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: 6
Horário de Laboratório: 6ªfeira - 16:30
Docente: Sara Batista
Grupo
Nº 202002203, Fernando Ramalho
Nº 202001870, Guilherme Bernardino

Índice

1. Introdução

Este projeto tem como objetivo satisfazer a avaliação da cadeira Complementos Base de Dados, do curso Engenheira de Software, da Escola Superior de Tecnologias no Instituto Politécnico de Setúbal. Para esse fim, foi pedido aos alunos que desenvolvessem e que administrassem uma nova base de dados para a empresa denominada Wide World Importers (WWI), uma empresa importadora e distribuidora de produtos, que opera no mercado de vendas e retalho.

Com suporte de ficheiros Excel, texto e a antiga base de dados da WWI, os alunos foram desafiados a desenvolver uma nova base de dados, atualizada, normalizada e escalável.

Na primeira fase, o projeto consistia da projeção da nova tabela, com especial atenção à normalização e otimização de dados, através de uma modelagem acertiva da base de dados, com tabelas destinadas a certas funções, e ainda tabelas com novas funcionalidades. Criação de schemas, um layout, funcionalidades de utilização da base de dados como migração de dados, verificação através de consultas, criação de Storage Procedures, Functions, Views e Triggers.

Na segunda fase deste projeto foram adicionadas alterações e melhorias, assim como mais algumas funcionalidades. A potêncial melhoria da base de dados, tal como a introdução de novos schemas para separar grupos de tabelas em relação ao tipo de dados trabalhados, é um dos exemplos.

Foi proposto para a segunda fase e eventual finalização do projeto, o grupo resolver um plano de backups e recuperação da nova base de dados, indexação e otimização de consultas, níveis de segurança, tais como criação de utilizadores com vários níveis de permissões e ainda encriptação de campos de tabelas e transações.

Ao longo deste documento vão ser relatadas as implementações nesta nova base de dados.

2. Especificação de Requisitos

ID	Descrição	Implementado (S/N)
R01	O sistema deve permitir a autenticação com recurso à conta de email e password.	S
R02	O sistema deve permitir adicionar um utilizador.	S
R03	O sistema deve permitir atualizar um utilizador.	S
R04	O sistema deve permitir remover um utilizador.	S
R05	O sistema deve permitir recuperar um utilizador através de um *token* que permite alterar a password.	S
R06	O sistema deve permitir definir uma promoção sobre um ou mais produtos.	S
R07	O sistema deve permitir alterar as datas de início e fim de uma promoção.	S

R08	O sistema deve permitir criar uma venda.	S
R09	O sistema deve permitir adicionar um produto a uma venda.	S
R10	O sistema deve permitir alterar a quantidade de um produto numa venda.	S
R11	O sistema deve permitir remover um produto de uma venda.	S
R12	O sistema deve permitir verificar se uma venda contém 0 produtos associados para se poder remover.	S
R13	O sistema deve permitir calcular o preço total de uma venda.	S
R14	O sistema deve permitir verificar se a data de entrega está de acordo com o tempo previsto de entrega de um produto (“Lead Time Days”).	S
R15	O sistema não deve permitir uma venda conter produtos com e sem “Chiller Stock”.	S
RM01	O sistema deve verificar na inserção da tabela de utilizadores se já existe um utilizador com um email duplicado.	S
RM02	O sistema deve permitir encriptar as passwords de utilizadores.	S

3. Alterações/Melhorias ao Relatório da 1ª Fase

O projeto foi alvo de melhorias e alterações, numa maior parte, no modelo e no layout, visto que consistia em alguns problemas relacionados com relacionamentos de tabelas e a não existência de schemas que separacem as tabelas por tipo de dados e funcionalidades. Um dos maiores problemas também seria a migração, ou neste caso, a falta de migração em certas tabelas, nomeadamente as tabelas Continent, State, Country, entre outros. Por fim a melhor organização de tabelas, dados e funcionalidades ajudou definitivamente na melhoria geral do projeto.

3.1. Lista de Melhorias / Alterações

No modelo:

- Colocar uma chave de discountId na tabela Product e não o inverso, pois um produto vai ter os discontos associados, não o inverso.
- Eliminar redundâncias em relação a dados da tabela Person, i.e., distinção de Customer VS Employee (isSalePerson, PersonType, isActive). Possou a ter apenas a chave estrangeira da tabelas PersonType.
- Invoice (fatura) ser associado apenas a uma venda e não um customer / city (eliminação da InvoiceCustomer).
- Tabela States e Countries, relação alterada de um-para-muitos para muitos-para-muitos (criação do CountryState).
- Alteração da tabela Sale (faz referência à Invoice através de FK; faz referência à Customer através de FK e faz referência à City através de FK).

No layout:

