IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Complementos de Bases de Dados

2022/2023

Licenciatura em Engª. Informática

1ª Fase Relatório Técnico

Turma: 2ºL EI-SW-01
Horário de Laboratório: 5ªf 17h00
Docente: João Portelinha
Grupo
Nº202100225, André Meseiro
Nº202100230, Pedro Anjos

1. Introdução

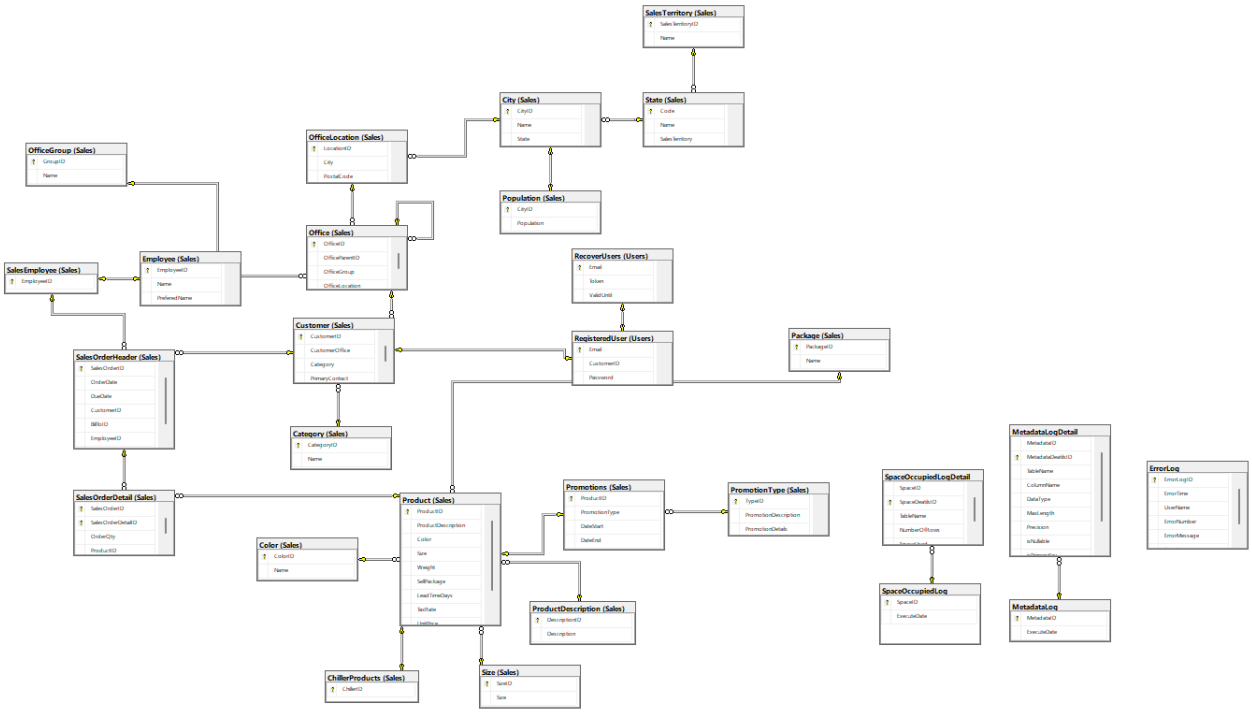
O projeto final da disciplina de Complementos de Bases de Dados tem como objetivo a familiarização com a parte administrativa de bases de dados relacionais. O trabalho irá incidir tanto ao nível da base de dados como do sistema gestor de bases de dados e também sobre a lógica subjacente à utilização dos dados disponibilizados, ou seja, a partir de uma base de dados não normalizada e um conjunto de ficheiros em formato de texto e Excel (desatualizados, portanto) de uma empresa com o nome "World Wide Importers" (WWI), reformular este sistema para uma nova base de dados, de forma integrada, com todo o processo de vendas desta empresa.

