELECT COUNT(DISTINCT AddressLine1) AS OldAddressCount FROM AdventureWorksLegacy.dbo.Customer;
```

Products

```
SELECT COUNT(SELECT COUNT(DISTINCT AddressLine1) AS OldAddressCount FROM
AdventureWorksLegacy.dbo.Customer;
```

Color

```
SELECT COUNT(DISTINCT Color) AS OldColorCount FROM AdventureWorksLegacy.dbo.Products WHERE Color IS NOT
NULL;
```

Currency

```
SELECT COUNT(*) AS OldCount FROM AdventureWorksLegacy.dbo.Currency;
```

Groups

```
SELECT COUNT(DISTINCT SalesTerritoryGroup) AS OldGroupCount
FROM AdventureWorksLegacy.dbo.SalesTerritory
WHERE SalesTerritoryGroup IS NOT NULL;
```

Countries

```
SELECT DISTINCT SalesTerritoryCountry FROM AdventureWorksLegacy.dbo.SalesTerritory
EXCEPT
SELECT countryName FROM AdventureWorks.dbo.Country;
```

Regions

```
SELECT COUNT(DISTINCT SalesTerritoryRegion) AS OldRegionCount
FROM AdventureWorksLegacy.dbo.SalesTerritory
WHERE SalesTerritoryRegion IS NOT NULL;
```

Sales

```
SELECT 'Sales (Legacy)' AS TableName, COUNT(DISTINCT SalesOrderNumber) AS LegacyHeaderCount
FROM AdventureWorksLegacy.dbo.Sales
UNION ALL
SELECT 'Sales (Legacy)' AS TableName, COUNT(*) AS LegacyDetailCount
FROM AdventureWorksLegacy.dbo.Sales;
```

5.2 Consultas sobre a nova base de dados

Consulta específica de Categorias e Subcategorias

```
SELECT COUNT(*) AS NewCategoryCount
FROM AdventureWorks.Reference.Category
WHERE parentCategoryID IS NULL;
SELECT COUNT(*) AS NewSubcategoryCount
FROM AdventureWorks.dbo.Category
WHERE parentCategoryID IS NOT NULL;
```

Customers

```
SELECT COUNT(*) AS NewCount FROM AdventureWorks.Sales.Customers;
```

Gender

```
SELECT COUNT(*) AS NewGenderCount FROM AdventureWorks.Reference.Gender;
```

Education

```
SELECT COUNT(*) AS NewEducationCount FROM AdventureWorks.Reference.Education;
```

Occupation

```
SELECT COUNT(*) AS NewOccupationCount FROM AdventureWorks.Occupation.Occupation;
```

Address

```
SELECT COUNT(*) AS NewAddressCount FROM AdventureWorks.Address.Address;
```

Products

```
SELECT COUNT(*) AS NewCount FROM AdventureWorks.Production.Products;
```

Color

```
SELECT COUNT(*) AS NewColorCount FROM AdventureWorks.Reference.Color;
```

Currency

```
SELECT COUNT(*) AS NewCount FROM AdventureWorks.Reference.Currency;
```

Groups

```
SELECT COUNT(*) AS NewGroupCount
FROM AdventureWorks.Location.[Group];
```

Countries

```
SELECT COUNT(*) AS NewCountryCount
FROM AdventureWorks.Location.Country;
```

Regions

```
SELECT COUNT(*) AS NewRegionCount FROM AdventureWorks.Location.Region;
```

Sales

