IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Complementos de Bases de Dados

2022/2023

Licenciatura em Engª. Informática

Relatório Técnico

Turma: 2ªL_EI-SW-08
Horário de Laboratório: 14:30h – 16:30h
Docente: Gabriel Pestana
Grupo
Nº202000753, Nuno Reis

1. Introdução

Este relatório tem como objetivo documentar e apresentar algumas justificações para as principais decisões tomadas na primeira fase de desenvolvimento do projeto de CBD, projeto esse que visa a familiarização com a administração de bases de dados relacionais. Ao realizar este projeto será consolidada e posta em prática toda a matéria teórica.

A Wide World Importers (WWI), é uma empresa importadora e distribuidora de produtos que opera no mercado de vendas a retalho. O seu sistema de informação está baseado numa base de dados não normalizada e está desatualizado. A administração tomou a decisão de reformular o sistema, num novo ERP que lhe permita gerir, de forma integrada, todo o processo de vendas.

O projeto irá incidir na modelação e integração de uma nova base de dados, que dará apoio a esse novo sistema de informação. Foram exportados e disponibilizados fragmentos de informação do sistema existente, estes apresentam-se fracamente relacionados e carecem de uma otimização segundo as boas praticas de modelação e regras da normalização.

2. Especificação de Requisitos

ID	Descrição	Implementado (S/N)
RF01	O sistema não deverá permitir importar o mesmo registo duas vezes.	S
RF02	O sistema deverá guardar o registo de tokens gerados (tokens gerados ao tentar recuperar password).	S
RF03	O sistema deverá verificar se um token está ativo (tokens têm validade de 24 horas)	S
RF04	O sistema não deverá permitir que numa venda existam produtos que necessitem de refrigeração e produtos que não necessitem.	S
RF05	O sistema não deverá permitir a atribuição de uma promoção que não esteja ativa.	S
RF06	O sistema deverá eliminar uma venda se forem removidos todos os produtos.	S
RM01	O sistema deverá guardar a password do utilizador usando codificação uma em SHA1.	S
RM02	O sistema deverá alterar a validade de um token quando um utilizador criar um segundo token. De forma a existir apenas um token valido para cada utilizador.	S
RM03	O sistema deverá calcular a data prevista de entrega e os valores associados a uma venda, assim que esta é finalizada.	S
RM04	O sistema deverá alterar o stock de um produto quando este é adicionado ou removido de uma venda.	S
RM05	O sistema não deverá permitir que existam dois utilizadores com o mesmo email.	S
RM06	O sistema não deverá permitir adicionar um produto a uma venda acabada.	S

3. Alterações/Melhorias ao Relatório da 1ª Fase

Nesta segunda fase passaram a ser utilizadas as convenções de escrita adequadas para o Microsoft SQL Server.

3.1 Modelo Relacional

A tabela RH.ErrorLog deixou de estar ligada á tabela RH.SysUser.

3.2 Layout

Os filegroups e schemas foram alterados:

- Filegroups para tabelas em que é mais frequente a escrita ou a leitura de registos (Read para as de leitura e Write para as de escrita).
- Schemas para as tabelas referentes a dados de utilizadores (RH), do armazém (Storage) e das vendas (Sales).

3.3 Programação

- Funções e Stored Procedures:

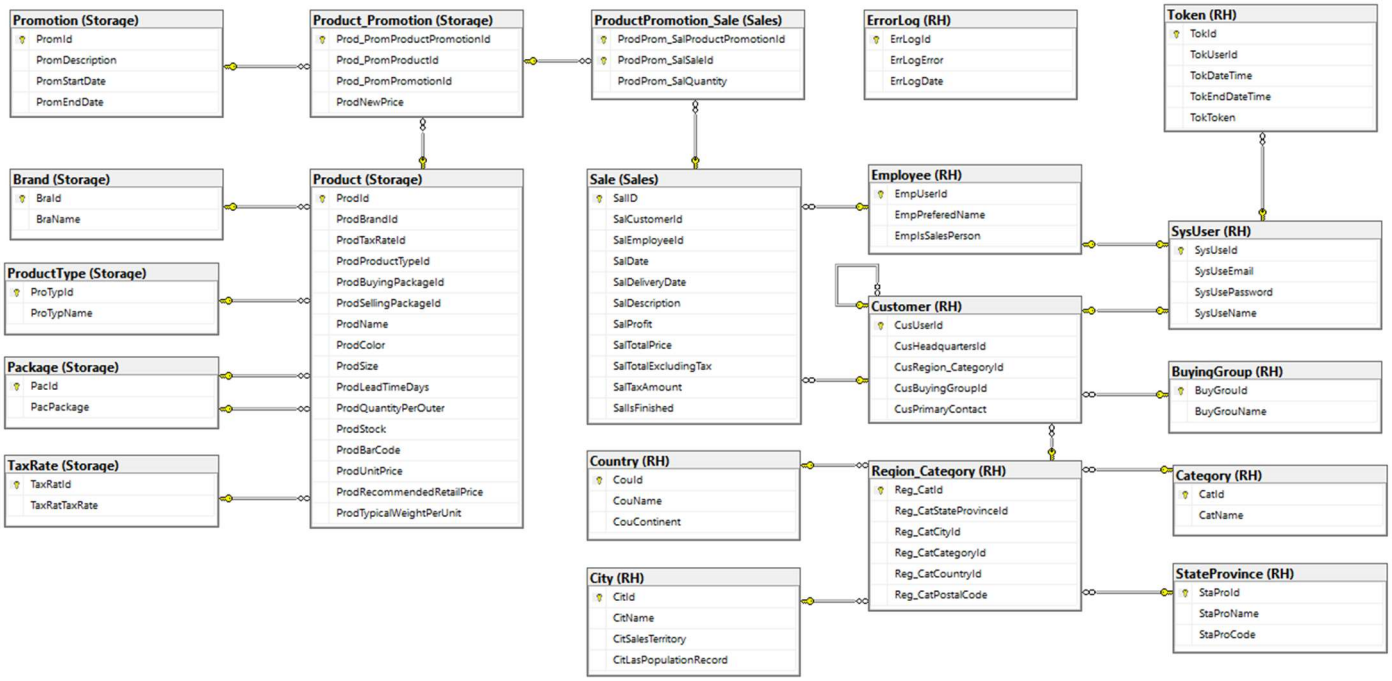
  Os erros, em todos os stored procedures e funções, passaram a ser lançados utilizando o RaiseError.