2. Especificação de Requisitos

ID	Descrição	Implementado (S/N)
R01	*Criação da nova base de dados normalizada*	*S*
R02	*Migração dos dados da base de dados antiga para a nova*	*S*
R03	*Consultas de verificação/validação da migração (requesitos mínimos)*	*S*
R04	*Gestão de vendas utilizando SPs, UDFs, e Triggers para os processos de negócio propostos*	*S*
R05	*SPs geradores e monitorização (metadados)*	*S*
R06	*Testes*	*S*
R07	*Tratamento de erros*	*S*
R08	*Definição do layout da nova base de dados*	*S*
RM01	*Existência de utilizadores de acesso à aplicação, para os clientes*	*S*
RM02	*Gestão de utilizadores*	*S*
RM03	*Gestão de promoções, para os produtos*	*S*

3. Modelo Relacional (*Modelo de dados*)

3.1 Diagrama do Modelo Relacional



4. Definição do Layout

O layout proposto para a nova base de dados, considerando todas as tabelas criadas e o tipo de ficheiros existentes é o seguinte (assumir taxa de crescimento para 1 ano, respetiva para cada FG definida abaixo na tabela):

- Primary FileGroup: WWIGlobalDat com um primary file wwiglobaldat.mdf;

- LogFG WWIGlobalLog com log file wwigloballog.ldf;

- UFG1: ProductDat – ChillerProducts + Product + ProductDescription + Promotions + PromotionType (tudo o que é relacionado com produtos) wwiglobalproduct.ndf;

- UFG2: SaleDat – SalesEmployee + SalesOrderDetail + SalesOrderHeader + SalesTerritory (tudo o que é relacionado com produtos) wwiglobalsale.ndf

- UFG3: LocDat - City + State + Population (tudo o que é relacionado com localização) wwigloballoc.ndf

-- UFG4: OfficeDat - Office + OfficeGroup + OfficeLocation (tudo o que é relacionado com escritórios) wwiglobaloffice.ndf

- UFG5: PersonDat - Customer + Employee (tudo o que é relacionado com pessoas) wwiglobalperson.ndf

- UFG6: MiscDat - Category + Color + Package + Size (tudo o que é informação variada/diversa) wwiglobalmisc.ndf

- UFG7: UserDat - RecoverUsers + RegisteredUser (tudo o que é relacionado com utilizadores) wwiglobaluser.ndf

- UFG8: MetaDat – MetadataLog + MetadataLogDetail + SpaceOccupiedLog + SpaceOccupiedLogDetail (tudo o que é relacionado com metadados) wwiglobalmeta.ndf

-

4.1 Identificação do espaço ocupado por tabela

Nome Tabela	Dimensão do Registo	Nº de Registos (inicial/final)
ErrorLog	*548KB*	*Inicial: 0, final: 25000*
ChillerProducts	*76KB*	*Inicial: 8, final: 10*
Product	*128KB*	*Inicial: 229, final: 270*
ProductDescription	*176KB*	*Inicial: 92, final: 115*
Promotions	*86KB*	*Inicial: 0, final: 10*
PromotionType	*206KB*	*Inicial: 0, final: 3*
SalesEmployee	*76KB*	*Inicial: 10, final: 20*
SalesOrderDetail	*5208KB*	*Inicial: 115383, final: 120383*
SalesOrderHeader	*2534KB*	*Inicial: 63719, final: 68719*
SalesTerritory	*126KB*	*Inicial: 9, final: 12*
City	*1196KB*	*Inicial: 37940 , final: 38120*
State	*128KB*	*Inicial: 57 , final: 57*
Population	*528KB*	*Inicial: 26878, final: 27005*
Office	*88KB*	*Inicial: 402 , final: 410*