```
SELECT 'SalesOrderHeader (New)' AS TableName, COUNT(*) AS NewRowCount FROM Sales.SalesOrder
UNION ALL
SELECT 'SalesOrderDetail (New)' AS TableName, COUNT(*) AS NewRowCount FROM Sales.SalesOrderLine
```

6. Programação

6.1 Views

Nome	Descrição
Location.vFullAddress	Permite obter uma lista de todas as moradas, incluindo a localidade.
UserManagement.vUserProfile	Permite obter uma lista com todas as informações de um User.
Sales.vOrderDetails	Permite obter uma lista com todos os detalhes de uma venda.
Sales.vCustomerOrders	Permite obter uma lista de todas as encomendas de um Customer.
Sales.vProductSalesSummary	Permite obter uma lista de todas as vendas por produto.

6.2 Functions

Nome	Atributos	Requisito	Descrição
Location.fnGetFullAddress(@addressID INT)	@addressID INT	R35	Devolve o endereço completo em uma única string
Sales.fnGetCustomerFullName(@customerID INT)	@customerID INT	R36	Devolve o nome completo do cliente, incluindo título
Sales.fnGetTotalSalesByDate(@salesDate DATE)	@salesDate DATE	R37	Devolve o valor total das vendas realizadas em um dia específico

6.3 Stored procedures

Nome	Atributos	Requisito	Descrição
Reference.spAddSubcategory	@SubcategoryName VARCHAR(50) @CategoryName VARCHAR (50)	R33	Permite adicionar uma nova subcategoria
Sales.spAddCustomer	@FirstName VARCHAR (50) @MiddleName VARCHAR (50) @LastName VARCHAR (50) @CustomerEmail VARCHAR (255) @CustomerPassword VARCHAR (255) @SecurityQuestion VARCHAR (100) @SecurityAnswer VARCHAR (100) @Phone VARCHAR (20) @Address VARCHAR (50) @PostalCode VARCHAR (10) @MartialStatus VARCHAR (15) @BirthDate DATE @Education VARCHAR (50) @Occupation VARCHAR (50)	R30	Permite adicionar um novo utilizador

	@Gender VARCHAR (50)		
Sales.spDeleteCustomer	@CustomerEmail VARCHAR(255)	R31	Permite remover um customer
Monitoring.spdbStatistics	nenhum atributo	R30	Permite obter as estatísticas sobre todas as tabelas
Sales.spEditFirstName	@ID INT @first VARCHAR (50)	R32	Permite editar o primeiro nome de um Customer
Sales.spEditMiddleName	@ID INT @middle VARCHAR (50)	R32	Permite editar o nome do meio de um Customer
Sales.spEditLastName	@ID INT @last VARCHAR (50)	R32	Permite editar o último nome de um Customer
Sales.spEditYearlyIncome	@ID INT @income DECIMAL (10,2)	R32	Permite editar o salário anual de um Customer
Sales.spEditNumberCars	@ID INT @num TINYINT	R32	Permite editar o número de carros de um Customer
Sales.spEditTitle	@ID INT @title VARCHAR (5)	R32	Permite editar o título de um Customer
Sales.spEditMartialStatus	@ID INT @stat VARCHAR(15)	R32	Permite editar o estado civíl de um Customer

Sales.spEditGender	@ID INT @gender TINYINT	R32	Permite editar o género de um Customer
Sales.spEditOccupation	@ID INT @occupation INT	R32	Permite editar a ocupação de um Customer
Sales.spEditEducation	@ID INT @education TINYINT	R32	Permite editar a educação de um Customer
Sales.spEditAddress	@ID INT @AddressID INT	R32	Permite editar a morada de um Customer
Sales.spEditEmail	@ID INT @email VARCHAR (255)	R32	Permite editar o email de um Customer

6.4 Triggers

Nome	Tipo	Tabela	Requisito	Descrição
trDeleteSubcategories	ISTED OF DELETE	Referance.Category	R34	Apaga todas as subcategorias