- Geradores:

  Foram nesta fase adicionados todos os stored procedures geradores.

4. Relacional (*Modelo de dados*)

4.1 Diagrama do Modelo Relacional



5. Definição do Layout

5.1 Identificação do espaço ocupado por tabela

Nome Tabela	Dimensão do Registo	Nº de Registos (inicial/final)
RH.Country	44	0 – 1
RH.City	108	116294 - 23272
RH.StateProvince	59	57 - 60
RH.Category	24	5 - 5
RH.BuyingGroup	24	0 - 2
RH.SysUser	124	0 - 421
RH.Region_Category	24	0 - 401
RH.Customer	56	402 - 402
RH.Employee	15	212 - 20
Storage.Package	29	0 - 5
Storage.Brand	29	0 - 2
Storage.ProductType	29	0 - 2
Storage.TaxRate	12	0 - 6
Storage.Product	250	671 - 227
Storage.Promotion	110	0 - 1
Storage.Product_Promotion	20	0 - 227
Sales.Sale	155	228265 - 70510
Sales.ProductPromotion_Sale	12	0 - 228265
RH.ErrorLog	312	0 -

RH.Token	28	0 -

5.2 Especificação dos Filegroups

Nome Filegroup	Tabelas associadas	Parâmetros
Read	RH.Country RH.City RH.StateProvince RH.Category RH.BuyingGroup RH.Region_Category Storage.Package Storage.Brand Storage.ProductType Storage.TaxRate	Dimensão inicial: 2MB Dimensão final: 4MB Taxa de crescimento: 100%
Write	RH.SysUser RH.Customer RH.Employee Storage.Product Storage.Promotion Storage.Product_Promotion Sales.Sale Sales.ProductPromotion_Sale RH.ErrorLog RH.Token	Dimensão inicial: 15MB Dimensão final: 30MB Taxa de crescimento: 100%

5.3 Schemas

Nome	Descrição
RH	Este schema tem como objetivo o agrupamento das tabelas que contêm dados relacionados com os utilizadores.
Storage	Este schema tem como objetivo o agrupamento das tabelas que contêm dados relacionados com os produtos e promoções.
Sales	Este schema tem como objetivo o agrupamento das tabelas que contêm dados relacionados com as vendas.

6. Verificação da migração de dados

6.1 Consultas sobre a base de dados original

No final do ficheiro *WWWI_DS Query.sql* estão presentes as seguintes consultas:

- Número de clientes
- Número de clientes por categoria
- Total de vendas por funcionário
- Total monetário de vendas por produto
- Total monetário de vendas por produto por ano
- Total monetário de vendas por cidade por ano

6.2 Consultas sobre a nova base de dados

No final do ficheiro *WWWIGlobal Query.sql* estão presentes as seguintes consultas:

- Número de clientes
- Número de clientes por categoria
- Total de vendas por funcionário
- Total monetário de vendas por produto
- Total monetário de vendas por produto por ano
- Total monetário de vendas por cidade por ano

7. Programação

7.1 Views

Nome	Descrição
RH.viewNorthAmericaCountry	Esta view permite obter a lista de países na américa do norte.
RH.viewCitySalesTerritory	Esta view permite obter a lista de território de vendas.
RH.viewRegion_Category	Esta view permite obter a lista de relações entre cidade, estado, continente e categoria, com os campos das chaves externas.
RH.viewCustomer	Esta view permite obter a lista de customers, com os campos das chaves externas.
RH.viewEmployee	Esta view permite obter a lista funcionários.
Storage.viewProduct	Esta view permite obter a lista de produtos, com os campos das chaves externas.

Storage.viewProductPromotion	Esta view permite obter a lista de produtos associados a promoções, com os campos das chaves externas.
Sales.viewProductPromotionSale	Esta view permite obter a lista de produtos e promoções associados a vendas, com os campos das chaves externas.

7.2 Functions

Nome	Atributos	Requisito	Descrição
RH.udf_countryExists	@countryName varchar (20), @continentName varchar (20)	RF01	Permite obter o id de um país com o nome do país e nome do continente passados, ou 0 caso não exista.
RH.udf_countryExistsById	@countryId int	RF01	Permite obter o id de um país com o id passado, ou 0 caso não exista.
RH.udf_stateProvinceExists	@stateProvinceName varchar (50)	RF01	Permite obter o id de um estado com o nome passado, ou 0 caso não exista.
RH.udf_stateProvinceExistsByCode	@stateProvinceCode varchar(50)	RF01	Permite obter o id de um estado com o código passado, ou 0 caso não exista.
RH.udf_stateProvinceExistsById	@stateProvinceId int	RF01	Permite obter o id de um estado com o id passado, ou 0 caso não exista.
RH.udf_cityExists	@cityName varchar (50)	RF01	Permite obter o id de uma cidade com o nome passado, ou 0 caso não exista.
RH.udf_cityExistsById	@cityId int	RF01	Permite obter o id de uma cidade com o id passado, ou 0 caso não exista.
RH.udf_categoryExists	@categoryName varchar (50)	RF01	Permite obter o id de uma categoria com o nome passado, ou 0 caso não exista.
RH.udf_categoryExistsById	@categoryId int	RF01	Permite obter o id de uma categoria com o id passado, ou 0 caso não exista.
RH.udf_regionCategoryExists	@countryId int, @stateProvinceId int, @cityId int,	RF01	Permite obter o id de uma relação entre pais, estado, cidade e categoria com os

