

|  |
| --- |
| Complementos de Bases de Dados 2022/2023  Licenciatura em Engª. Informática |
| Relatório Técnico |

Turma:

Horário de Laboratório:

Docente: Sara Batista

Grupo

Nº201900301, Gabriel Alves

# Sumário Executivo

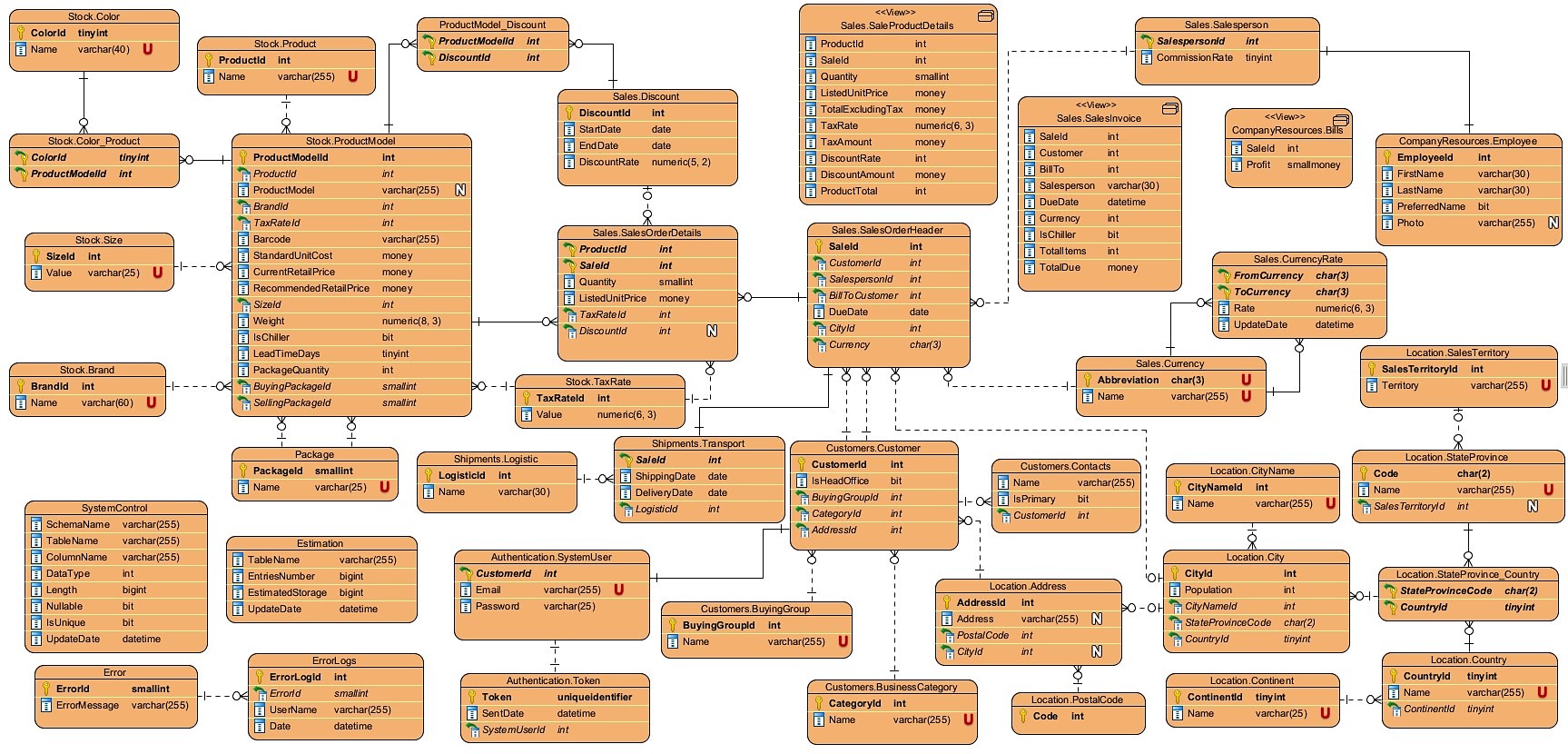
No âmbito da Unidade Curricular de Complementos de Bases de Dados, surgiu a necessidade de reformular o sistema de informação de uma empresa importadora e distribuidora de produtos, designada por “Wide World Importers” (WWI). Assim, este projeto tem como objetivo solucionar o problema apresentado e facilitar a adição de novas funcionalidades como a gestão de produtos, com a possibilidade de criar descontos para os mesmos, a gestão de utilizadores, através de email e password e ainda controlo de entregas e distribuidoras mais utilizadas, associadas às mesmas.