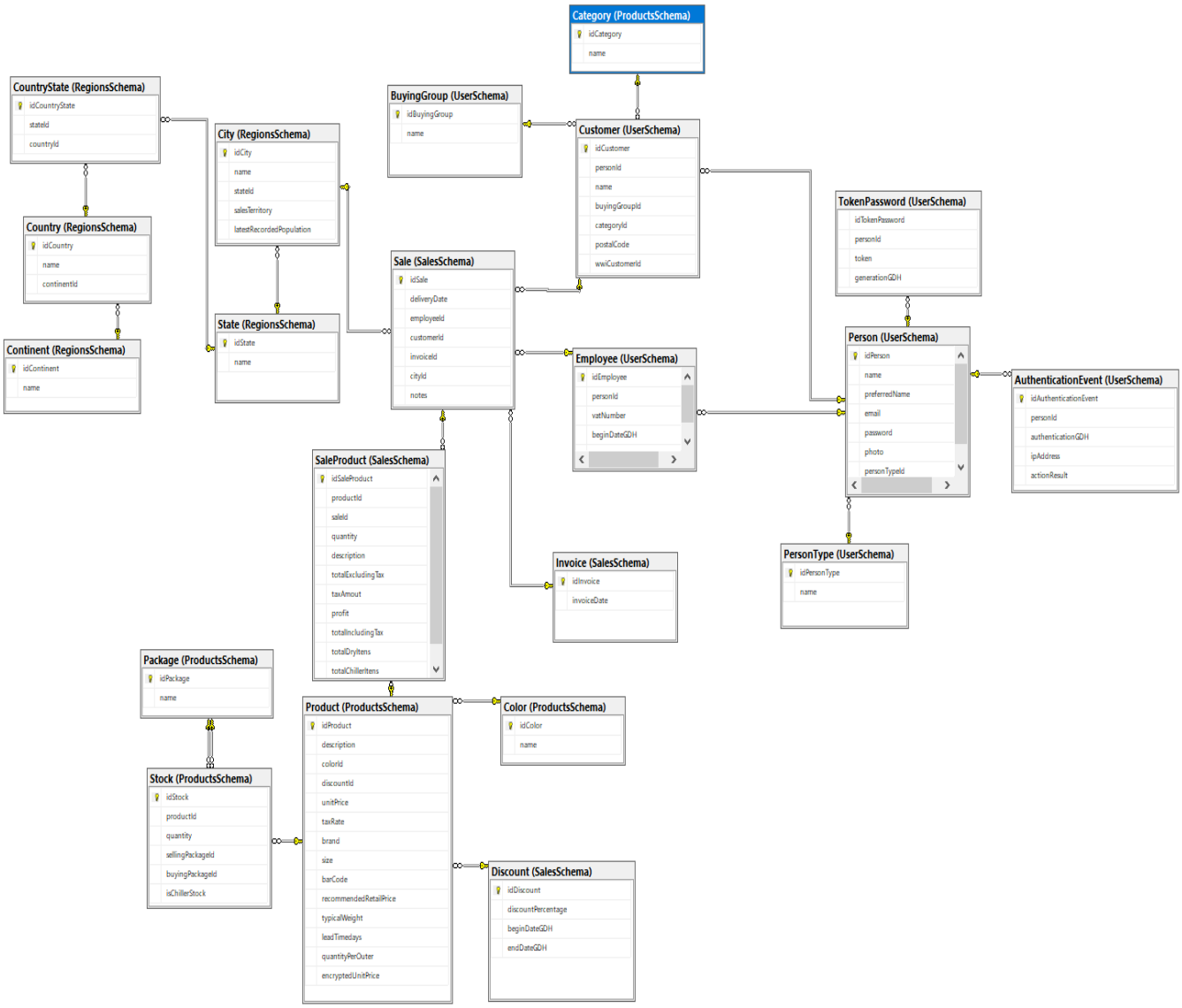
- Criar schemas para ajudar a separar grupos.
- FileGroup atualizados e agora incluiem todas as tabelas.

Na programação:

- Melhorar a parte de user (autenticação, mudança de password).
- Migração para tabelas Category, Color, CountryState e Package (alteração das primary keys para auto-incrementáveis quando inseridas).
- Tabelas Country e Continent incluem migração da DB antiga.
- Inserção, migração e alteração da tabela Stock (não contém colorId e quantity pode ser null).
- Inserção em bulk (deixou de ser manual), da tabela State através do ficheiro "states.txt".
- Storage Procedure Person_Into agora utiliza a function EncryptPassword para criar uma hashedPassword.
- Maior parte das funções, views, storage procedures e triggers foram alterados para corresponder às mudanças descritas anteriormente.

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 (Bites)	Nº de Registos
AuthenticationEvent	8	0
BuyingGroup	36 000	2
Category	15 000	5
City	60	116294
Color	800	9
Continent	1 000	7
Country	60	249
Customer	100	402
Discount	8 000	1
Employee	30	212
Invoice	20	70510
CountryState	20	228265
Package	44 564 000	6
Person	80	615
PersonType	4 000	2
Product	100	671
Sale	100	228266
SaleProduct	100	228266
State	100	57
Stock	8	0
TokenPassword	8	0

5.2 Especificação dos Filegroups

Nome Filegroup	Tabelas associadas	Parâmetros
ProductFileGroup	Stock, Package, Color, Product, Category, Package	20 MB / 400 MB / 5 %
PersonFileGroup	Person, AuthenticationEvent, PersonType, TokenPassword, Employee, Customer, BuyingGroup	30 MB / 600 MB / 5 %
SaleFileGroup	Discount, Sale, SaleProduct, Invoice	25 MB / 500 MB / 5 %
RegionFileGroup	City, Continent, Country, State, CountryState	15 MB / 300 MB / 5 %

5.3 Schemas

Nome	Descrição
UserSchema	Schema relacionado a um grupo de tabelas de dados relativos aos utilizadores e funcionalidades dos mesmos.
ProductsSchema	Schema relacionado a um grupo de tabelas de dados relativos aos produtos e stocks.
SalesSchema	Schema relacionado a um grupo de tabelas de dados relativos às vendas e funcionalidades das mesmas.
RegionsSchema	Schema relacionado a um grupo de tabelas de dados relativos a regiões e territórios.

6. Verificação da migração de dados

6.1. Consultas sobre a base de dados original

6.1.1. Nº de Customers

```
SELECT count([CUSTOMER KEY]) as [Number Of Customers On Old WWI]
FROM [WWI_DS].[dbo].[Customer]
```

	Number Of Customers On Old WWI
1	402

6.1.2. Nº de Customers por categoria

```
SELECT [Category] , count([CUSTOMER KEY]) as [Number Of Customers On Old WWI]
FROM [WWI_DS].[dbo].[Customer]
GROUP BY [Category]
```

	Category	Number Of Customers On Old WWI
1	24H Shop	73
2	Gas Station Shop	54
3	Gift Shop	62
4	Kiosk	62
5	Novelty Shop	151

6.1.3. Total de Vendas por Employee

```
SELECT Emp.[Employee], count(sa.[Salesperson Key]) FROM [WWI_DS].[dbo].[Sale] as sa
join [WWI_DS].[dbo].[Employee] as Emp on Emp.[Employee Key] = sa.[Salesperson Key]
group by Emp.[Employee]
```

	Employee	(No column name)
1	Amy Trefl	22444
2	Anthony Grosse	22521
3	Archer Lamble	23331
4	Hudson Hollinworth	22902
5	Hudson Onslow	22681
6	Jack Potter	22784
7	Kayla Woodcock	23079
8	Lily Code	22642

6.1.4. Total monetário de vendas por "Stock Item"

```
SELECT si.[Stock Item], sum(sale.[Quantity] * sale.[Unit Price]) as [MonetaryValue]
FROM [WWI_DS].[dbo].[Sale] as sale
JOIN [WWI_DS].[dbo].[Stock Item] as si on si.[Stock Item Key] = sale.[Stock Item Key]
group by si.[Stock Item]
```

	Stock Item	MonetaryValue
1	Furry gorilla with big eyes slippers (Black) XL	185024.00
2	Plush shark slippers (Gray) L	179424.00
3	Ogre battery-powered slippers (Green) L	189856.00
4	Red and white urgent heavy despatch tape 48mmx...	533032.80
5	Black and orange handle with care despatch tape 4...	565308.00
6	Clear packaging tape 48mmx100m	400750.00
7	DBA joke mug - mind if I join you? (White)	79768.00
8	USB food flash drive - donut	184859.20