OfficeGroup	*106KB*	*Inicial: 2 , final: 2*
OfficeLocation	*90KB*	*Inicial: 402 , final: 410*
Customer	*124KB*	*Inicial: 402 , final: 410*
Employee	*136KB*	*Inicial: 19 , final: 39*
Category	*96KB*	*Inicial: 5 , final: 5*
Color	*106KB*	*Inicial: 9 , final: 9*
Package	*106KB*	*Inicial: 5 , final: 5*
Size	*116KB*	*Inicial: 44 , final: 44*
RecoverUsers	*230KB*	*Inicial: 0 , final: 17*
RegisteredUsers	*298KB*	*Inicial: 0 , final: 103*
MetadataLog	*84KB*	*Inicial: 0 , final: 1*
MetadataLogDetail	*211KB*	*Inicial: 0 , final: 72*
SpaceOccupiedLog	*84KB*	*Inicial: 0, final: 1*
SpaceOccupiedLogDetail	*138KB*	*Inicial: 0, final: 23*

4.2 Especificação dos Filegroups

Nome Filegroup	Tabelas associadas	Parâmetros
WWIGlobalDat	----------	*Dimensão inicial: 100MB, dimensão final: 200MB, taxa de crescimento: 50MB*

WWIGlobalLog	----------	*Dimensão inicial: 1MB, dimensão final: 15MB, taxa de crescimento: 14MB*
ProductDat	*ChillerProducts, Product, ProductDescription, Promotions, PromotionType*	*Dimensão inicial: 5MB, dimensão final: 6MB, taxa de crescimento: 1MB*
SaleDat	*SalesEmployee, SalesOrderDetail, SalesOrderHeader, SalesTerritory*	*Dimensão inicial: 10MB, dimensão final: 15MB, taxa de crescimento: 5MB*
LocDat	*City, State, Population*	*Dimensão inicial: 5MB, dimensão final: 6MB, taxa de crescimento: 1MB*
OfficeDat	*Office, OfficeGroup, OfficeLocation*	*Dimensão inicial: 2MB, dimensão final: 3MB, taxa de crescimento: 1MB*
PersonDat	*Customer, Employee*	*Dimensão inicial: 1MB, dimensão final: 4MB, taxa de crescimento: 3MB*
MiscDat	*Category, Color, Package, Size*	*Dimensão inicial: 1MB, dimensão final: 1MB, taxa de crescimento: 0MB*
UserDat	*RecoverUsers, RegisteredUsers*	*Dimensão inicial: 1MB, dimensão final: 5MB, taxa de crescimento: 4MB*
MetaDat	*MetadataLog. MetadataLogDetail, SpaceOccupiedLog, SpaceOccupiedLogDetail*	*Dimensão inicial: 1MB, dimensão final: 1MB, taxa de crescimento: 0MB*

4.3 Schemas

Nome	Descrição
Sales	*Este schema tem como objetivo representar tudo o que está relacionado com as vendas e a sua gestão*

Users	*Este schema tem como objetivo representar tudo o que está relacionado com utilizadores e a sua gestão*

5. Verificação da migração de dados

```
-- Verify Data Migration

-- Verifies the number of customers in both Databases
EXEC Sales.spNumberOfCustomersBothBDs;

-- Verifies the number of customers per category in both Databases
EXEC Sales.spNumberOfCustomersBothBDsPerCategory;

-- Verifies the number of sales per employee in both Databases
EXEC Sales.spNumberOfSalesBothBDsPerEmployee;

-- Verifies the total of sales per stock item in both Databases
EXEC Sales.spTotalValuePerProductBothBDs;

-- Verifies the total of sales per stock item per year in both Databases
EXEC Sales.spGetTotalSalesPerYearPerStockItem;

-- Verifies the number of sales per city per year in both Databases
EXEC spTotalSalesPerYearPerCity;
```

6. Programação

6.1 Views

Nome	Descrição
vCityStates	*Esta view permite obter a lista de cidades e estados*
vStateSalesTerritory	*Esta view permite obter a lista de estados e territórios*
vCustomer	*Esta view permite obter a lista de clientes com a respetiva informação relevante*
vSalesEmployee	*Esta view permite obter a lista de vendas associadas a cada empregado*

vProducts	*Esta view permite obter a lista de produtos com a respetiva informação relevante*
vSalesHeader	*Esta view permite obter a lista com informação relevante das "faturas" das vendas*
vSalesDetails	*Esta view permite obter a lista com os detalhes relevantes das vendas*