# Especificação de Requisitos

|  |  |  |
| --- | --- | --- |
| ID | Descrição | Implementado  (S/N) |
| *RF01* | O sistema deverá permitir criar uma venda |  |
| *RF02* | O sistema deverá permitir adicionar/remover produtos às vendas |  |
| *RF03* | O sistema deverá permitir alterar a quantidade de produtos numa venda |  |
| *RF04* | O sistema deverá permitir calcular o preço total de uma venda |  |
| *RF05* | O sistema deverá permitir a verificação da data de entrega de uma venda de acordo com as datas de entrega dos produtos a ela associados |  |
| *RF06* | O sistema deverá permitir adicionar apenas um tipo de “Chiller Stock” a uma venda |  |
| *RF07* | O sistema deverá permitir a autenticação por parte dos clientes com recurso ao ‘email’ e ‘password’ |  |
| *RF08* | O sistema deverá permitir adicionar/atualizar/remover utilizadores |  |
| *RF09* | O sistema deverá permitir a um utilizador recuperar a ‘password’ com recurso a um ‘token’ de verificação gerado e enviado automaticamente para o email do utilizador |  |
| *RF10* | O sistema deverá permitir aplicar/remover promoções aos produtos em Stock |  |
| *RF11* | O sistema deverá permitir alterar as datas de início e fim de uma promoção |  |
| *RF12* | *O sistema deverá permitir a verificação da data de validade de uma promoção antes de esta poder ser aplicada a um produto* |  |

# Modelo Relacional (Modelo de dados)

## Diagrama do Modelo Entidade Relação



Observação: Se necessário, para melhor perceção do diagrama, este está disponível na pasta “MR” - [mr.jpg](MR/WWI3.0.jpg) .

# Dimensionamento do Layout

A abordagem ao dimensionamento e predisposição do layout da nova base de dados teve em conta as recomendações, estratégias de preenchimento e regras de design de ficheiros e filegroups mencionadas pela Microsoft na respetiva documentação ([Microsoft docs - files & filegroups](https://learn.microsoft.com/en-us/sql/relational-databases/databases/database-files-and-filegroups?view=sql-server-ver16%23file-and-filegroup-fill-strategy)).

Tendo isto, foram criados filegroups secundários e a um destes foi-lhe atribuído a característica de “default”, deixando o filegroup primário apenas para “system tables and objects”. Além disso tabelas que concorrem por uma grande de espaço foram dispersadas por diferentes filegroups.

O seguinte ficheiro anexado contem as tabelas presentes na base de dados, o número de registos e os cálculos das dimensões.

[*scripts\Filegroups\Filegroups\_support.xlsx*](scripts/Filegroups/Filegroups_support.xlsx)

|  |  |  |
| --- | --- | --- |
| Nome Filegroup | Tabelas associadas | Parâmetros |
| *Wwiglobal\_primary* | System tables | *Dimensão inicial: 10MB*  *Dimensão final: 30MB*  *Taxa de crescimento: 10MB* |
| *Wwiglobal\_fg1* | *Continent, Country, StateProvince, SalesTerritory, State\_Country, CityName, City, Token, Error, Logistic, TaxRate, Currency, Color, Package, BusinessCategory* | *Dimensão inicial: 10MB*  *Dimensão final: 30MB*  *Taxa de crescimento: 10MB* |
| *Wwiglobal\_fg2 (default)* | SalesOrderHeader, Employee, ErrorLogs, ColumnInfo, Estimation, SystemUser, Discount, ProductModel, Size, Contact, BuyingGroup, Transport | *Dimensão inicial: 40MB*  *Dimensão final: 120MB*  *Taxa de crescimento: 40MB* |
| *Wwiglobal\_fg3* | SalesOrderDetails, CurrencyRate, Salesman, PostalCode, Address, Customer, Color\_Product, Product, Brand | *Dat1:*  *Dimensão inicial: 20MB*  *Dimensão final: 60MB*  *Taxa de crescimento: 20MB*  *Dat2:*  *Dimensão inicial: 30MB*  *Dimensão final: 80MB*  *Taxa de crescimento: 25MB* |