	@categoryId int		ids associados, passados, ou 0 caso não exista.
RH.udf_regionCategoryExistsById	@regionCategoryId int	RF01	Permite obter o id de uma relação entre pais, estado, cidade e categoria com o id passado, ou 0 caso não exista.
RH.udf_buiyngGroupExists	@buiyngGroupName varchar (50)	RF01	Permite obter o id de um grupo com o nome passado, ou 0 caso não exista.
RH.udf_buiyngGroupExistsById	@buiyngGroupId int	RF01	Permite obter o id de um grupo com o id passado, ou 0 caso não exista.
RH.udf_sysUserExists	@sysUserEmail varchar (50)	RF01	Permite obter o id de um utilizador com o email passado, ou 0 caso não exista.
RH.udf_sysUserExistsByName	@sysUserName varchar(50)	RF01	Permite obter o id de um utilizador com o nome passado, ou 0 caso não exista.
RH.udf_sysUserExistsById	@sysUserId int	RF01	Permite obter o id de um utilizador com o id passado, ou 0 caso não exista.
RH.udf_customerExists	@customerName varchar (50)	RF01	Permite obter o id de um cliente com o nome passado, ou 0 caso não exista.
RH.udf_customerExistsById	@customerId int	RF01	Permite obter o id de um cliente com o id passado, ou 0 caso não exista.
RH.udf_employeeExists	@employeeName varchar (50)	RF01	Permite obter o id de um funcionário com o nome passado, ou 0 caso não exista.
RH.udf_employeeExistsById	@employeeId int	RF01	Permite obter o id de um funcionário com o id passado, ou 0 caso não exista.
RH.udf_tokenExists	@token int	RF01	Permite obter o id de um token com o token passado, ou 0 caso não exista.
RH.udf_tokenExistsByUser	@token int, @userId int	RF01	Permite obter o id de um token com o token e id de utilizador passados caso seja válido, ou 0 caso não exista.
RH.udf_tokenExistsById	@tokenId int	RF01	Permite obter o id de um token com o id passado, ou 0 caso não exista.

RH.udf_fnHashPassword	@password varchar (20)	RF01	Permite encriptar uma palavra-passe usando codificação SH1.
Storage.udf_taxRateExists	@taxRate float	RF01	Permite obter o id de uma taxa com o valor passado, ou 0 caso não exista.
Storage.udf_taxRateExistsById	@taxRateId int	RF01	Permite obter o id de uma taxa com o id passado, ou 0 caso não exista.
Storage.udf_productTypeExists	@productType varchar (25)	RF01	Permite obter o id de um tipo de produto com o nome passado, ou 0 caso não exista.
Storage.udf_productTypeExistsById	@productTypeId int	RF01	Permite obter o id de um tipo de produto com o id passado, ou 0 caso não exista.
Storage.udf_packageExists	@package varchar (25)	RF01	Permite obter o id de um pacote com o nome passado, ou 0 caso não exista.
Storage.udf_packageExistsById	@packageId int	RF01	Permite obter o id de um pacote com o id passado, ou 0 caso não exista.
Storage.udf_brandExists	@brand varchar (25)	RF01	Permite obter o id de uma marca com o nome passado, ou 0 caso não exista.
Storage.udf_brandExistsById	@brandId int	RF01	Permite obter o id de uma marca com o id passado, ou 0 caso não exista.
Storage.udf_productExists	@productName varchar (100)	RF01	Permite obter o id de um produto com o nome passado, ou 0 caso não exista.
Storage.udf_productExistsById	@productId int	RF01	Permite obter o id de um produto com o id passado, ou 0 caso não exista.
Storage.udf_promotionExists	@promotionDescription varchar (100)	RF01	Permite obter o id de uma promoção com a descrição passada, ou 0 caso não exista.
Storage.udf_promotionExistsById	@promotionId int	RF01	Permite obter o id de uma promoção com o id passado, ou 0 caso não exista.
Storage.udf_productPromotionExists	@productId int, @promotionId int	RF01	Permite obter o id de uma relação entre produto e promoção com os ids

			associados, passados, ou 0 caso não exista.
Storage.udf_productPromotionExistsById	@productPromotionId int	RF01	Permite obter o id de uma relação entre produto e promoção com o id passado, ou 0 caso não exista.
Storage.udf_productPromotionExistsByName	@name varchar(100), @promotion varchar(100)	RF01	Permite obter o id de uma relação entre produto e promoção com os nomes passados, ou 0 caso não exista.
Storage.udf_productStock	@product varchar(100)	RF01	Permite obter o stock de um entre produto com o nome passado.
Sales.udf_saleExistsById	@saleId int	RF01	Permite obter o id de uma venda com o id passado, ou 0 caso não exista.
Sales.udf_productPromotionSaleExists	@productPromotionId int, @saleId int	RF01	Permite obter o id de uma relação entre produto-promoção e venda com os ids associados, passado, ou 0 caso não exista.
Sales.udf_saleExistsByDescription	@description varchar(100)	RF01	Permite obter o id de uma venda com a descrição passada, ou 0 caso não exista.
Sales.udf_saleType	@sale varchar(100)	RF01	Permite obter o tipo de uma venda com a descrição passada.

7.3 Stored procedures

Nome	Atributos	Requisito	Descrição
dbo.Migrate_OldData_CityTable		RF01	Permite migrar os registos da tabela OldData.City para as tabelas UsersInfo.Country, UsersInfo.StateProvince e UsersInfo.City
dbo.Migrate_OldData_StatesTable		RF01	Permite migrar os registos da tabela OldData.States para a tabela UsersInfo.StateProvince
dbo.Migrate_OldData_lookupTable		RF01	Permite migrar os registos da tabela OldData.lookup para a tabela UsersInfo.Category

dbo.Migrate_OldData_CustomerTable		RF01	Permite migrar os registos da tabela OldData.Customer para as tabelas UsersInfo.SysUser, UsersInfo.Customer, UsersInfo.Region_Category e UsersInfo.BuyingGroup
dbo.Migrate_OldData_EmployeeTable		RF01	Permite migrar os registos da tabela OldData.Employee para as tabelas UsersInfo.Employee e UsersInfo.Region_Category
dbo.Migrate_OldData_ItemStockTable		RF01	Permite migrar os registos da tabela OldData.[Stock Item] para as tabelas ProductsInfo.Package, ProductsInfo. Brand, ProductsInfo.TaxRate, ProductsInfoProductType, ProductsInfo.Product e ProductsInfo.Product_Promotion
dbo.Migrate_OldData_SaleTable		RF01	Permite migrar os registos da tabela OldData.Sale para as tabelas SalesInfo.Sale e SalesInfo.ProductPromotion_Sale
dbo.MigrateAll		RF01	Permite executar todos os stored procedures de migração
RH.sp_recuperarPassword	@userEmail VARCHAR (50), @token int, @newUserPassword VARCHAR (20)		Permite que se altere a palavra-passe de um utilizador.
Sales.sp_finishSale	@id int		Permite finalizar uma venda.