6.2 Functions

Nome	Atributos	Requisito	Descrição
Sales.fnCalcTotalPriceSale	*@SalesOrderDetailID INT*	*R04*	*Permite obter o preço total de uma venda*
Sales.fnCheckIfSaleAllowsChillerProducts	*@SalesOrderID INT*	*R04*	*Auxiliar - permite saber se uma venda pode ou não receber um produto do tipo "Chiller"*
Sales.fnGetNumberCustomersPerCategoryOldDataBase	*@Category VARCHAR(40)*	*R03*	*Permite obter o número de clientes, agrupado por categoria, da base de dados antiga*
Sales.fnGetTotalSalesPerStockItemOldDataBase	*@Description VARCHAR(100)*	*R03*	*Permite obter o total de vendas por stock item da base de dados antiga*

6.3 Stored procedures

Nome	Atributos	Requisito	Descrição
Sales.spCreateSale	*@DueDate DATE, @CustomerID INT, @BillToID INT, @EmployeeID INT,*	*R04*	*Permite criar uma nova venda*

	@TaxAmount MONEY		
Sales.spAddProductToSale	*@SalesOrderID INT, @OrderQty INT, @ProductID INT, @UnitPrice MONEY*	*R04*	*Permite adicionar um produto a uma venda*
Sales.spUpdateProductQty	*@SalesOrderDetailID INT, @NewOrderQty INT*	*R04*	*Permite alterar a quantidade de um produto numa venda*
Sales.spRemoveProduct	*@SalesOrderDetailID INT, @DeleteSale BIT*	*R04*	*Permite remover um produto de uma venda, com a opção de também remover a venda caso esta não tenha nenhum produto associado*
Users.spAddUser	*@email VARCHAR(50), @customerID INT, @password VARCHAR(100)*	*RM01*	*Permite adicionar um utilizador à base de dados*
spLogError	----------	*R07*	*Permite adicionar um erro à tabela de log de erros*
Sales.spNumberOfCustomersBothBDs	----------	*R03*	*Permite saber o número de clientes tanto da base de dados antiga como da nova*
Sales.spTotalValuePerProductBothBDs	----------	*R03*	*Permite obter o valor total por produto tanto da base de dados antiga como da nova*
Sales.spNumberOfCustomersBothBDsPerCategory	----------	*R03*	*Permite obter o número de clientes tanto da base de dados antiga como da nova*
Sales.spNumberOfSalesBothDBsPerEmployee	----------	*R03*	*Permite obter o número de vendas agrupado por empregado tanto da base de dados antiga como da nova*

color_migrate	----------	*R02*	*Permite migrar os registos das cores da base de dados antiga para a nova*
state_migrate	----------	*R02*	*Permite migrar os registos dos estados da base de dados antiga para a nova*
package_migrate	----------	*R02*	*Permite migrar os registos dos pacotes da base de dados antiga para a nova*
city_migrate	----------	*R02*	*Permite migrar os registos das cidades da base de dados antiga para a nova*
salesTerritory_migrate	----------	*R02*	*Permite migrar os registos dos territórios das vendas da base de dados antiga para a nova*
category_migrate	----------	*R02*	*Permite migrar os registos das categorias da base de dados antiga para a nova*
customer_migrate	----------	*R02*	*Permite migrar os registos dos clientes da base de dados antiga para a nova*
employee_migrate	----------	*R02*	*Permite migrar os registos das empregados da base de dados antiga para a nova*
product_migrate	----------	*R02*	*Permite migrar os registos dos products da base de dados antiga para a nova*
sale_migrate	----------	*R02*	*Permite migrar os registos das vendas da base de dados antiga para a nova*
Sales.spGetTotalSalesBothBDsPerYearPerStockItem	----------	*R03*	*Permite obter o número total de vendas das duas bases de por ano e por stock item*
Sales.spCreatePromotion	*@productID INT, @promotionDetail VARCHAR(30), @PromotionDescription VARCHAR(100),*	*RM03*	*Permite adicionar uma promoção a um produto*