6.1.5. Total monetário de vendas por ano por "Stock Item" ~

```
SELECT si.[Stock Item], YEAR(sale.[Delivery Date Key]) AS YearOfSale, sum(sale.[Quantity] *
sale.[Unit Price]) as [MonetaryValue]
FROM [WWI_DS].[dbo].[Sale] as sale
JOIN [WWI_DS].[dbo].[Stock Item] as si on si.[Stock Item Key] = sale.[Stock Item Key]
group by si.[Stock Item], YEAR(sale.[Delivery Date Key])
order by  YEAR(sale.[Delivery Date Key]) desc
```

	Stock Item	YearOfSale	MonetaryValue
1	Medium sized bubblewrap roll 20m	2016	150400.00
2	Furry animal socks (Pink) XL	2016	41640.00
3	Plush shark slippers (Gray) XL	2016	23424.00
4	Alien officer hoodie (Black) 4XL	2016	20825.00
5	Plush shark slippers (Gray) L	2016	20832.00
6	Clear packaging tape 48mmx75m	2016	67256.80
7	"The Gu" red shirt XML tag t-shir...	2016	145152.00
8	Developer joke mug - old C dev...	2016	8502.00

6.1.6. Total monetário de vendas por ano por "City"

```
SELECT c.[City], YEAR(sale.[Delivery Date Key]) AS YearOfSale, sum(sale.[Quantity] *
sale.[Unit Price]) as [MonetaryValue]
FROM [WWI_DS].[dbo].[Sale] as sale
JOIN [WWI_DS].[dbo].[City] as c on c.[City Key] = sale.[City Key]
group by c.[City], YEAR(sale.[Delivery Date Key])
order by  YEAR(sale.[Delivery Date Key]) desc
```

	City	YearOfSale	MonetaryValue
1	Twodot	2016	46996.10
2	Valdese	2016	18005.90
3	Cadogan	2016	36721.45
4	Raton	2016	30257.20
5	Bock	2016	8198.65
6	Airport ...	2016	38580.45
7	Guin	2016	35886.70
8	Newberg	2016	47676.80

6.2. Consultas sobre a nova base de dados

6.2.1. Nº de Customers

```
SELECT count(idCustomer) as [Number Of Customers On New WWI]
FROM [WWI].[UserSchema].[Customer]
```

	Number Of Customers On New WWI
1	402

6.2.2. Nº de Customers por categoria

```
SELECT ca.name , count(c.idCustomer) as NumberOfCustomersOnNewWWI
FROM [WWI].[UserSchema].[Customer] as c
JOIN [WWI].[ProductsSchema].[Category] as ca on ca.idCategory = c.categoryId
group by ca.name
```

	name	NumberOfCustomersOnNewWWI
1	24H Shop	73
2	Gas Station Shop	54
3	Gift Shop	62
4	Kiosk	62
5	Novelty Shop	151

6.2.3. Total de Vendas por Employee

```
SELECT Person.name , count(Sale.employeeId) as  [Total of Sales]
FROM [WWI].[SalesSchema].[Sale] as Sale
join [WWI].[UserSchema].[Employee] as Employee on Employee.idEmployee = Sale.employeeId
join [WWI].[UserSchema].[Person] as Person on Person.idPerson = Employee.personId
group by Person.name
```

	name	Total of Sales
1	Amy Trefl	22444
2	Anthony Grosse	22521
3	Archer Lamble	23331
4	Hudson Hollinworth	22902
5	Hudson Onslow	22681
6	Isabella Rupp	1
7	Jack Potter	22784
8	Kayla Woodcock	23079

6.2.4. Total monetário de vendas por "Stock Item"

```
SELECT p.description , sum(sp.quantity * p.unitPrice) as [MonetaryValue]
FROM [WWI].[SalesSchema].[SaleProduct] as sp
JOIN [WWI].[ProductsSchema].[Product] as p on p.idProduct = sp.productId
group by p.description

-- OU --

SELECT p.description , sum(totalExcludingTax) as [MonetaryValue]
FROM [WWI].[SalesSchema].[SaleProduct] as sp
JOIN [WWI].[ProductsSchema].[Product] as p on p.idProduct = sp.productId
group by p.description] as p on p.idProduct = sp.productId
```

	description	MonetaryValue
1	"The Gu" red shirt XML tag t-shirt (Black) 3XL	1246536,00
2	"The Gu" red shirt XML tag t-shirt (Black) 3XS	1213704,00
3	"The Gu" red shirt XML tag t-shirt (Black) 4XL	418392,00
4	"The Gu" red shirt XML tag t-shirt (Black) 5XL	1184328,00
5	"The Gu" red shirt XML tag t-shirt (Black) 6XL	1261440,00
6	"The Gu" red shirt XML tag t-shirt (Black) 7XL	1307448,00
7	"The Gu" red shirt XML tag t-shirt (Black) L	1195776,00
8	"The Gu" red shirt XML tag t-shirt (Black) M	1245456,00

6.2.5. Total monetário de vendas por ano por "Stock Item"

```
SELECT p.description , YEAR(s.deliveryDate) as YearOfSale, sum(sp.quantity * p.unitPrice) as
[MonetaryValue]
FROM [WWI].[SalesSchema].[SaleProduct] as sp
JOIN [WWI].[ProductsSchema].[Product] as p on p.idProduct = sp.productId
JOIN [WWI].[SalesSchema].[Sale] as s on s.idSale = sp.saleId
group by p.description, YEAR(s.deliveryDate)
order by YEAR(s.deliveryDate) desc

-- OU –

SELECT p.description , YEAR(s.deliveryDate) as YearOfSale, sum(totalExcludingTax) as
[MonetaryValue]
FROM [WWI].[SalesSchema].[SaleProduct] as sp
JOIN [WWI].[ProductsSchema].[Product] as p on p.idProduct = sp.productId
JOIN [WWI].[SalesSchema].[Sale] as s on s.idSale = sp.saleId
group by p.description, YEAR(s.deliveryDate)
order by YEAR(s.deliveryDate) desc
```

	description	YearOfSale	MonetaryValue
1	Packing knife with metal insert blade (Yellow) 18mm	2022	200,00
2	RC big wheel monster truck with remote control (B...	2016	31905,00
3	Plush shark slippers (Gray) M	2016	17856,00
4	Developer joke mug - there are 10 types of people...	2016	9373,00
5	Red and white urgent despatch tape 48mmx75m	2016	68109,60
6	RC toy sedan car with remote control (Red) 1/50 ...	2016	16950,00
7	"The Gu" red shirt XML tag t-shirt (White) 6XL	2016	167400,00
8	Developer joke mug - (hip, hip, array) (Black)	2016	9776,00