7.4 Triggers

Nome	Tipo	Tabela	Requisito	Descrição
RH.tr_validade_token	INSERT	UsersInfo.Token	RM02	Altera a data de validade de um token repetido.
Sales.tr_eliminateSale	DELETE	SalesInfo.ProductPromotion_Sale	RF06	Elimina uma venda se forem removidos todos os produtos.
Sales.tr_calculateSaleInfo	UPDATE	SalesInfo.Sale	RM03	Calcula a data prevista de entrega e os valores associados a uma venda finalizada.

8. Catálogo/Metadados

8.1 Geradores

Nome	Atributos	Descrição
RH.errorLog_insert	@error varchar (300)	Implementa o procedimento para inserir registos na tabela RH.ErrorLog.
RH.country_insert	@countryName varchar (20), @continentName varchar (20)	Implementa o procedimento para inserir registos na tabela RH.Country.
RH.country_update	@countryId int, @countryName varchar (20), @continentName varchar (20)	Implementa o procedimento para atualizar registos na tabela RH.Country.
RH.country_delete	@countryId int	Implementa o procedimento para apagar registos na tabela RH.Country.
RH.stateProvince_insert	@stateProvinceName varchar (50), @stateProvinceCode varchar (5)	Implementa o procedimento para inserir registos na tabela RH.StateProvince.
RH.stateProvince_update	@stateProvinceId int, @stateProvinceName varchar (50), @stateProvinceCode varchar (5)	Implementa o procedimento para atualizar registos na tabela RH.StateProvince.
RH.stateProvince_delete	@stateProvinceId int	Implementa o procedimento para apagar registos na tabela RH.StateProvince.
RH.city_insert	@cityName varchar (50), @citySalesTerritory varchar (50), @cityLastPopulationRecord int	Implementa o procedimento para inserir registos na tabela RH.City.
RH.city_update	@cityId int, @cityName varchar (50), @citySalesTerritory varchar (50), @cityLastPopulationRecord int	Implementa o procedimento para atualizar registos na tabela RH.City.
RH.city_delete	@cityId int	Implementa o procedimento para apagar registos na tabela RH.City.
RH.category_insert	@categoryName varchar (50)	Implementa o procedimento para inserir registos na tabela RH.Category.

RH.category_update	@categoryId int, @categoryName varchar (50)	Implementa o procedimento para atualizar registos na tabela RH.Category.
RH.category_delete	@categoryId int	Implementa o procedimento para apagar registos na tabela RH.Category.
RH.regionCategory_insert	@countryId int, @stateProvinceId int, @cityId int, @categoryId int, @postalCode int	Implementa o procedimento para inserir registos na tabela RH.Region_Category.
RH.regionCategory_update	@regionCategoryId int, @countryId int, @stateProvinceId int, @cityId int, @categoryId int, @postalCode int	Implementa o procedimento para atualizar registos na tabela RH.Region_Category.
RH.regionCategory_delete	@regionCategoryId int	Implementa o procedimento para apagar registos na tabela RH.Region_Category.
RH.buiyngGroup_insert	@buiyngGroupName varchar (50)	Implementa o procedimento para inserir registos na tabela RH.BuyingGroup.
RH.buiyngGroup_update	@buiyngGroupId int, @buiyngGroupName varchar (50)	Implementa o procedimento para atualizar registos na tabela RH.BuyingGroup.
RH.buiyngGroup_delete	@buiyngGroupId int	Implementa o procedimento para apagar registos na tabela RH.BuyingGroup.
RH.sysUser_insert	@sysUserName varchar (50), @sysUserEmail varchar (50), @sysUserPassword varchar (50)	Implementa o procedimento para inserir registos na tabela RH.SysUser.
RH.sysUser_update	@sysUserId int, @sysUserName varchar (50), @sysUserEmail varchar (50), @sysUserPassword varchar (50)	Implementa o procedimento para atualizar registos na tabela RH.SysUser.
RH.sysUser_delete	@sysUserId int	Implementa o procedimento para apagar registos na tabela RH.SysUser.
RH.customer_insert	@userId int, @headquartersId int, @regionCategoryId int,	Implementa o procedimento para inserir registos na tabela RH.Customer.

	@buyingGroupId int, @primaryContact varchar (40)	
RH.customer_update	@userId int, @headquartersId int, @regionCategoryId int, @buyingGroupId int, @primaryContact varchar (40)	Implementa o procedimento para atualizar registos na tabela RH.Customer.
RH.customer_delete	@userId int	Implementa o procedimento para apagar registos na tabela RH.Customer.
RH.employee_insert	@userId int, @employeePreferedName varchar (10), @employeeIsSalesPerson BIT	Implementa o procedimento para inserir registos na tabela RH.Employee.
RH.employee_update	@userId int, @employeePreferedName varchar (10), @employeeIsSalesPerson BIT	Implementa o procedimento para atualizar registos na tabela RH.Employee.
RH.employee_delete	@userId int	Implementa o procedimento para apagar registos na tabela RH.Employee.
RH.token_insert	@tokenUserId INT	Implementa o procedimento para inserir registos na tabela RH.Token.
RH.token_update	@tokenId int	Implementa o procedimento para atualizar registos na tabela RH.Token.
Storage.taxRate_insert	@taxRate float	Implementa o procedimento para inserir registos na tabela Storage.TaxRate.
Storage.taxRate_update	@taxRateId int, @taxRate float	Implementa o procedimento para atualizar registos na tabela Storage.TaxRate.
Storage.taxRate_delete	@taxRateId int	Implementa o procedimento para apagar registos na tabela Storage.TaxRate.
Storage.productType_insert	@productType varchar (25)	Implementa o procedimento para inserir registos na tabela Storage.ProductType.
Storage.productType_update	@productTypeId int, @productType varchar (25)	Implementa o procedimento para atualizar registos na tabela Storage.ProductType.