# Schemas

|  |  |
| --- | --- |
| Nome | Descrição |
| *Stock* | Este schema tem como objetivo agrupar as tabelas relacionadas com os produtos e suas características, como modelo, cor, marca, taxa aplicada, entre outras |
| *Sales* | Este schema tem como objetivo agrupar a informação referente às vendas como por exemplo as tabelas das faturas que contêm todas as vendas, produtos e suas informações |
| *Shipments* | Este schema tem como objetivo agrupar a informação referente às entregas dos produtos vendidos, agrupando assim tabelas como transportadora e características da entrega |
| *Customers* | Este schema tem como objetivo agrupar as tabelas relacionadas com os clientes, como clientes, companhias a que estes pertencem, contactos e categorias |
| *Location* | Este schema tem como objetivo agrupar as tabelas referentes às localizações, tendo assim associado países, estados, cidades, moradas |
| *CompanyResources* | Este schema tem como objetivo agrupar as tabelas relacionadas com recursos da wwi como por exemplo os seus funcionários |
| *Authentication* | Este schema tem como objetivo agrupar o sistema de autenticação de clientes incluindo tabelas como ‘SystemUser’ e ‘Token’ |

# Views

|  |  |
| --- | --- |
| Nome | Descrição |
| *Dbo.all\_fk\_cols* | Esta view permite obter a lista de *foreign keys* e as colunas, incluindo correspondentes tabelas, a que cada uma destas se refere |
| Dbo.all\_pk\_cols | Esta view permite obter a lista de colunas com chaves primárias e correspondentes tabelas e schemas |

# Functions

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Atributos | Descrição |
| *dbo.udf\_getUtilizador* | *Table* | *@id\_user INT* | *Permite obter informação detalhada sobre um utilizador* |
|  |  |  |  |

# Stored procedures

Observação: Os stored procedures referentes à migração de dados e à inserção de dados por defeito (e.g. sp\_insert\_error\_messages, …) foram ocultados por motivos apresentação.

|  |  |  |
| --- | --- | --- |
| Nome | Atributos | Descrição |
| *Dbo.sp\_generate\_action* | @tableName varchar(100) @action varchar(100) | Permite criar stored procedures para as operações *insert*, *update* e *delete* nas tabelas existentes na nova base de dados. |
| Dbo.sp\_throw\_messages | @errorId int = 1,  @state int = 1,  @param1 varchar(100) = null,  @param2 varchar(100) = null | Permite mostrar mensagens de erro ao utilizador e guardar os logs destes mesmos erros na tabela *ErrorLogs*. |
| Dbo.sp\_validate\_pk | @table varchar(100),  @col varchar(100),  @param\_val int | Permite verificar se uma coluna é a chave primária de correspondente tabela e se o parâmetro de entrada @param\_val é um valor válido desta coluna |
| Dbo.sp\_validate\_fk | @parent\_table varchar(100),  @parent\_col varchar(100),  @param\_val int | Permite verificar se a coluna de entrada é uma chave estrangeira e se o valor de entrada @param\_val existe na coluna @parent\_col |

# Triggers

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Tabela | Descrição |
| *dbo.tr\_utlizador\_historico* | *AFTER UPDATE* | *dbo.utilizador* | *Guarda o histórico de alterações sobre o utilizador* |
|  |  |  |  |
|  |  |  |  |

# Consultas

## Verificação da conformidade dos dados

* *Comparação do número de “customers” entre a base de dados antiga e a nova*

-- Number of Customers :)

select count(Customer) from WWI\_OldData.dbo.Customer;

GO

--

select count(\*) from Customers.Customer;

GO

* *Comparação do número de “customers” agrupados por cada categoria (“category”) existente entre a base de dados WWI\_OldData e WWIGlobal*

select Category, count(\*)

from WWI\_OldData.dbo.Customer

group by Category

GO

--

select c.CategoryId, bc.Name, count(\*)

from Customers.Customer c

left join Customers.BusinessCategory bc

on bc.CategoryId = c.CategoryId

group by c.categoryId, bc.Name

GO

* *Comparação do número total de vendas por funcionário (“employee”) entre a base de dados nova e antiga*

select e.Employee, count(distinct s.[WWI Invoice ID])

from WWI\_OldData.dbo.Sale s

join WWI\_OldData.dbo.Employee e

on e.[Employee Key] = s.[Salesperson Key]

group by  e.Employee

order by e.Employee

GO

--

select concat(e.FirstName, ' ', e.LastName), count(\*)