7. Catálogo/Metadados

7.1 Monitorização

Nome	Atributos	Descrição
Statistics	statisticsID INT IDENTITY (1,1) PRIMARY KEY, tableName SYSNAME, registersNum INT, spaceUsedKB DECIMAL (18,2), lastUpdate DATE DEFAULT GETDATE()	Informação sobre todas as tabelas da base de dados

8. Índices

Para a query pesquisa de vendas por cidade, implementámos três índices:

- Índice IX_SalesOrder_CustomerID_Covering (nonclustered index) que contém o customerID e o totalSalesAmt, para acelerar o cálculo das somas;
- Índice IX_Customers_AddressID (nonclustered index) que contém o addressID, para facilitar o join com a tabela address;
- Índice IX_City_StateProvinceID (nonclustered index) que contém stateProvinceID, para facilitar o join com a tabela SateProvince.

Para a query pesquisa de produtos associados a vendas com valor total monetário superior a 1000 implementámos dois índices:

- Índice IX_SalesOrder_OrderDate (nonclustered index) que contém a orderDate e a totalSalesAmt;
- Índice IX_SalesOrderLine_ProductID (nonclustered index) que contém o productID.

Para a query número de produtos vendidos por categoria por ano implementámos um índice:

- Índice IX_SalesOrderLine_ProductID (nonclustered index) que contém productID.

SQL Profiler e Tunning Advisor

Contraste entre o plano de execução das pesquisas na base de dados antiga e na base de dados nova normalizada e otimizada com o recurso a índices.

Pesquisa de vendas por cidade

AdventureWorksLegacy:

Rows	Executes	StmtText
285	1	SELECT ↵ distinct c.StateProvinceName,↵ c.City,↵ SUM(s.To...
0	0	|--Compute Scalar(DEFINE:([Expr1004]=CASE WHEN [Expr1009]=(...
285	1	|--Hash Match(Aggregate, HASH:([c].[City], [c].[StateProvince...
60398	1	|--Hash Match(Inner Join, HASH:([c].[CustomerKey])=([s].[C...
18484	1	|--Table Scan(OBJECT:([AdventureWorksLegacy].[dbo].[C...
60398	1	|--Table Scan(OBJECT:([AdventureWorksLegacy].[dbo].[S...

AdventureWorks (Sem índices):

275	1	SELECT ↵ ci.cityName AS 'City', ↵ sp.stateProvinceName AS 'S...
0	0	|--Compute Scalar(DEFINE:([Expr1005]=CASE WHEN [globalagg10...
275	1	|--Stream Aggregate(GROUP BY:([sp].[stateProvinceName], [ci]...
275	1	|--Sort(ORDER BY:([sp].[stateProvinceName] ASC, [ci].[city...
275	1	|--Hash Match(Inner Join, HASH:([sp].[stateProvinceID])=...
53	1	|--Clustered Index Scan(OBJECT:([AdventureWorks].[L...
275	1	|--Hash Match(Aggregate, HASH:([ci].[cityName], [ci]....
27659	1	|--Hash Match(Inner Join, HASH:([ci].[cityID])=([a].[...
285	1	|--Clustered Index Scan(OBJECT:([AdventureWor...
27659	1	|--Hash Match(Inner Join, HASH:([cu].[customerI...
18484	1	|--Hash Match(Inner Join, HASH:([cu].[addres...
18484	1	| |--Clustered Index Scan(OBJECT:([Adventu...
20068	1	| |--Clustered Index Scan(OBJECT:([Adventu...
27659	1	|--Clustered Index Scan(OBJECT:([Adventure...

AdventureWorks (Com índices)

275	1	SELECT ↵ ci.cityName AS 'City', ↵ sp.stateProvinceName AS 'S...
0	0	|--Compute Scalar(DEFINE:([Expr1005]=CASE WHEN [globalagg10...
275	1	|--Stream Aggregate(GROUP BY:([sp].[stateProvinceName], [ci]...
275	1	|--Sort(ORDER BY:([sp].[stateProvinceName] ASC, [ci].[city...
275	1	|--Hash Match(Inner Join, HASH:([sp].[stateProvinceID])=...