	@DateStart DATE, @DateEnd DATE		
Users.recoverPassword	*@email VARCHAR(100)*	*RM02*	*Permite enviar um token ao utilizador para este recuperar a sua password*
Users.spCheckTokenValid	*----------*	*RM02*	*Permite verificar os tokens para dar reset à password e remove as que já expiraram*
Users.spEditUser	*@email VARCHAR(50), @propertyToChange VARCHAR(20), @value VARCHAR(100)*	*RM02*	*Permite editar uma propriedade de um utilizador*
Users.spRemoveUser	*@email VARCHAR(50)*	*RM02*	*Permite remover um utilizador*
Sales.spEditPromotionDate	*@productID INT, @startDate DATE, @endDate DATE*	*RM03*	*Permite alterar a data de uma promoção*

7. Catálogo/Metadados

7.1 Geradores

Nome	Atributos	Descrição
spGenInserts	*@tableName VARCHAR(30)*	*Implementa o procedimento para inserir registos numa tabela*
spGenUpdates	*@tableName VARCHAR(30)*	*Implementa o procedimento para inserir updates numa tabela*
spGenDelete	*@tableName VARCHAR(30)*	*Implementa o procedimento para remover registos numa tabela*

7.2 Monitorização

Nome	Atributos	Descrição
vTableMetadata	----------	*Mostra a última execução da procedure spAutoGenerateMetadata*
vSizeOccupied	----------	*View auxiliar para listar o número de registos e espaço ocupado por tabela*
vTableSizeOccupied	----------	*Mostra a última execução da procedure spAutoGenerateSpaceUsed*
dbo.spGenerateMetadataPerTable	*@MetadataID INT, @tableName VARCHAR(50), @schemaName VARCHAR(30)*	*Implementa o procedimento para gerar os metadados para todas as tabelas da base de dados*
dbo.spGenMetadata	*@MetadataID INT*	*Implementa o procedimento para gerar metadados para uma tabela da base de dados*
dbo.spAutoGenerateMetadata	----------	*Implementa o procedimento para gerar de forma automática os metadados para todas as tabelas da base de dados*
spGenerateSpacePerTable	*@SpaceID INT, @tableName VARCHAR(50)*	*Implementa o procedimento para gerar o espaço ocupado por tabela da base de dados*
dbo.spGenSpaceOccupied	*@SpaceID INT*	*Implementa o procedimento para gerar o espaço ocupado por todas as tabelas da base de dados*
dbo.spAutoGenerateSpaceUsed	----------	*Implementa o procedimento para gerar de forma automática o espaço utilizado e o número de registos de todas as tabelas da base de dados*