6.2.6. Total monetário de vendas por ano por "City"

```
SELECT c.name , YEAR(s.deliveryDate) as YearOfSale, sum(sp.quantity * p.unitPrice) as
[MonetaryValue]
FROM [WWI].[SalesSchema].[SaleProduct] as sp
JOIN [WWI].[ProductsSchema].[Product] as p on p.idProduct = sp.productId
JOIN [WWI].[SalesSchema].[Sale] as s on s.idSale = sp.saleId
JOIN [WWI].[RegionsSchema].[City] as c on c.idCity = s.cityId
group by c.name, YEAR(s.deliveryDate)
order by YEAR(s.deliveryDate) desc

-- OU --

SELECT c.name , YEAR(s.deliveryDate) as YearOfSale, sum(totalExcludingTax) as [MonetaryValue]
FROM [WWI].[SalesSchema].[SaleProduct] as sp
JOIN [WWI].[ProductsSchema].[Product] as p on p.idProduct = sp.productId
JOIN [WWI].[SalesSchema].[Sale] as s on s.idSale = sp.saleId
JOIN [WWI].[RegionsSchema].[City] as c on c.idCity = s.cityId
group by c.name, YEAR(s.deliveryDate)
order by YEAR(s.deliveryDate) desc
```

	name	YearOfSale	MonetaryValue
1	North Eastham	2016	27551,80
2	Bratt	2016	40995,75
3	Bokeelia	2016	23158,20
4	Bakers Mill	2016	49482,00
5	Drakesboro	2016	23206,60
6	Shell	2016	16258,60
7	Schoharie	2016	22058,65
8	Willow Valley	2016	13047,00

7. Programação

7.1 Views

Nome	Descrição
dbo.ViewRecordOfWWI	Visualizar a inserção mais recente de dados na tabela de Records, ou seja, Select na tabela RecordOfWWI
dbo.ViewNumberOfRecords	Visualizar a inserção mais recente de dados na tabela de Records, ou seja, Select na tabela NumberOfRecords
dbo.view_SalesTerritory	Visualizar a a informação toda sobre a tabela RegionsSchema.City, onde o salesTerritory corresponde a 'Rocky Mountain'. Nota: esta view foi criada como auxilia à visualização do user SalesTerritory, da parte dos Níveis de acesso.

7.2 Functions

Nome	Atributos	Requisito	Descrição
dbo.returnIfSaleEmpty	@idSale int	R12	Função que retorna um valor verdadeiro ou falso se a venda conter produtos ou não
dbo.totalSalePriceFunction	@saleId int	R13	Função que retorna o valor total de uma venda (sum de todos os totais por produto na venda)
dbo.verifyLeadTimeDaysFunction	@idSale int	R14	Função que retorna uma mensagem caso a diferença entre data de entrega e a data corrente de uma venda não esteja de acordo com o leadTimeDays de um produto
dbo.verifySaleContainsChillerStockFunction	@idSale int	R15	Função que verifica se uma venda contém produtos com chiller stock e sem chiller stock ao mesmo tempo
dbo.encryptPassword	@password nvarchar(50)	RM02	Função que retorna um hash do campo que tem como parametro de entrada, neste caso @password.

7.3 Stored procedures

Nome	Atributos	Requisito	Descrição
dbo.Person_insertInto	@name nvarchar(100) @preferredName nvarchar(50) @email nvarchar(50) @password nvarchar(50) @photo varbinary(max) @personTypeId int	R02	Stored Procedure que cria um utilizador (Person)
dbo.Person_update	@idPerson int @name nvarchar(100) @preferredName nvarchar(50) @isSalesPerson bit @email nvarchar(50) @password nvarchar(50) @photo varbinary(max) @isActive bit @personTypeId int	R03	Stored Procedure que atualiza um utilizador (Person)
dbo.Person_removeFrom	@idPerson int	R04	Stored Procedure que remove um utilizador (Person)
dbo.newDiscount	@productId int @discountPercentage int @beginDateGDH datetime @endDateGDH datetime	R06	Stored Procedure que cria uma promoção (Discount)
dbo.updateDiscountDates	@idDiscount int, @beginDateGDH datetime, @endDateGDH datetim	R07	Stored Procedure que atualiza as datas de início e fim de uma promoção (Discount)
dbo.createSale	@idSale int, @deliveryDate datetime @employeeId int @customerId @invoiceId @cityId @notes nvarchar(500)	R08	Stored Procedure que cria uma venda (Sale)
dbo.addProductToSale	@productId int @saleId int @quantity int @description nvarchar(250) @totalExcludingTax money @taxAmout money @profit money @totalIncludingTax money @totalDryItens int @totalChillerItens int	R09	Stored Procedure que cria uma venda relacionada a um produto (SaleProduct)
dbo.updateProductQuantityOnSale	@idSaleProduct int @quantity int @totalExcludingTax money @taxAmout money @profit money	R10	Stored Procedure que atualiza a quantidade de um produto da venda (SaleProduct)

	@totalIncludingTax money @totalDryItens int @totalChillerItens int		
dbo.deleteFromSaleProductTable	@idSaleProduct int @idSale int	R11	Stored Procedure que remove uma relacionada a um produto (SaleProduct)
dbo.totalSalePriceProcedure	@saleId int	R13	StoredProcedure que retorna o valor total de uma venda (sum de todos os totais por produto na venda)
dbo.authenticateUser	@email nvarchar(50) @password nvarchar(50)	R01	Stored Procedure para verificar a existência de um utilizador com através de email e password
dbo.tokenGeneration	@email nvarchar(50)	R05	Stored Procedure que gera um token na tabela TokenPassword para recuperar a password agregada a um email da conta de utilizador

7.4 Triggers

Nome	Tipo	Tabela	Requisito	Descrição
dbo.dbo_Insert_User_Email_TriggerDuplicate	AFTER INSERT	[UserSchema].[Person]	RM01	Trigger para verificar a existência de emails duplicados em utilizadores (Person).