53	1	|--Clustered Index Scan(OBJECT:([AdventureWorks].[L...
275	1	|--Hash Match(Aggregate, HASH:([ci].[cityName], [ci]....
27659	1	|--Hash Match(Inner Join, HASH:([ci].[cityID])=([a].[...
285	1	|--Index Scan(OBJECT:([AdventureWorks].[Locat...
27659	1	|--Hash Match(Inner Join, HASH:([cu].[customerI...
18484	1	|--Merge Join(Inner Join, MERGE:([a].[addres...
20066	1	| |--Clustered Index Scan(OBJECT:([Adventu...
18484	1	| |--Index Scan(OBJECT:([AdventureWorks]....
27659	1	|--Index Scan(OBJECT:([AdventureWorks].[Sa...

Pesquisa de produtos associados a vendas com valor total monetário superior a 1000

AdventureWorksLegacy:

5985	1	SELECT DISTINCT ↵ p.EnglishProductName, ↵ OrderTotals.Ord...
5985	1	|--Parallelism(Gather Streams)
5985	8	|--Hash Match(Aggregate, HASH:([p].[EnglishProductName], [E...
31412	8	|--Parallelism(Repartition Streams, Hash Partitioning, PARTIT...
31412	8	|--Hash Match(Inner Join, HASH:([p].[ProductKey])=([s].[...
397	8	|--Parallelism(Repartition Streams, Hash Partitioning, P...
397	8	| |--Table Scan(OBJECT:([AdventureWorksLegacy].[d...
31412	8	|--Parallelism(Repartition Streams, Hash Partitioning, P...
31412	8	|--Hash Match(Inner Join, HASH:([AdventureWorksL...
13974	8	|--Bitmap(HASH:([AdventureWorksLegacy].[dbo]...
13974	8	| |--Parallelism(Repartition Streams, Hash Partit...
13974	8	| |--Filter(WHERE:([Expr1007]>(1.00000000...
0	0	| |--Compute Scalar(DEFINE:([Expr1007]...
27659	8	| |--Hash Match(Aggregate, HASH:([A...
60398	8	| |--Parallelism(Repartition Streams...
60398	8	| |--Table Scan(OBJECT:([Adven...
31417	8	|--Parallelism(Repartition Streams, Hash Partition...
31417	8	|--Table Scan(OBJECT:([AdventureWorksLega...

AdventureWorks (Sem Índice):

5985	1	SELECT DISTINCT ↵ p.productName, ↵ so.totalSalesAmt↵FRO...
5985	1	|--Hash Match(Aggregate, HASH:([p].[productName], [so].[totalSa...
31412	1	|--Hash Match(Inner Join, HASH:([p].[productID])=([sol].[produ...
397	1	|--Clustered Index Scan(OBJECT:([AdventureWorks].[Produc...
31412	1	|--Merge Join(Inner Join, MERGE:([so].[salesOrderID])=([sol]...
13974	1	|--Clustered Index Scan(OBJECT:([AdventureWorks].[Sale...
60124	1	|--Clustered Index Scan(OBJECT:([AdventureWorks].[Sale...

AdventureWorks (Com índice):

5985	1	SELECT DISTINCT ↵ p.productName, ↵ so.totalSalesAmt↵FRO...
5985	1	|--Hash Match(Aggregate, HASH:([p].[productName], [so].[totalSa...
31412	1	|--Hash Match(Inner Join, HASH:([so].[salesOrderID])=([sol].[sa...
13974	1	|--Index Seek(OBJECT:([AdventureWorks].[Sales].[SalesOrd...
60398	1	|--Merge Join(Inner Join, MERGE:([p].[productID])=([sol].[pr...
397	1	|--Clustered Index Scan(OBJECT:([AdventureWorks].[Pro...
60398	1	|--Index Scan(OBJECT:([AdventureWorks].[Sales].[SalesO...

Pesquisa de número de produtos vendidos por categoria por ano

AdventureWorksLegacy:

53	1	SELECT ↵ YEAR(s.OrderDate) AS SalesYear,↵ p.EnglishProduct...
0	0	|--Compute Scalar(DEFINE:([Expr1005]=CONVERT_IMPLICIT(int,[g...
53	1	|--Stream Aggregate(GROUP BY:([Expr1004], [p].[EnglishProdu...
288	1	|--Sort(ORDER BY:([Expr1004] ASC, [p].[EnglishProductSub...
288	1	|--Hash Match(Inner Join, HASH:([p].[ProductKey])=([s].[...
397	1	|--Table Scan(OBJECT:([AdventureWorksLegacy].[dbo]...