8. Descrição da Demonstração

8.1 Script de demonstração

```
-- Tests to the implemented procedures and functions in the programming file
USE WWIGlobal;
GO

-- Test procedure to create a sale
EXEC Sales.spCreateSale '2022-11-30', 1, 3, 4, 4.99;
EXEC Sales.spCreateSale '2022-12-20', 1, 3, 4, 5.99;
EXEC Sales.spCreateSale '2022-12-06', 1, 3, 4, 6.99;
EXEC Sales.spCreateSale '2022-12-14', 1, 3, 4, 7.99;

SELECT *
FROM WWIGlobal.Sales.SalesOrderHeader
WHERE TaxAmount = 4.99;

-- to return to original state: delete this OrderHeader

-- Test procedure to add a product to a sale
EXEC Sales.spAddProductToSale 63716, 1, 1;
EXEC Sales.spAddProductToSale 63716, 1, 215;

EXEC Sales.spAddProductToSale 63717, 1, 220;
EXEC Sales.spAddProductToSale 63717, 1, 2;

EXEC Sales.spAddProductToSale 63718, 1, 215;
EXEC Sales.spAddProductToSale 63718, 1, 220;

EXEC Sales.spAddProductToSale 63719, 1, 1;
EXEC Sales.spAddProductToSale 63719, 1, 2;

SELECT *
FROM WWIGlobal.Sales.SalesOrderDetail
WHERE SalesOrderID = 63716

-- Check for errors
SELECT *
FROM dbo.ErrorLog;

-- to return to original state: delete this OrderDetail
```

```
-- Test procedure to update product quantity on a sale
select *
FROM WWIGlobal.Sales.SalesOrderDetail
WHERE SalesOrderDetailID = 115378;

EXEC Sales.spUpdateProductQty 115378, 2;

select *
FROM WWIGlobal.Sales.SalesOrderDetail
WHERE SalesOrderDetailID = 115378;

-- to return to original state: execute the same procedure using 1 instead of 2
```

```
-- Test procedure to remove product
SELECT *
FROM WWIGlobal.Sales.SalesOrderHeader
WHERE SalesOrderID = 63718;

EXEC Sales.spRemoveProduct 115378, 1;

SELECT *
FROM WWIGlobal.Sales.SalesOrderHeader
WHERE SalesOrderID = 63718;
```

```
-- Test function to calculate the total price of a sale
SELECT Sales.fnCalcTotalPriceSale(115375) 'Total';
```

```
-- Test add Promotion To Product Sale
EXEC Sales.spUpdateProductQty 115380, 7;
EXEC Sales.spUpdateProductQty 115381, 3;
SELECT sd.SalesOrderID, sd.SalesOrderDetailID, p.ProductID , sd.OrderQty, p.UnitPrice, Sales.fnCalcTotalPriceSale(sd.SalesOrderDetailID) SubTotal
FROM WWIGlobal.Sales.SalesOrderDetail sd
    JOIN WWIGlobal.Sales.Product p on p.ProductID=sd.ProductID
WHERE SalesOrderID = 63718;
```

```
-- Create promotions
SELECT sd.SalesOrderID, sd.SalesOrderDetailID, p.ProductID , sd.OrderQty, p.UnitPrice, Sales.fnCalcTotalPriceSale(sd.SalesOrderDetailID) SubTotal
FROM WWIGlobal.Sales.SalesOrderDetail sd
    JOIN WWIGlobal.Sales.Product p on p.ProductID=sd.ProductID
WHERE SalesOrderID = 63718;

EXEC Sales.spCreatePromotion 215,'50','50% de Desconto','2022-11-20','2022-11-30'
EXEC Sales.spCreatePromotion 220,'100','100% de Desconto','2022-11-29','2022-11-30'

SELECT p.ProductID, pt.PromotionDescription, p.DateStart, p.DateEnd
FROM WWIGlobal.Sales.Promotions p
    JOIN WWIGlobal.Sales.PromotionType pt on pt.TypeID=p.PromotionType

SELECT sd.SalesOrderID, sd.SalesOrderDetailID, p.ProductID , sd.OrderQty, p.UnitPrice, Sales.fnCalcTotalPriceSale(sd.SalesOrderDetailID) SubTotal
FROM WWIGlobal.Sales.SalesOrderDetail sd
    JOIN WWIGlobal.Sales.Product p on p.ProductID=sd.ProductID
WHERE SalesOrderID = 63718;

EXEC Sales.spEditPromotionDate 215,'2022-11-29','2022-11-30'