8. Catálogo/Metadados

8.1 Geradores

Nome	Atributos	Descrição
dbo.sp_insertProcedure	@table_name NVARCHAR(255)	Gera um sp de Insert Into @table_name, e obtém e armazena informação em variáveis da tabela escolhida: parâmetros, colunas e tipos de dados associados
dbo.sp_updateProcedure	@table_name NVARCHAR(255)	Gera um sp de Update @table_name, e obtém e armazena informação em variáveis da tabela escolhida: parâmetros, coluna e tipo de dados no ID, colunas e tipos de dados associados
dbo.sp_deleteProcedure	@table_name NVARCHAR(255)	Gera um sp de Delete From @table_name, e obtém e armazena informação em variáveis da tabela escolhida: parâmetro ID, coluna do ID e tipo de dado no ID

8.2 Monitorização

Nome	Atributos	Descrição
dbo.sp_RecordOfDatabase	Nenhum	Cria uma tabelas dedicadas (se não existir) que guarda os registos e os registos anteriores de cada tabela da base de dados (nome da tabela, colunas, etc). Se já existirem, atualiza os dados das tabelas criadas.
dbo. sp_NumberOfRecords	Nenhum	Cria uma tabela (se não existir) que guarda o número de registos de cada tabela da base de dados. Se existir, guarda os dados anteriores na coluna RowCount_Previous.

9. Índices

9.1 Views

Nome	Descrição
dbo.view_SalesTotalPerCity	Visualizar o número total vendas por cidade. Deve ser retornado o nome da cidade, o nome do vendedor,o total de vendas.
dbo.view_CategoryGrowthPerYear	Visualizar as vendas e o cálcula da taxa de crescimento de cada ano, face ao ano anterior, por categoria de cliente.
dbo.view_NOfProductsOnSalesPerColor	Visualizar o nº de produtos nas vendas por cor.

9.2 Índices

Designação	Tabela	Justificação/Consultas
_dta_index_City_6_997578592__K1_K2	[RegionsSchema].[City]	Consulta otimizada (exe: dbo.view_SalesTotalPerCity) , com indexação na coluna idCity, de chave primária, com a coluna name como agregada. É justificado a criação deste indice visto que é feito numa chave primária, tipo de colunas geralmente otimas para indexação.
_dta_index_Sale_6_1381579960__K4_2	[SalesSchema].[Sale]	Consulta otimizada (exe: dbo.view_CategoryGrowthPerYear), com indexação na coluna customerId, de chave estrangeira. É justificado a criação deste indice visto que é feito numa chave estrangeira de base chave primária, tipo de colunas geralmente ótimas para indexação.

_dta_index_Product_6_1349579846__K3	[ProductsSchema].[Product]	Consulta otimizada (exe: dbo.view_NOfProductsOnSalesPerColor), com indexação na coluna colorId, de chave estrangeira. É justificado a criação deste indice visto que é feito numa chave estrangeira de base chave primária, tipo de colunas geralmente ótimas para indexação.

9.3 Otimização e Execução de Consultas

Nesta parte são mostradas estatisticas que demonstram a execução das consultas, sobre:

A base de dados otimizada (normalizada) sem índices:

- 1ª View

```
SQL Server parse and compile time:
   CPU time = 16 ms, elapsed time = 31 ms.

(6524 rows affected)

 SQL Server Execution Times:
   CPU time = 78 ms,  elapsed time = 4587 ms.

Completion time: 2023-01-14T16:29:03.5847776+00:00
```

- 2ª View

```
SQL Server parse and compile time:
   CPU time = 15 ms, elapsed time = 42 ms.

(25 rows affected)

 SQL Server Execution Times:
   CPU time = 16 ms,  elapsed time = 1178 ms.

Completion time: 2023-01-14T16:24:13.0663872+00:00
```

- 3ª View

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 6 ms, elapsed time = 6 ms.

(8 rows affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 13 ms.

Completion time: 2023-01-14T16:50:48.5815923+00:00
```

A base de dados otimizada (normalizada) com índices:

- 1ª View

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 12 ms, elapsed time = 12 ms.

(6524 rows affected)

 SQL Server Execution Times:
   CPU time = 15 ms,  elapsed time = 95 ms.

Completion time: 2023-01-14T16:39:22.9326022+00:00
```

- 2ª View

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 7 ms, elapsed time = 7 ms.

(25 rows affected)

 SQL Server Execution Times:
   CPU time = 16 ms,  elapsed time = 22 ms.

Completion time: 2023-01-14T16:44:51.3316591+00:00
```

- 3ª View

```
 SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 0 ms.
 SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 5 ms.

 (8 rows affected)