from Sales.SalesOrderHeader soh

join CompanyResources.Employee e

on e.EmployeeId = soh.SalespersonId

group by e.FirstName, e.LastName

order by e.FirstName, e.LastName

GO

* *Comparação do valor total arrecadado em vendas por cada produto (“stock item”)*
* select x.product, sum(sum) from ( select distinct case when si.product
* COLLATE Latin1\_General\_CS\_AS not like '%[ABCDEFGHIJKLMNOPKRSTUVXWYZ]% -%'
* THEN
* -- Product without sub-model
* case when si.product COLLATE Latin1\_General\_CS\_AS like '%([ABCDEFGHIJKLMNOPKRSTUVXWYZ]%'
* then
* -- Remove color from the product name
* trim(substring(si.product, 1, charindex('(', si.product)-2))
* else
* -- Products without color on the name
* case when si.product like '%[0-9][gm]' or si.product like '%[1-9]mm'
* then
* -- Remove size from products
* trim(SUBSTRING(si.product, 1,  len(si.product) - charindex(' ', reverse(si.product))))
* else
* -- Products without the size on the name
* si.product
* end
* end
* ELSE
* -- Product with sub-model
* trim(substring(si.product, 1, charindex('-', si.product)-2))
* END as product,
* sum(s.[Total Excluding Tax]) as sum
* from (
* select distinct [Stock Item] as product,
* [Stock Item Key]
* from WWI\_OldData.dbo.[Stock Item]
* ) si
* join WWI\_OldData.dbo.Sale s on si.[Stock Item Key] = s.[Stock Item Key]
* group by si.product ) x group by x.product order by x.product
* GO
* --
* select
* p.Name, sum(sod.Quantity \* pm.StandardUnitCost)
* from Sales.SalesOrderDetails sod
* inner join Stock.ProductModel pm on sod.ProductId = pm.ProductId
* inner join Stock.Product p on p.ProductId = pm.ProductId
* group by p.Name
* order by p.Name
* GO
* *Comparação do valor total arrecadado por ano e produto*
* select x.product, year, sum(sum) from ( select distinct case when si.product
* COLLATE Latin1\_General\_CS\_AS not like '%[ABCDEFGHIJKLMNOPKRSTUVXWYZ]% -%'
* THEN
* -- Product without sub-model
* case when si.product COLLATE Latin1\_General\_CS\_AS like '%([ABCDEFGHIJKLMNOPKRSTUVXWYZ]%'
* then
* -- Remove color from the product name
* trim(substring(si.product, 1, charindex('(', si.product)-2))
* else
* -- Products without color on the name
* case when si.product like '%[0-9][gm]' or si.product like '%[1-9]mm'
* then
* -- Remove size from products
* trim(SUBSTRING(si.product, 1,  len(si.product) - charindex(' ', reverse(si.product))))
* else
* -- Products without the size on the name
* si.product
* end
* end
* ELSE
* -- Product with sub-model
* trim(substring(si.product, 1, charindex('-', si.product)-2))
* END as product,
* sum(s.[Total Excluding Tax]) as sum,
* Year(s.[Invoice Date Key]) as year
* from (
* select distinct [Stock Item] as product,
* [Stock Item Key]
* from WWI\_OldData.dbo.[Stock Item]
* ) si
* inner join WWI\_OldData.dbo.Sale s on si.[Stock Item Key] = s.[Stock Item Key]
* group by si.product, s.[Invoice Date Key]) x
* group by x.product, year
* order by year, x.product
* GO
* --
* select p.name, YEAR(soh.DueDate), SUM(pm.StandardUnitCost \* sod.Quantity)
* from Sales.SalesOrderDetails sod
* inner join Stock.ProductModel pm on sod.Productid = pm.ProductModelId
* inner join Stock.Product p on pm.ProductId = p.ProductId
* inner join Sales.SalesOrderHeader soh on sod.SaleId = soh.Saleid
* group by  p.Name, YEAR(soh.DueDate)
* order by YEAR(soh.DueDate), p.Name
* GO
* *Comparação do valor total arrecadado em vendas por ano e cidade*
* select c.City, YEAR(s.[Invoice Date Key]), SUM(s.[Total Excluding Tax])
* from WWI\_OldData.dbo.Sale s
* inner join WWI\_OldData.dbo.City c on s.[City Key] = c.[City Key]
* group by c.City, c.[State Province], YEAR(s.[Invoice Date Key])
* order by YEAR(s.[Invoice Date Key]), c.City
* GO
* --
* select  cn.Name, YEAR(soh.DueDate), SUM(pm.StandardUnitCost \* sod.Quantity)
* from Sales.SalesOrderDetails sod
* inner join Stock.ProductModel pm on sod.Productid = pm.ProductModelId
* inner join Sales.SalesOrderHeader soh on sod.SaleId = soh.Saleid
* inner join [Location].City c on c.CityId = soh.CityId
* inner join [Location].CityName cn on cn.CityNameId = c.CityNameId
* group by  YEAR(soh.DueDate), cn.Name
* order by YEAR(soh.DueDate)