288	1	|--Hash Match(Aggregate, HASH:([s].[ProductKey], [E...
0	0	|--Compute Scalar(DEFINE:([Expr1004]=datepart(y...
60398	1	|--Table Scan(OBJECT:([AdventureWorksLegacy]...

AdventureWorks (Sem índices):

54	1	SELECT ↵ YEAR(so.orderDate) AS SalesYear,↵ cat.categoryNa...
0	0	|--Compute Scalar(DEFINE:([Expr1005]=CONVERT_IMPLICIT(int,[g...
54	1	|--Stream Aggregate(GROUP BY:([Expr1004], [cat].[categoryNa...
54	1	|--Sort(ORDER BY:([Expr1004] ASC, [cat].[categoryName] A...
54	1	|--Merge Join(Inner Join, MERGE:([cat].[categoryID])=([p]...
23	1	|--Clustered Index Scan(OBJECT:([AdventureWorks].[R...
54	1	|--Sort(ORDER BY:([p].[categoryID] ASC))
54	1	|--Hash Match(Aggregate, HASH:([p].[categoryID], ...
60398	1	|--Hash Match(Inner Join, HASH:([p].[productID]...
397	1	|--Clustered Index Scan(OBJECT:([Adventure...
60398	1	|--Merge Join(Inner Join, MERGE:([so].[sales...
0	0	|--Compute Scalar(DEFINE:([Expr1004]=da...
27659	1	| |--Clustered Index Scan(OBJECT:([Adve...
60398	1	|--Clustered Index Scan(OBJECT:([Adventu...

AdventureWorks (Com índices):

54	1	SELECT ↵ YEAR(so.orderDate) AS SalesYear,↵ cat.categoryNa...
0	0	|--Compute Scalar(DEFINE:([Expr1005]=CONVERT_IMPLICIT(int,[g...
54	1	|--Stream Aggregate(GROUP BY:([Expr1004], [cat].[categoryNa...
54	1	|--Sort(ORDER BY:([Expr1004] ASC, [cat].[categoryName] A...
54	1	|--Merge Join(Inner Join, MERGE:([cat].[categoryID])=([p]...
23	1	|--Clustered Index Scan(OBJECT:([AdventureWorks].[R...
54	1	|--Sort(ORDER BY:([p].[categoryID] ASC))
54	1	|--Hash Match(Aggregate, HASH:([p].[categoryID], ...
60398	1	|--Hash Match(Inner Join, HASH:([p].[productID]...
397	1	|--Index Scan(OBJECT:([AdventureWorks].[Pr...
60398	1	|--Merge Join(Inner Join, MERGE:([so].[sales...
0	0	|--Compute Scalar(DEFINE:([Expr1004]=da...
27659	1	| |--Clustered Index Scan(OBJECT:([Adve...
60398	1	|--Clustered Index Scan(OBJECT:([Adventu...

Contraste:

A execução das primeiras queries, na AdventureWorksLegacy, usam hash match com imensos valores, o que torna as pesquisas pesadas e dispendiosas.

Na execução das queries na AdventureWorks normalizada já conseguimos observar uma otimização da execução, ainda com o auxílio de hash match mas em em tabelas mais pequenas.

Enquanto na execução das queries na AdventureWorks com a ajuda dos índices não é necessário realizar hash match tantas vezes e o programa realiza um merge join que é mais eficiente.

9. Backup e Recuperação

Modelo de Recuperação:

Um backup completo (full) inicialmente, e backups diferenciais (differential) nas próximas vezes que se der backup à base de dados. Consideramos também fazer backups de logs diariamente.

Simulação de um crash:

Antes de qualquer coisa fazemos um backup dos logs com o comando “WITH NO_TRUNCATE”, este comando permite-nos fazer um backup da base dados mesmo que esta esteja danificada.

Em seguida, expulsamos todos os utilizadores, dado que o SQL Server não permite restaurar a base de dados se houverem sessões ativas.

Finalmente, restauramos a base de dados a partir dos backups “completo”, “diferencial” e “logs”, respetivamente. A ordem pela qual restauramos a base de dados é crucial para respeitar o sistema LSN.

10. Níveis de Acesso à Informação