  SQL Server Execution Times:
    CPU time = 16 ms,  elapsed time = 13 ms.
```

Com estes resultados podemos concluir que a utilização de indíces foi útil para aumentar a velocidade das consultas. Nas primeiras duas views nota-se uma diferença maior, o que é em si muito positivo, pois demostra que os indices foram bem usados e escolhidos. O terceiro porém não se nota grande diferença, uma vez que este é relativamente simples e rápido, e se calhar não se justifica a criação de um índice, porém foi criado pois foi pedido no enunciado.

10. Backup e Recuperação

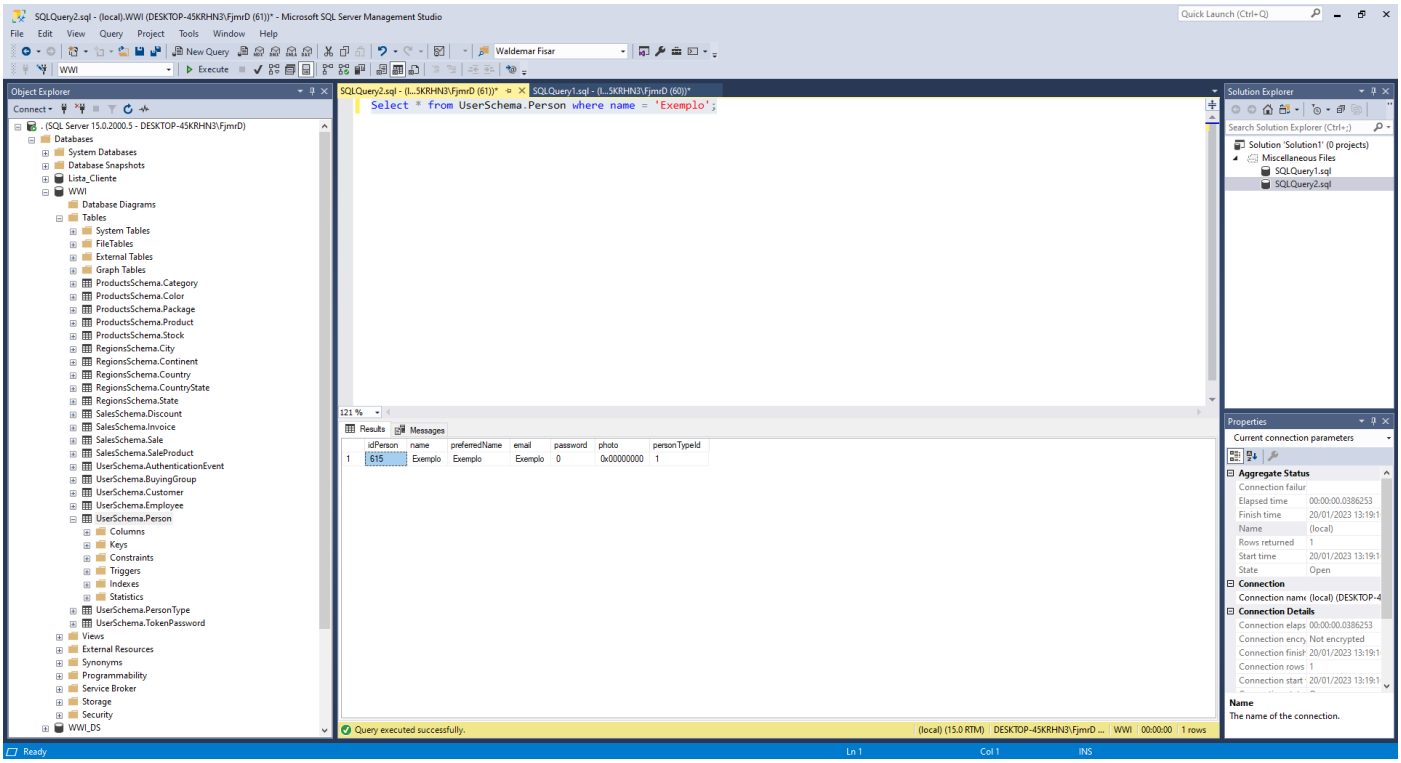
A politica de backups decidida, organiza se em base de Full backups semanais, seguidos por Diferencial Backups diários, e por ultimo Backup de Transaction Logs a cada hora.

O grupo decidiu que esta seria a estratégia mais eficiente para a motivação deste projeto, sendo este uma base de dados de vendas de produtos.

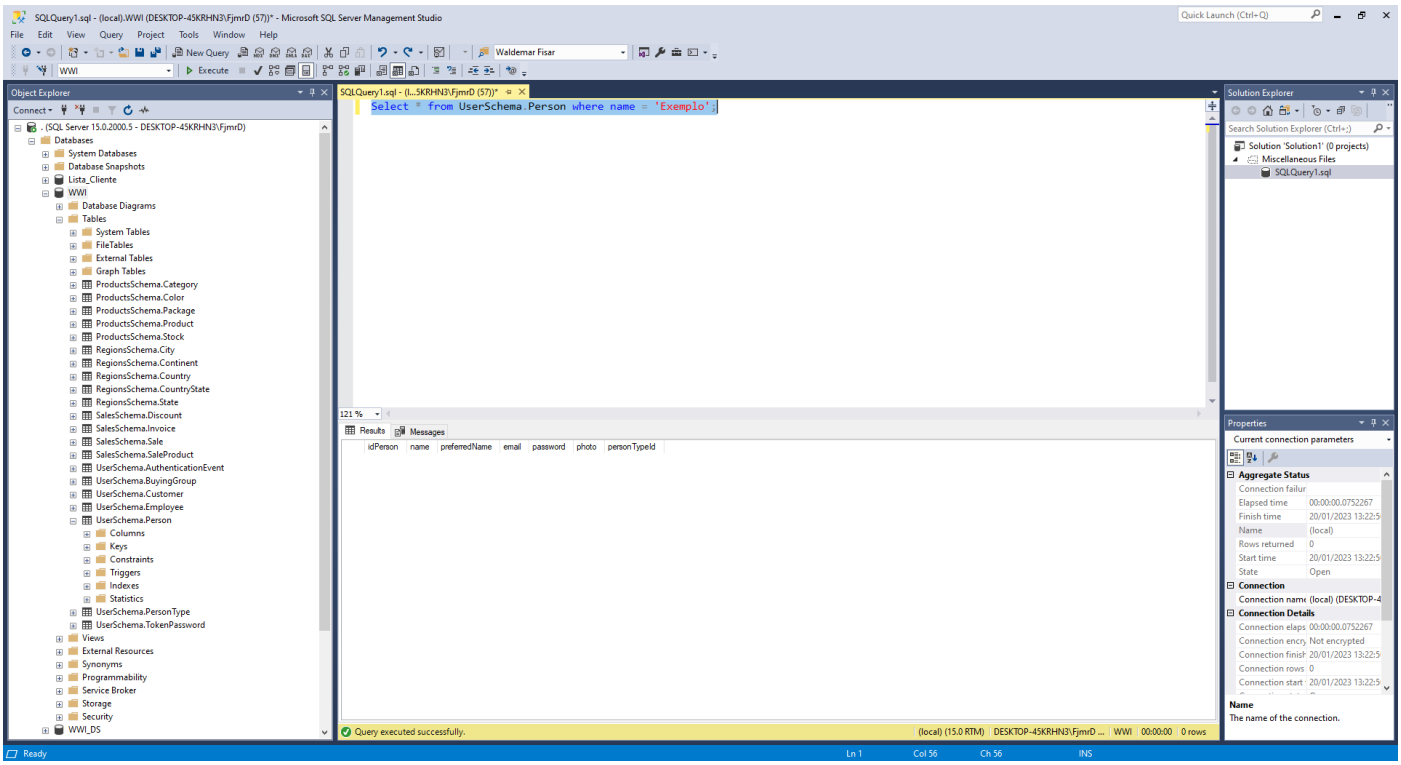
Os Full backups garantem que em caso de colapso total da base de dados, esta fica assegurada na sua totalidade. Os Diferencial Backups garantem que por exemplo, novos utilizadores registados, sejam recuperáveis.
Por último, para recuperar informação mais frequente, realiza-se backups de Transaction Logs a cada hora. Estes garantem informação recorrente, como por exemplo vendas realizadas.

O grupo traçou este plano de backups, com a logica previamente relatada. Um utilizador pode fazer X vendas, logo a quantidade de dados nas vendas poderá ser maior que a quantidade de dados nos utilizadores.