Storage.productType_delete	@productTypeId int	Implementa o procedimento para apagar registos na tabela Storage.ProductType.
Storage.package_insert	@package varchar (25)	Implementa o procedimento para inserir registos na tabela Storage.Package.
Storage.package_update	@packageId int, @package varchar (25)	Implementa o procedimento para atualizar registos na tabela Storage.Package.
Storage.package_delete	@packageId int	Implementa o procedimento para apagar registos na tabela Storage.Package.
Storage.brand_insert	@brand varchar (25)	Implementa o procedimento para inserir registos na tabela Storage.Brand.
Storage.brand_update	@brandId int, @brand varchar (25)	Implementa o procedimento para atualizar registos na tabela Storage.Brand.
Storage.brand_delete	@brandId int	Implementa o procedimento para apagar registos na tabela Storage.Brand.
Storage.product_insert	@brandId int, @taxRateId int, @productTypeId int, @buyingPackageId int, @sellingPackageId int, @productName varchar (100), @productColor varchar (50), @productSize varchar (20), @productLeadTimeDays int, @productQuantityPerOuter int, @productStock int, @productBarCode varchar (20), @productUnitPrice float, @productRecommendedRetailPrice float, @productTypicalWeightPerUnit float	Implementa o procedimento para inserir registos na tabela Storage.Product.
Storage.product_update	@productId int, @brandId int, @taxRateId int, @productTypeId int, @buyingPackageId int, @sellingPackageId int, @productName varchar (100), @productColor varchar (50),	Implementa o procedimento para atualizar registos na tabela Storage.Product.

	@productSize varchar (20), @productLeadTimeDays int, @productQuantityPerOuter int, @productStock int, @productBarCode varchar (20), @productUnitPrice float, @productRecommendedRetailPrice float, @productTypicalWeightPerUnit float	
Storage.product_delete	@productId int	Implementa o procedimento para apagar registos na tabela Storage.Product.
Storage.promotion_insert	@promotionDescription varchar (100), @promotionStartDate varchar (20), @promotionEndDate varchar (20)	Implementa o procedimento para inserir registos na tabela Storage. Promotion.
Storage.promotion_update	@promotionId int, @promotionDescription varchar (100), @promotionStartDate varchar (20), @promotionEndDate varchar (20)	Implementa o procedimento para atualizar registos na tabela Storage.Promotion.
Storage.promotion_delete	@promotionId int	Implementa o procedimento para apagar registos na tabela Storage.Promotion.
Storage.productPromotion_insert	@productId int, @promotionId int	Implementa o procedimento para inserir registos na tabela Storage.Product_Promotion.
Storage.productPromotion_update	@productPromotionId int, @productId int, @promotionId int	Implementa o procedimento para atualizar registos na tabela Storage.Product_Promotion.
Storage.productPromotion_delete	@productPromotionId int	Implementa o procedimento para apagar registos na tabela Storage.Product_Promotion.
Sales.sale_insert	@saleID int, @customerId int, @employeeId int, @saleDescription varchar (100)	Implementa o procedimento para inserir registos na tabela Sales. Sale.
Sales.sale_update	@saleID int, @customerId int, @employeeId int,	Implementa o procedimento para atualizar registos na tabela Sales. Sale.

	@saleDescription varchar (100)	
Sales.sale_delete	@saleID int	Implementa o procedimento para apagar registos na tabela Sales. Sale.
Sales.productPromotionSale_insert	@productPromotionId int, @saleID int, @quantity int	Implementa o procedimento para inserir registos na tabela Sales.ProductPromotion_Sale.
Sales.productPromotionSale_update	@productPromotionId int, @saleID int, @quantity int	Implementa o procedimento para atualizar registos na tabela Sales.ProductPromotion_Sale.
Sales.productPromotionSale_delete	@productPromotionId int, @saleID int	Implementa o procedimento para apagar registos na tabela Sales.ProductPromotion_Sale.

8.2 Monitorização

Nome	Atributos	Descrição
dbo.viewLastMonitorizacaoColunas		Monitora os espaços ocupados por cada resisto e por cada tabela.

9. Índices

9.1 Views

Nome	Descrição
dbo.viewSalesPerCity_OldData	Esta view permite a pesquisa de vendas por cidade sobre a base de dados original.
dbo.viewSalesPerCity	Esta view permite a pesquisa de vendas por cidade sobre a base de dados otimizada.
dbo.viewYearGrowthPerSale_OldData	Esta view permite o cálculo para as vendas da taxa de crescimento de cada ano, face ao ano anterior, por categoria de cliente sobre a base de dados original.
dbo.viewYearGrowthPerSale	Esta view permite o cálculo para as vendas da taxa de crescimento de cada ano, face ao ano anterior, por categoria de cliente sobre a base de dados otimizada.
dbo.viewNProductsPerColor_OldData	Esta view permite o nº de produtos nas vendas por cor sobre a base de dados original.

dbo.viewNProductsPerColor	Esta view permite obter o nº de produtos nas vendas por cor sobre a base de dados otimizada.

9.2 Índices

Designação	Tabela	Justificação/Consultas
_dta_index_City_6_645577338__K1_K2	RH.City	Consultas: dbo.viewSalesPerCity. Esta consulta foi otimizada com este índice, pois esta acede às colunas CitId e CitName da tabela RH.City.
_dta_index_Sale_6_1365579903__K2_K3_8	Sales.Sale	Consultas: dbo.viewSalesPerCity, dbo.viewYearGrowthPerSale. Estas consultas foram otimizadas com este índice, pois ambas acedem às colunas SalCustomerId e SalEmployeeId da tabela Sales.Sale.
_dta_index_ProductPromotion_Sale_6_1429580131__K1	Sales.ProductPromotion_Sale	Consultas: dbo.viewNProductsPerColor Esta consulta foi otimizada com este índice, pois esta acede à coluna ProdProm_SalProductPromotionId da tabela Sales.ProductPromotion_Sale.