```
------------------------
-- CRIAÇÃO DOS LOGINS --
------------------------

IF EXISTS (SELECT * FROM sys.server_principals WHERE name = 'AdminLogin') DROP LOGIN AdminLogin;
IF EXISTS (SELECT * FROM sys.server_principals WHERE name = 'SalesPersonLogin') DROP LOGIN
SalesPersonLogin;
IF EXISTS (SELECT * FROM sys.server_principals WHERE name = 'SalesTerritoryLogin') DROP LOGIN
SalesTerritoryLogin;
GO

CREATE LOGIN AdminLogin WITH PASSWORD = 'AdminPassword1!', CHECK_POLICY = OFF;
CREATE LOGIN SalesPersonLogin WITH PASSWORD = 'SalesPersonPassword2!', CHECK_POLICY = OFF;
CREATE LOGIN SalesTerritoryLogin WITH PASSWORD = 'SalesTerritoryPassword3!', CHECK_POLICY = OFF;
GO

-----------------------------
-- CRIAÇÃO DOS UTILIZADORES --
-----------------------------

USE AdventureWorks;
GO

IF EXISTS (SELECT * FROM sys.database_principals WHERE name = 'User_Admin') DROP USER User_Admin;
IF EXISTS (SELECT * FROM sys.database_principals WHERE name = 'User_SalesPerson') DROP USER
User_SalesPerson;
IF EXISTS (SELECT * FROM sys.database_principals WHERE name = 'User_SalesTerritory') DROP USER
User_SalesTerritory;
GO

CREATE USER User_Admin FOR LOGIN AdminLogin;
CREATE USER User_SalesPerson FOR LOGIN SalesPersonLogin;
CREATE USER User_SalesTerritory FOR LOGIN SalesTerritoryLogin;
GO

-------------------------
-- ROLES E PERMISSÕES --
-------------------------

-- Admininstrador
ALTER ROLE db_owner ADD MEMBER User_Admin;

-- SalesPerson
```

```
IF DATABASE_PRINCIPAL_ID('Role_Vendas') IS NOT NULL
    DROP ROLE Role_Vendas;
GO
CREATE ROLE Role_Vendas;
ALTER ROLE db_datareader ADD MEMBER Role_Vendas;
GRANT INSERT, UPDATE, DELETE ON SCHEMA::Sales TO Role_Vendas;

ALTER ROLE Role_Vendas ADD MEMBER User_SalesPerson;

-- SalesTerritory
IF DATABASE_PRINCIPAL_ID('Role_Territory') IS NOT NULL
    DROP ROLE Role_Territory;
GO
CREATE ROLE Role_Territory;

GRANT SELECT ON OBJECT::Sales.vAustraliaCustomers TO Role_Territory;
GO

-- 3. Adicionar o utilizador ao Role
ALTER ROLE Role_Territory ADD MEMBER User_SalesTerritory;
GO
```

11. Encriptação

Decidimos usar hashing nas passwords por ser um campo sensível. Com o hashing é possível garantir a segurança e a integridade dos dados, assegurando que as passwords nunca são guardadas em texto simples na base de dados.

12. Controlo de Transações

Para exemplificar um possível conflito, pensámos no seguinte cenário: Um gestor ou um processo automático tenta aplicar uma taxa sobre o preço de um produto ao mesmo tempo de outro. Isto poderia levar à perda de uma destas atualizações (lost-update).

Como solução propusemos utilizar “UPDLOCK” para bloquear possíveis alterações do valor a ser alterado, até que a transação acabe. Ou seja, se uma transação B ocorrer durante uma transação A, esta espera que a transação A (a primeira a ser chamada) acabe, e só depois é que começa. Desta forma, as duas alterações são preservadas.

13. NoSQL

Para satisfazer o requisito específico de logins de acesso criámos uma a tabela *UserManagement.AccessLogs* e criámos a coleção **access_logs.**

Estrutura da coleção:

```
{
  "_id": "ObjectId(...)",
  "user_email": "string",
  "timestamp": "datetime",
  "action": "string",
  "ip_address": "string",
  "status": "string"
}
```

14. Descrição da Demonstração

14.1 Script de demonstração