De seguida estão documentadas as simulações de crash nas tabelas Person (Diferential) e Sale (Transaction):

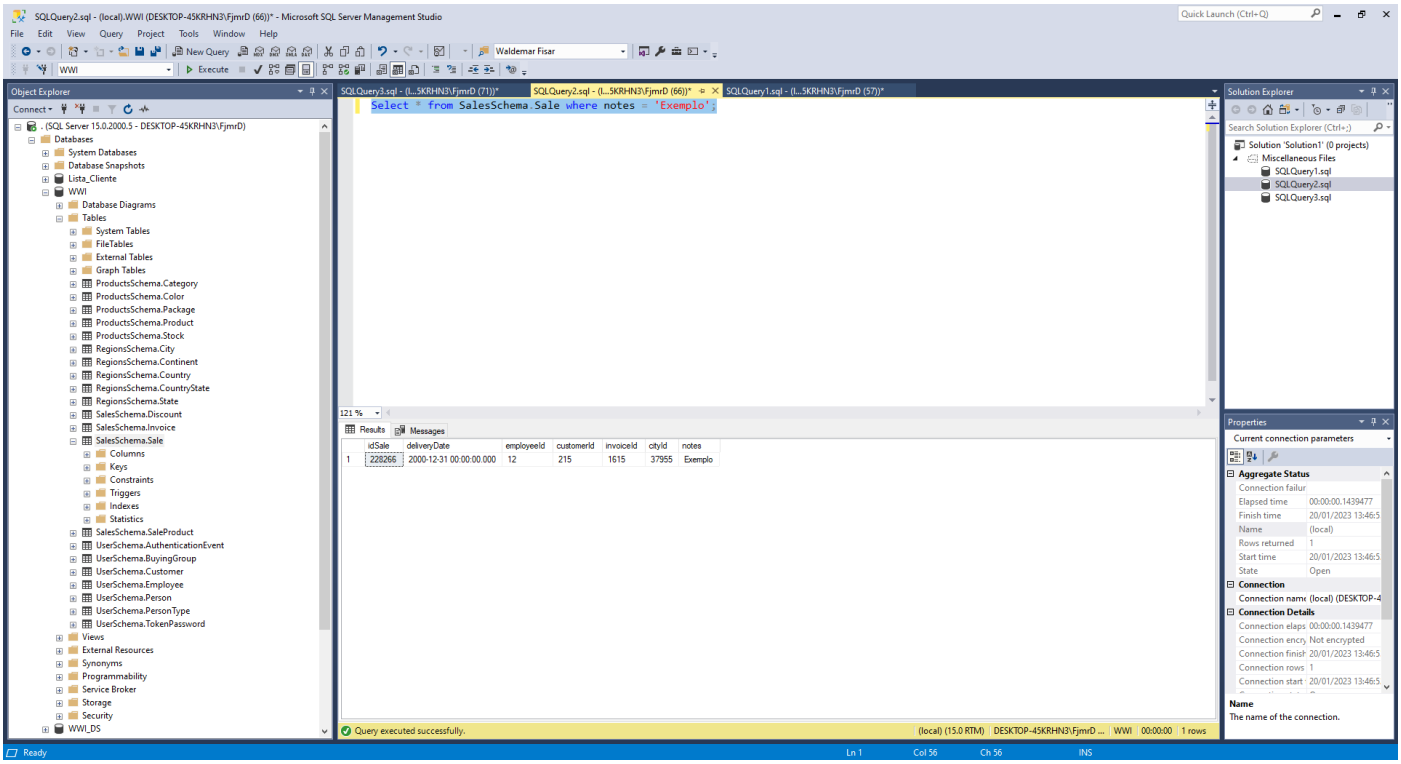


Para testar o backup Diferencial, foi feito um Full buckup previamente seguido por um backup Diferencial. Foi introduzido um dado exemplo na tabela Person, e realizou se a simulação do crash.

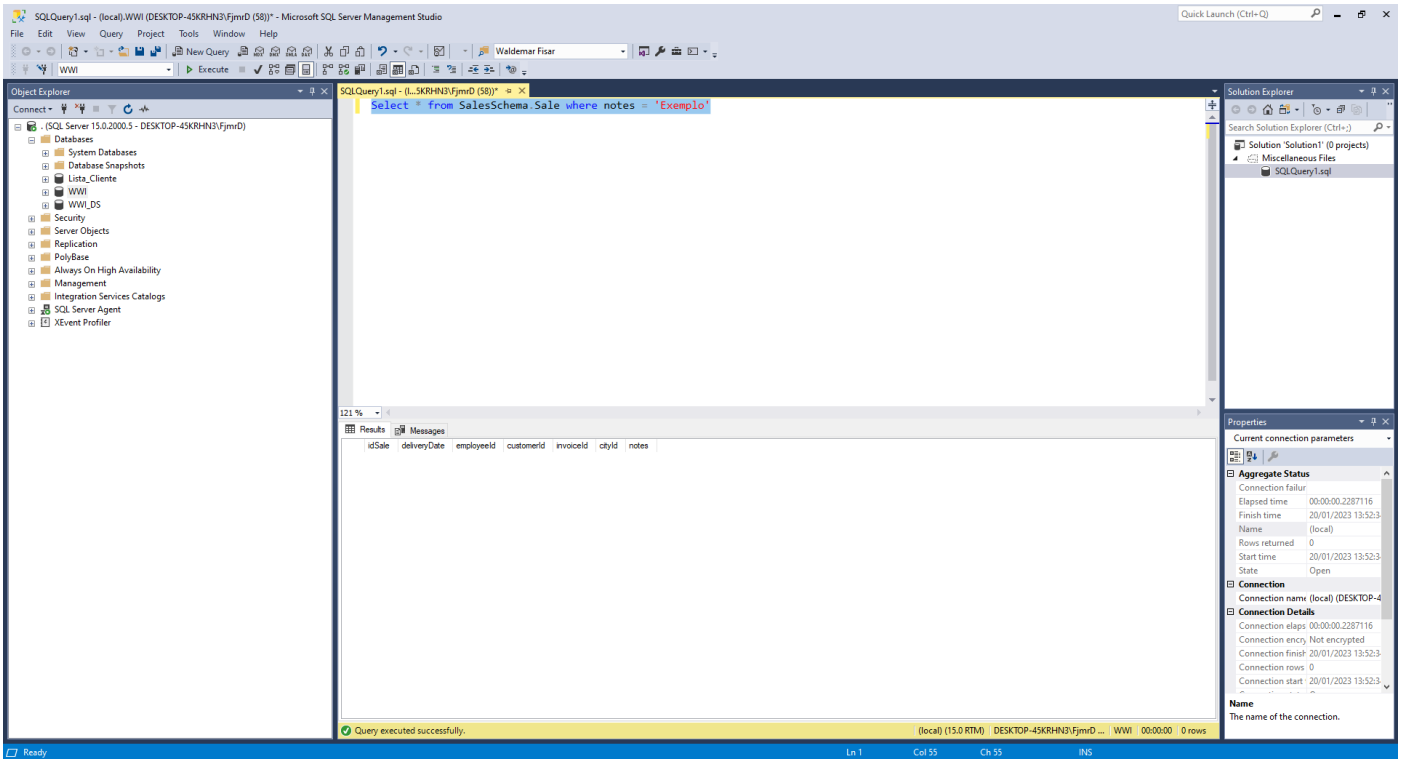


Ao realizar o restore, realizou-se uma query para procurar o dado exemplo, ao qual se verificou que ele não estava presente na base de dados, o que significa que o Backup Diferencial foi efetuado com sucesso.

O procedimento para o Backup Transaction foi semelhante:



Foi realizado um backup Transaction, e introduzido um dado exemplo na tabela Sale. Deu se a simulação do crash, ao qual a imagem seguinte revela o resultado:



Como esperado, ao procurarmos pelo dado Exemplo, ele não se encontra na base de dados, ao que podemos concluir que o backup Transaction também foi efetuado com sucesso.

11. Segurança e Controlo de Acessos

Para assegurar uma maior segurança na interação com a base da dados e os utilizadores e desenvolvedores, foram propostos criação de alguns níveis de acesso à informação, com a inclusão de logins, users, roles e views auxiliares. A parte da encriptação foi nos pedida para esconder informação relativa a passwords de utilizadores e preços de produtos.

11.1 Níveis de acesso à informação

Admin:

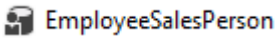
Admin

Descrição: Tem acesso a toda a informação. Não pode alterar a base de dados em si, apenas dados dentro das tabelas.