9.3 Otimização e Execução de Consultas

Os resultados da execução das consultas foram os seguintes:

- dbo.viewSalesPerCity_OldData sobre a base de dados original (não normalizada);
  - SQL Server Profiler

CPU – 3321

Reads – 1499966

Writes – 3216

Duration – 2442

  - Database Engine TurningAdvisor

Tab Reports

Tuning Summary	
Date	20/01/2023
Time	14:42:34
Server	DESKTOP-NMMQKD0
Database(s) to tune	[WWIGlobal]
Workload file	C:\Users\nunor\Desktop\CBD\2ªL_EI-SW-08_202000753\Projeto.trc
Maximum tuning time	58 Minutes
Time taken for tuning	1 Minute
Estimated percentage improvement	81.19
Maximum space for recommendation (MB)	242
Space used currently (MB)	86
Space used by recommendation (MB)	90
Number of events in workload	99
Number of events tuned	99
Number of statements tuned	1
Percent SELECT statements in the tuned set	100
Number of indexes recommended to be created	2

- dbo.viewSalesPerCity sobre a base de dados otimizada (normalizada) sem índices;
  - SQL Server Profiler

CPU – 125

Reads – 3752

Writes – 0

Duration – 248

  - Database Engine TurningAdvisor

Tab Reports

Tuning Summary

Date	20/01/2023
Time	15:00:56
Server	DESKTOP-NMMQKD0
Database(s) to tune	[WWIGlobal]
Workload file	C:\Users\nunor\Desktop\CBD\2ªL_EI-SW-08_202000753\Projeto.trc
Maximum tuning time	59 Minutes
Time taken for tuning	1 Minute
Estimated percentage improvement	27.57
Maximum space for recommendation (MB)	242
Space used currently (MB)	86
Space used by recommendation (MB)	86
Number of events in workload	77
Number of events tuned	77
Number of statements tuned	1
Percent SELECT statements in the tuned set	100
Number of indexes recommended to be created	2

Tab Recommendations

Index Recommendations

Database Name	Object Name	Recommendation	Target of Recommendation	Details	Partition Scheme	Size (KB)	Definition
WWIGlobal	[RH].[City]	create	_dta_index_City_6_645577338__K1_K2			496	([CtId] asc, [CtName] asc)
WWIGlobal	[Sales].[Sale]	create	_dta_index_Sale_6_1365579903__K2_K3_8			2632	([SalCustomerId] asc, [SalEmployeeId] asc) include ([SalTotalPrice])

- dbo.viewSalesPerCity sobre a base de dados otimizada (normalizada) com índices.
  - SQL Server Profiler

CPU – 93

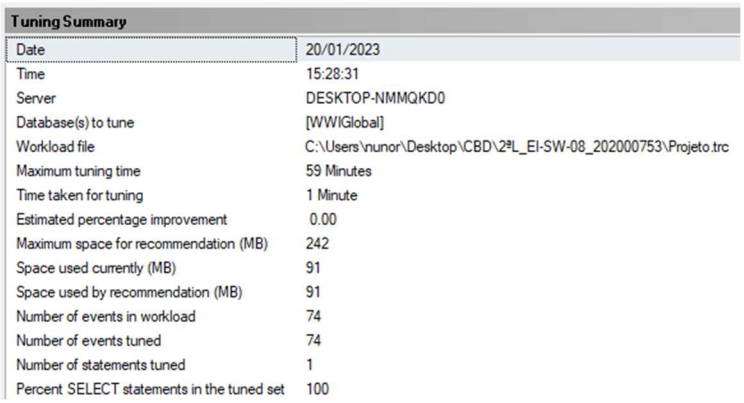
Reads – 1280

Writes – 0

Duration – 165

  - Database Engine TurningAdvisor

Tab Reports



Tuning Summary	
Date	20/01/2023
Time	15:28:31
Server	DESKTOP-NMMQKD0
Database(s) to tune	[WWIGlobal]
Workload file	C:\Users\nunor\Desktop\CBD\2ªL_EI-SW-08_202000753\Projeto.trc
Maximum tuning time	59 Minutes
Time taken for tuning	1 Minute
Estimated percentage improvement	0.00
Maximum space for recommendation (MB)	242
Space used currently (MB)	91
Space used by recommendation (MB)	91
Number of events in workload	74
Number of events tuned	74
Number of statements tuned	1
Percent SELECT statements in the tuned set	100

- dbo.viewYearGrowthPerSale _OldData sobre a base de dados original (não normalizada);
  - SQL Server Profiler

CPU – 30

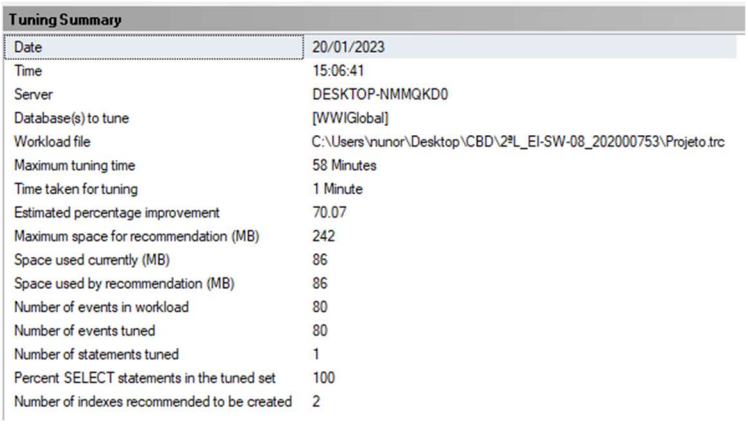
Reads – 6204

Writes – 0

Duration – 22

  - Database Engine TurningAdvisor

Tab Reports



Tuning Summary	
Date	20/01/2023
Time	15:06:41
Server	DESKTOP-NMMQKD0
Database(s) to tune	[WWIGlobal]
Workload file	C:\Users\nunor\Desktop\CBD\2ªL_EI-SW-08_202000753\Projeto.trc
Maximum tuning time	58 Minutes
Time taken for tuning	1 Minute
Estimated percentage improvement	70.07
Maximum space for recommendation (MB)	242
Space used currently (MB)	86
Space used by recommendation (MB)	86
Number of events in workload	80
Number of events tuned	80
Number of statements tuned	1
Percent SELECT statements in the tuned set	100
Number of indexes recommended to be created	2