```
--------------------------------
-- Subcategories Trigger Test --
--------------------------------
select * from Reference.Category;
-- Insert test data
INSERT INTO Reference.Category (categoryName, parentCategoryID) VALUES ('Electronics', NULL);  -- 1
INSERT INTO Reference.Category (categoryName, parentCategoryID) VALUES ('Phones', 42);          --
2
INSERT INTO Reference.Category (categoryName, parentCategoryID) VALUES ('Smartphones', 42);     --
3
INSERT INTO Reference.Category (categoryName, parentCategoryID) VALUES ('Laptops', 42);         --
4
INSERT INTO Reference.Category (categoryName, parentCategoryID) VALUES ('Gaming Laptops', 42);  --
5
SELECT * FROM Reference.Category ORDER BY categoryID;
-- Delete the Electronics category and all its subcategories
DELETE FROM Reference.Category WHERE categoryName = 'Electronics';
-- Check results
SELECT * FROM Reference.Category ORDER BY categoryID;
DBCC CHECKIDENT ('Category', RESEED, 41);
-------------------------
-- Add Subcategory Test --
```

```
--------------------------
INSERT INTO Reference.Category (categoryName, parentCategoryID) VALUES ('Electronics', NULL);

EXEC spAddSubcategory @SubcategoryName = 'Smartphones', @CategoryName = 'Electronics';

SELECT * FROM Reference.Category ORDER BY categoryID;

DELETE FROM Reference.Category WHERE categoryName = 'Electronics';

SELECT * FROM Reference.Category ORDER BY categoryID;

DBCC CHECKIDENT ('Category', RESEED, 41);
-----------------------
-- Add Customer Test --
-----------------------
EXEC Sales.spAddCustomer @FirstName = 'Lúcio', @MiddleName = 'Correia', @LastName = 'Dos Santos',
@CustomerEmail = 'luciocdssds@gmail.com',
                   @CustomerPassword = 'frogger123', @SecurityQuestion = 'Why are you so angry?',
@SecurityAnswer = 'boop',
                   @Phone = '111111111', @Address = 'Paraíso', @PostalCode = '2805', @MartialStatus
= "S", @BirthDate = '1930-01-10',
                   @Education = 'Bachelors', @Occupation = "Professional", @Gender = "M";

select * from Sales.Customers
where firstName = 'Lúcio';
select * from [Location.Address]
where addressLine1 = 'Paraíso';
select * from [Location.Address]
where addressID = 20072;
--------------------------
-- Delete Customer Test --
--------------------------
EXEC Sales.spDeleteCustomer @CustomerEmail = 'luciocdssds@gmail.com';

select * from Sales.Customers c
join UserManagement.UserSecurity u ON c.userEmail = u.userEmail
where c.userEmail = 'luciocdssds@gmail.com';
--------------------------
-- spEditFirstName Test --
--------------------------
EXEC Sales.spAddCustomer @FirstName = 'Lúcio', @MiddleName = 'Correia', @LastName = 'Dos Santos',
@CustomerEmail = 'luciocdssds@gmail.com',
                   @CustomerPassword = 'frogger123', @SecurityQuestion = 'Why are you so angry?',
@SecurityAnswer = 'boop',
                   @Phone = '111111111', @Address = 'Paraíso', @PostalCode = '2805', @MartialStatus
= "S", @BirthDate = '1930-01-10',
                   @Education = 'Bachelors', @Occupation = "Professional", @Gender = "M";

select * from Sales.Customers where firstName = 'Lúcio';
```

```
EXEC Sales.spEditFirstName @ID = 20071, @first = 'Luciano';
select * from Sales.Customers where firstName = 'Luciano';
-------------------------
-- spEditMiddleName Test --
-------------------------
EXEC Sales.spEditMiddleName @ID = 20071, @middle = 'Corrente';
select * from Sales.Customers where middleName = 'Corrente';
-------------------------
-- spEditLastName Test --
-------------------------
EXEC Sales.spEditLastName @ID = 20071, @last = 'Abreu';
select * from Sales.Customers where lastName = 'Abreu';
---------------------------
-- spEditYearlyIncome Test --
---------------------------
select * from Sales.Customers where customerID = 20071;
EXEC Sales.spEditYearlyIncome @ID = 20071, @income = 10000;
select * from Sales.Customers where customerID = 20071;
-------------------------
-- spEditNumberCars Test --
-------------------------
select * from Sales.Customers where customerID = 20071;
EXEC Sales.spEditNumberCars @ID = 20071, @num = 2;
select * from Sales.Customers where customerID = 20071;
---------------------
-- spEditTitle Test --
---------------------
select * from Sales.Customers where customerID = 20071;
EXEC Sales.spEditTitle @ID = 20071, @title = Ms;
select * from Sales.Customers where customerID = 20071;
-----------------------------
-- spEditMartialStatus Test --
-----------------------------
select * from Sales.Customers where customerID = 20071;
EXEC Sales.spEditMartialStatus @ID = 20071, @stat = M;
select * from Sales.Customers where customerID = 20071;
---------------------
-- spEditGender Test --
---------------------
select * from Sales.Customers where customerID = 1;
EXEC Sales.spEditGender @ID = 1, @gender = 'F';
select * from Sales.Customers where customerID = 1;
select * from Reference.Gender;
-------------------------
-- spEditOccupation Test --
-------------------------
select * from Sales.Customers where customerID = 1;
EXEC Sales.spEditOccupation @ID = 1, @occupation = 'Professional';
select * from Sales.Customers where customerID = 1;
```