## Outras consultas

Foram executadas outras consultas, como a comparação dos produtos e seus modelos entre as bases de dados, mas foram ocultadas devido à sua extensão.

# Índices

## Views

|  |  |
| --- | --- |
| Nome | Descrição |
| *dbo.view\_TaxaCrescimento* | *Esta view permite a taxa de crescimento, …* |
|  |  |

## Índices

|  |  |  |
| --- | --- | --- |
| Designação | Tabela | Justificação/Consultas |
| *Nome do índice* | *Tabela indexada* | *Consulta otimizada, justificando as opções tomadas, e.g., colunas indexadas, colunas incluídas…* |
|  |  |  |

## Otimização e Execução de Consultas

*Apresentar os resultados da execução das consultas, com informação sobre as estatísticas e planos de execução, nos seguintes cenários:*

*- Sobre a base de dados original (não normalizada);*

*- Sobre a base de dados otimizada (normalizada) sem índices;*

*- Sobre a base de dados otimizada (normalizada) com índices.*

# Backup e Recuperação

*Descrição fundamentada da política de backups implementada, e descrição dos possíveis cenários de recuperação.*

# Segurança e Controlo de Acessos

*Definição de Utilizadores, Roles, Schemas e Encriptação.*

## Níveis de acesso à informação

## Encriptação

# Controlo de Concorrência

*Definir níveis de isolamento adotados no controlo transacional.*

# MongoDB

*Descrição do código implementado em MongoDB, justificando as opções tomadas.*

# Descrição da Demonstração

## Requisitos implementados

*Para os requisitos apresentados no capitulo 2, identificar a solução implementada através da referência ao código utilizado (restrições da BD, consultas/views, SPs, triggers,…).*

|  |  |
| --- | --- |
| ID | Solução |
| *RF0#* | *Restrição CHECK na tabela ….* |
| *RF##* | *SP\_nome\_... e Trigger tg\_....* |
|  |  |
| *RFM01* |  |

## Scripts de demonstração

Antes da execução dos *scripts* é necessário ter em atenção que o correto funcionamento da migração de dados está dependente da existência e disponibilidade do modelo antigo “WWI\_OldData” e da importação do ficheiro “data/categories.csv” para esta base o mesmo.

Além das condições anteriormente enumeradas, pode ser ainda necessário **alterar o caminho para *states.txt***no stored procedure ***sp\_import\_states,*** para o correto funcionamento da importação de dados***.*** O caminho pode ser alterado diretamente no stored procedure ou ser inserido por parâmetro na linha de execução (**linha 1223 -** *migration.sql*).

Para testar o funcionamento da nova base de dados normalizada é sugerido a seguinte ordem de execução dos ficheiros presentes na pasta *“scripts”*.

|  |  |  |
| --- | --- | --- |
| Número | Nome do ficheiro | Descrição |
| *1* | *Filegroups/filegroupsW.sql ou Filegroups/filegroupsLinux.sql* | *Criação dos ficheiros da base de dados* |
| *2* | *ddl/createTables.sql* | *Criação das tabelas da base de dados* |
| *3* | *Migration/migration.sql* | *Migração dos dados da base de dados antigos para o novo modelo* |
| *4* | *Migration/migration\_check.sql* | *Verificação dos dados* |
| *4* | *errorHandling.sql* | *Tratamento de erros* |
| *5* | *Generator.sql* | *Gerador de crud stored procedures* |

# Conclusões

Com o conhecimento lecionado na unidade curricular, consegui elaborar todas as funcionalidades propostas para este projeto. Além disso, o projeto serviu como consolidação das matérias aprendidas, o que facilitará a implementação de trabalhos futuros.