- dbo.viewYearGrowthPerSale sobre a base de dados otimizada (normalizada) sem índices;
  - SQL Server Profiler

CPU – 109

Reads – 1894

Writes – 0

Duration – 172

- Database Engine TurningAdvisor

Tab Reports

Tuning Summary	
Date	20/01/2023
Time	15:11:21
Server	DESKTOP-NMMQKD0
Database(s) to tune	[WWIGlobal]
Workload file	C:\Users\nunor\Desktop\CBD\2ªL_EI-SW-08_202000753\Projeto.trc
Maximum tuning time	58 Minutes
Time taken for tuning	1 Minute
Estimated percentage improvement	27.57
Maximum space for recommendation (MB)	242
Space used currently (MB)	86
Space used by recommendation (MB)	86
Number of events in workload	80
Number of events tuned	80
Number of statements tuned	1
Percent SELECT statements in the tuned set	100
Number of indexes recommended to be created	2

Tab Recommendations

Index Recommendations

Database Name	Object Name	Recommendation	Target of Recommendation	Details	Partition Scheme	Size (KB)	Definition
WWIGlobal	[RH].[City]	create	_dta_index_City_6_645577338__K1_K2			496	([CityId] asc, [CityName] asc)
WWIGlobal	[Sales].[Sale]	create	_dta_index_Sale_6_1365579903__K2_K3_8			2632	([SaleCustomerId] asc, [SaleEmployeeId] asc) include ([SaleTotalPrice])

- dbo.viewYearGrowthPerSale sobre a base de dados otimizada (normalizada) com índices.
  - SQL Server Profiler

CPU – 31

Reads – 1102

Writes – 0

Duration – 57

- Database Engine TurningAdvisor

Tab Reports

Tuning Summary	
Date	20/01/2023
Time	15:30:46
Server	DESKTOP-NMMQKD0
Database(s) to tune	[WWIGlobal]
Workload file	C:\Users\nunor\Desktop\CBD\2ªL_EI-SW-08_202000753\Projeto.trc
Maximum tuning time	59 Minutes
Time taken for tuning	1 Minute
Estimated percentage improvement	42.75
Maximum space for recommendation (MB)	242
Space used currently (MB)	91
Space used by recommendation (MB)	90
Number of events in workload	78
Number of events tuned	78
Number of statements tuned	1
Percent SELECT statements in the tuned set	100
Number of indexes recommended to be created	1

- dbo.viewNProductsPerColor _OldData sobre a base de dados original (não normalizada);
  - SQL Server Profiler

CPU – 0

Reads – 6207

Writes – 0

Duration – 21

  - Database Engine TurningAdvisor

Tab Reports

Tuning Summary	
Date	20/01/2023
Time	15:17:28
Server	DESKTOP-NMMQKD0
Database(s) to tune	[WWIGlobal]
Workload file	C:\Users\nunor\Desktop\CBD\2ªL_EI-SW-08_202000753\Projeto.trc
Maximum tuning time	59 Minutes
Time taken for tuning	1 Minute
Estimated percentage improvement	86.28
Maximum space for recommendation (MB)	242
Space used currently (MB)	86
Space used by recommendation (MB)	86
Number of events in workload	75
Number of events tuned	75
Number of statements tuned	1
Percent SELECT statements in the tuned set	100
Number of indexes recommended to be created	2

- dbo.viewNProductsPerColor sobre a base de dados otimizada (normalizada) sem índices;
  - SQL Server Profiler

CPU – 15

Reads – 978

Writes – 0

Duration – 53

  - Database Engine TurningAdvisor

Tab Reports

Tuning Summary	
Date	20/01/2023
Time	15:19:19
Server	DESKTOP-NMMQKD0
Database(s) to tune	[WWIGlobal]
Workload file	C:\Users\nunor\Desktop\CBD\2ªL_EI-SW-08_202000753\Projeto.trc
Maximum tuning time	59 Minutes
Time taken for tuning	1 Minute
Estimated percentage improvement	22.80
Maximum space for recommendation (MB)	242
Space used currently (MB)	86
Space used by recommendation (MB)	88
Number of events in workload	79
Number of events tuned	79
Number of statements tuned	1
Percent SELECT statements in the tuned set	100
Number of indexes recommended to be created	1

Tab Recommendations

Index Recommendations

Database Name	Object Name	Recommendation	Target of Recommendation	Details	Partition Scheme	Size (KB)	Definition
WWIGlobal	[Sales].[ProductPromotion_Sale]	create	_dta_index_ProductPromotion_Sale_6_1429580131__K1			4720	([ProdProm_SalProductPromotionId1 asc)

- dbo.viewNProductsPerColor sobre a base de dados otimizada (normalizada) com índices.
  - SQL Server Profiler

CPU – 15

Reads – 473

Writes – 0

Duration – 20

  - Database Engine TurningAdvisor

Tab Reports

Tuning Summary	
Date	20/01/2023
Time	15:32:59
Server	DESKTOP-NMMQKD0
Database(s) to tune	[WWIGlobal]
Workload file	C:\Users\nunor\Desktop\CBD\2ªL_EI-SW-08_202000753\Projeto.trc
Maximum tuning time	59 Minutes
Time taken for tuning	1 Minute
Estimated percentage improvement	0.00
Maximum space for recommendation (MB)	242
Space used currently (MB)	91
Space used by recommendation (MB)	91
Number of events in workload	80
Number of events tuned	80
Number of statements tuned	1
Percent SELECT statements in the tuned set	100

10. Backup e Recuperação

Optei por utilizar o modelo de recuperação Full, apesar de mais dispendioso em espaço necessário e menor desempenho tem uma proteção mais elevada contra a perda de informação.

O tipo de backups que decidi usar é Backups completos de sete em sete dias e Backups diferenciais a cada vinte e quatro horas.