```
-------------------------
-- spEditEducation Test --
-------------------------
select * from Sales.Customers where customerID = 1;
select * from Reference.Education;
EXEC Sales.spEditEducation @ID = 1, @education = 'Bachelors';
select * from Sales.Customers where customerID = 1;
-----------------------
-- spEditAddress Test --
-----------------------
select * from [Location].Address where addressLine1 = '1085 Greenbelt Way';
select * from Sales.Customers where customerID = 1;
EXEC Sales.spEditAddress @ID = 1, @address = '1085 Greenbelt Way', @postal = 90012;
select * from Sales.Customers where customerID =1;
----------------------
-- spEditEmail Test ---
----------------------
select * from UserManagement.UserSecurity where userEmail='kendra14@adventure-works.com';
select * from Sales.Customers where customerID=1;
EXEC Sales.spEditUserEmail @ID = 1, @email = 'kendra14@adventure-works-new.com';
select * from Sales.Customers where customerID=1;
-----------------------------
-- EXEC Statistics Procedure --
-----------------------------
EXEC spdbStatistics;
-- Before migration
select * from [Statistics];
-- After migration
select * from [Statistics];
----------------------
-- Test vFullAddress --
----------------------
select * from Location.vFullAddress;
-------------------------------------
-- Test UserManagement.vUserProfile --
-------------------------------------
select * from UserManagement.vUserProfile;
-----------------------------
-- Test Sales.vOrderDetails --
-----------------------------
select * from Sales.vOrderDetails;
-------------------------------
-- Test Sales.vCustomerOrders --
-------------------------------
select * from Sales.vCustomerOrders;
-----------------------------------
-- Test Sales.vProductSalesSummary --
-----------------------------------
select * from Sales.vProductSalesSummary;
```

```
-----------------------------------
-- Test fnGetFullAddress --
-----------------------------------
SELECT Location.fnGetFullAddress(1);
```

15. Conclusões

1ª Fase:

Normalizámos uma base de dados antiga e fracamente relacionada, propusemos um novo modelo de entidade relação, e fizemos a implementação da nova base de dados escolhendo um layout apropriado e gerindo os seus respectivos filgroups.

Também criamos tabelas e procedures para os metadados de forma a termos acesso às informações importantes de catálogo.

2ª Fase:

Desenvolvemos as queries propostas para a segunda fase, e criámos índices com visto a otimizar as respetivas pesquisas.

Para fortificar a segurança da nossa base de dados, definimos diferentes utilizadores e os seus respetivos níveis de acesso à informação, e implementamos encriptação nas passwords dos customers.

Fabricámos um cenário hipotético de conflito de transações (lost-updated), e propusemos uma solução que permite efetuar as mesmas em concorrência, sem haver perda de dados.

Criámos a base de dados AdventureWorksWeb, com MongoDB, implementámos as coleções necessárias para ter a informação disponibilizada no site (específicas da ficha de grupo 3) e as queries propostas.

Por fim, definimos uma estratégia de backup da base de dados, simulámos um cenário de crash e a sequência de recuperação de dados.

Em suma:

Este projeto foi uma boa forma de consolidar alguns conceitos fundamentais da disciplina de Base de Dados do semestre anterior, tal como pôr em prática os conceitos novos da disciplina de Complementos de Base de Dados.