SELECT p.ProductID, pt.PromotionDescription, p.DateStart, p.DateEnd
FROM WWIGlobal.Sales.Promotions p
    JOIN WWIGlobal.Sales.PromotionType pt on pt.TypeID=p.PromotionType
```

```
SELECT sd.SalesOrderID, sd.SalesOrderDetailID, p.ProductID , sd.OrderQty, p.UnitPrice, Sales.fnCalcTotalPriceSale(sd.SalesOrderDetailID) SubTotal
FROM WWIGlobal.Sales.SalesOrderDetail sd
    JOIN WWIGlobal.Sales.Product p on p.ProductID=sd.ProductID
WHERE SalesOrderID = 63717;
```

```
-- Test user insert
DECLARE @counter INT
SET @counter = 1
while @counter <=10
BEGIN
    DECLARE @email VARCHAR(50)
    SET @email = CONCAT('user',@counter,'@email.com')
    DECLARE @password VARCHAR(50)
    SET @password = NEWID()
    exec Users.spAddUser @email,@counter,@password;
    Set @counter=@counter+1
END

SELECT *
FROM Users.RegisteredUser

EXEC Users.spEditUser 'user1@email.com','customerid','21'
--Recover a password from a user
EXEC Users.recoverPassword 'user1@email.com'
SELECT *
FROM WWIGlobal.Users.RecoverUsers
EXEC Users.spRemoveUser 'user1@email.com'
```

```
select *
from Sales.SalesOrderDetail

select Sales.fnCheckIfSaleAllowsChillerProducts(1) 'check'
```

```
-- Test accordance of delivery date to lead time days of product
SELECT sd.ProductID, p.LeadTimeDays, sh.OrderDate, sh.DueDate,Sales.fnVerifyDeliveryDate(sd.SalesOrderID) 'Accordance'
FROM WWIGlobal.Sales.SalesOrderDetail sd
    JOIN WWIGlobal.Sales.SalesOrderHeader sh on sh.SalesOrderID=sd.SalesOrderID
    JOIN WWIGlobal.Sales.Product p on p.ProductID=sd.ProductID
WHERE sd.SalesOrderID=63717

SELECT sd.ProductID, p.LeadTimeDays, sh.OrderDate, sh.DueDate,Sales.fnVerifyDeliveryDate(sd.SalesOrderID) 'Accordance'
FROM WWIGlobal.Sales.SalesOrderDetail sd
    JOIN WWIGlobal.Sales.SalesOrderHeader sh on sh.SalesOrderID=sd.SalesOrderID
    JOIN WWIGlobal.Sales.Product p on p.ProductID=sd.ProductID
WHERE sd.SalesOrderID=510
```

```
-- Executes the metadata sps
EXEC spAutoGenerateMetadata

SELECT *
FROM vTableMetadata v
ORDER BY v.[Table Name],v.[Column Name]

EXEC spAutoGenerateSpaceUsed

SELECT *
FROM vTableSizeOccupied v
ORDER BY v.[Number Of Rows] DESC, v.[Table Name]
```

```
-- Test Generate DML for RegisteredUser
EXEC spGenInserts 'RegisteredUser'
EXEC spGenUpdates 'RegisteredUser'
EXEC spGenDelete 'RegisteredUser'

EXEC registereduser_insert 'newEmail@email.com',250,'newPassword'
SELECT *
FROM Users.RegisteredUser
EXEC registereduser_update 'newEmail@email.com',137,'newNewPassword'
SELECT *
FROM Users.RegisteredUser
EXEC registereduser_delete 'newEmail@email.com'
SELECT *
FROM Users.RegisteredUser
```

9. Conclusões

Após ter sido implementada esta primeira fase do projeto, podemos concluir que é de grande importância ter uma base de dados normalizada, sem ficheiros externos de armazenamento de informação (como era o caso da base de dados que nos foi fornecida, que tinha tanto um ficheiro de texto, como um ficheiro Excel), pois torna-se extremamente mais fácil ter tudo armazenado numa base de dados, uma vez que acelera o processo de gestão da informação. Por outro lado, pensamos ter atingido o objetivo pretendido na sua grande maioria, e as partes em que tivemos mais dúvidas/achámos mais difíceis de implementar foram o layout e os metadados.