Pode ser necessário recuperar apenas os dados inseridos ou alterados depois do fim da semana, neste caso a recuperação seria feita através do backup diferencial. Caso fossem perdidos todos os dados a recuperação seria feita usando o ficheiro do backup completo e o ficheiro do backup diferencial.

11. Segurança e Controlo de Acessos

11.1 Níveis de acesso à informação

Roles:

- `administrador` - tem acesso a toda a informação.
- employeeSalesPerson - tem acesso total às tabelas de suporte às vendas, e apenas acesso em modo de consulta às restantes tabelas.
- salesTerritory - Apenas pode consultar a informação relativa ao seu território (Rocky Mountain).

Utilizadores:

- Adminis - tem como credenciais de acesso login: “Admi”; password = “PASSWORD”; Tem o role “administrador”.
- EmpSalPerson - tem como credenciais de acesso login: “EmployeeSales”; password = “PASSWORD”; Tem o role “employeeSalesPerson”.
- SalTerri - tem como credenciais de acesso login: “SalesTer”; password = “PASSWORD”; Tem o role “salesTerritory”.

11.2 Encriptação

A nível de encriptação, os campos de password da tabela RH.SysUser são guardadas usando a encriptação com chaves assimétricas (SHA-1), apesar de diminuírem a performance são mais complexas e seguras.

12. MongoDB

Primeiro foi criada a base de dados WWWIWeb em MongoDB depois foram criadas as consultas necessárias para popular a base de dados anterior (estas consultas estão no ficheiro *MongoDB.sql*), através destas consultas foram exportados os dados para ficheiros .csv (presentes na pasta MongoDB) e por último foram importados os dados desses ficheiros para a base de dados WWWIWeb.

12.1 Coleções

BuyingGroup:

- BuyGrouId (number)
- BuyGrouName (string)

Brand:

- BraId (number)
- BraName (string)

Customer:

- CusUserId (number)
- CusHeadquartersId (number)
- CusRegion_CategoryId (number)
- CusBuyingGroupId (number)
- CusPrimaryContact (string)

Product:

- ProdId (number)
- ProdBrandId (number)
- ProdTaxRateId (number)
- ProdProductTypeId (number)
- ProdBuyingPakageId (number)
- ProdSellingPakageId (number)
- ProdName (string)
- ProdColor (string)
- ProdSize (string)
- ProdLeadTimesDays (number)
- ProdQuantityPerOuter (number)
- ProdStock (number)
- ProdBarCode (number)
- ProdUnitPrice (double)
- ProdRecommendedRetailPrice (double)
- ProdTypicalWeightPerUnit (double)

Product_Promotion:

- Prod_PromProductPromotionId (number)
- Prod_PromProductId (number)
- Prod_PromPromotionId (number)
- ProdNewPrice (string)

ProductPromotion_Sale:

- ProdProm_SalProductPromotionId (number)
- ProdProm_SalSaleId (number)

- ProdProm_SalQuantity (number)

Promotion:

- PromId (number)
- PromDescription (string)
- PromStartDate (date)
- PromEndDate (date)

Sale:

- SalID (number)
- SalCustomerId (number)
- SalEmployeeId (number)
- SalDate (date)
- SalDeliveryDate (date)
- SalDescription (string)
- SalProfit (double)
- SalTotalPrice (double)
- SalTotalExcludingTax (double)
- SalTaxAmount (double)
- SalIsFinished (boolean)

SysUser

- SysUseId (number)
- SysUseEmail (string)
- SysUsePassword (string)
- SysUseName (string)

13. Descrição da Demonstração

13.1 Script de demonstração sobre a base de dados relacional

O primeiro passo é criar a base de dados para tal temos de executar o ficheiro *CriacaoBD.sql* (cria a base de dados WWWIGlobal, schemas e tabelas), podemos também executar os ficheiros *Functions.sql*, *Geradores.sql*, *Triggers.sql*, *StoredProcedures.sql* e *Views.sql* para criar as funções, stored procedures geradores, triggers, stored procedures e views.

Para a importação dos dados da base de dados antiga para a nova teremos de realizar os seguintes passos:

- Executar todos os ficheiros da pasta Scripts de migração (cria e importa as tabelas e registos da base de dados antiga WWW_DS), a ordem da execução dos ficheiros apenas importa nos ficheiros Sale…, a ordem tem de ser *Sale1.sql*, *Sale2.sql*, *Sale3.sql* e por fim *Sale4.sql*.
- Para verificar que as tabelas foram bem criadas e os registos foram bem importados podemos executar as queries no final do ficheiro *WWWGlobal Query.sql* (apenas as queries associadas ao schema OldData).
- Por fim basta executar o ficheiro *OldDataMigration.sql* para que todos os registos sejam tratados e inseridos nas novas tabelas.

- Podemos também conferir se os registos foram bem tratados e inseridos executando as queries dos ficheiros do *WWWGlobal Query.sql* (queries associadas aos schemas RH, Storage e Sales).

Depois disso basta executar as funcionalidades que pretendemos testar (a forma como se executa todas as funcionalidades está nos ficheiros, no ficheiro *SQLQuery2.sql* são apresentadas algumas das mais importantes).

Para gerar e guardar as informações relativas á monitorização e espaço ocupado pelas tabelas e colunas precisamos de executar o ficheiro *monitorização.sql*.

13.2 Script de demonstração sobre a base de dados NoSQL

O primeiro passo é criar a base de dados WWWIWeb no MongoDB Compass, depois criar as coleções indicadas no ponto 13.1. Para importar os dados do MSSQL basta fazer a importação para as coleções através dos ficheiros presentes na pasta MongoDB.

Para verificar o correto funcionamento da base de dados NoSQL basta realizar as consultas do ponto 13.2.

14. Conclusões

Com a realização deste projeto fiquei mais familiarizado com a administração de uma base de dados relacional, apesar de não ter conseguido implementar tudo o que era proposto e ambicionei. Consegui aplicar o conhecimento que adquiri em anos anteriores na disciplina Bases de Dados e no ano atual em Complementos de Base de Dados.

Fazer este projeto sozinho foi muito desafiador pois tive de desenvolver todo o projeto apenas com a ajuda do professor e a pesquisa na internet.