```
/*Admin User*/
/*Create Login and User*/
USE [master]
GO
CREATE LOGIN [Admin] WITH PASSWORD=N'admin123' MUST_CHANGE, DEFAULT_DATABASE=[WWI],
CHECK_EXPIRATION=ON, CHECK_POLICY=ON ← Criação do login do Admin
GO
```

```
use [master];
GO
USE [WWI]
GO
CREATE USER [Admin] FOR LOGIN [Admin] ← Criação do user Admin com o login Admin
GO
USE [WWI]
GO
ALTER ROLE [db_datareader] ADD MEMBER [Admin] ← Role para ler dados
GO
USE [WWI]
GO
ALTER ROLE [db_datawriter] ADD MEMBER [Admin] ← Role para escrever dados
GO
```

EmployeeSalesPerson:



Descrição: Tem acesso total às tabelas de suporte às vendas, e apenas acesso em modo de consulta às restantes tabelas. Criamos uma role que apenas tem permissão no SalesSchema. Por consequência, o user subscrito a este role apenas tem permissões para esse schema.

```
/*EmployeeSalesPerson*/
/*Role db_employee*/
USE [WWI]
GO
CREATE ROLE [db_employee] ← Criação da role
GO
USE [WWI]
GO
ALTER AUTHORIZATION ON SCHEMA::[SalesSchema]
TO [db_employee] ← Permitir à role manipular o schema SalesSchema
GO

/*Create Login and User*/
USE [master]
GO
CREATE LOGIN [EmployeeSalesPerson] WITH PASSWORD=N'employee123' MUST_CHANGE,
DEFAULT_DATABASE=[WWI], CHECK_EXPIRATION=ON, CHECK_POLICY=ON ← Criação do login
GO
use [master];
GO
USE [WWI]
GO
CREATE USER [EmployeeSalesPerson]
FOR LOGIN [EmployeeSalesPerson] ← Criação do user EmployeeSalesPerson
GO
USE [WWI]
GO
ALTER ROLE [db_employee] ADD MEMBER [EmployeeSalesPerson] ← Role criada anteriormente
GO
```

SalesTerritory:

SalesTerritory

Descrição: Apenas pode consultar a informação relativa ao seu território. Considere apenas o território "Rocky Mountain". Foi criado uma view auxiliar para demonstrar o nível de acesso.

```
/*SalesTerritory*/
/*Create Login and User*/
USE [master]
GO
CREATE LOGIN SalesTerritory WITH PASSWORD=N'guest' MUST_CHANGE, DEFAULT_DATABASE=[WWI],
CHECK_EXPIRATION=ON, CHECK_POLICY=ON ← Criação do login
GO
use [master];
GO
USE [WWI]
GO
CREATE USER SalesTerritory FOR LOGIN SalesTerritory ← Criação do user SalesTerritory
GO
GRANT SELECT ON dbo.view_SalesTerritory
TO SalesTerritory; ← Atribuir permissão de seleção da view ao SalesTerritory USER.
GO
ALTER USER SalesTerritory WITH DEFAULT_SCHEMA=[RegionsSchema]
GO
ALTER AUTHORIZATION ON SCHEMA::[RegionsSchema]
TO SalesTerritory ← Permitir ao user manipular o schema RegionsSchema
GO



/*View for SalesTerritory*/
USE [WWI]
GO
CREATE OR ALTER VIEW view_SalesTerritory AS
select * from RegionsSchema.City
WHERE salesTerritory = 'Rocky Mountain'

/*Tem permissões*/
EXECUTE AS USER = 'SalesTerritory' ← Executar como user SalesTerritory
GO
select * from view_SalesTerritory
REVERT;
```

idCity	name	stateId	salesTerritory	latestRecordedPopulation
28	Carter	WY	Rocky Mountain	10
29	Carter	MT	Rocky Mountain	58
48	El Jebel	CO	Rocky Mountain	3801
53	El Moro	CO	Rocky Mountain	221
67	El Rancho	CO	Rocky Mountain	0
92	Austin	MT	Rocky Mountain	0
132	Browning	MT	Rocky Mountain	1016
172	Crook	CO	Rocky Mountain	110
233	Roy	UT	Rocky Mountain	36884
258	Saco	MT	Rocky Mountain	197

```
/*Não tem permissões*/
EXECUTE AS USER = 'EmployeeSalesPerson' ← Executar como user EmployeeSalesPerson
GO
select * from view_SalesTerritory
REVERT;
```

```
Msg 229, Level 14, State 5, Line 84
The SELECT permission was denied on the object 'view_SalesTerritory', database 'WWI', schema 'dbo'.

Completion time: 2023-01-20T19:02:27.6563787+00:00
```

11.2 Encriptação

Password Encryption:

Descrição: Função que faz hashing da password quando é criado um utilizador. Returns: hashed password.

```
USE WWI
GO
CREATE or ALTER FUNCTION encryptPassword
(@password nvarchar(50))
RETURNS nvarchar
AS
BEGIN
	RETURN HashBytes('MD5', @password)
END
```

	idPerson	name	preferredName	email	password	photo	personTypeId
1	615	ADAD	ADSAD	SDADADADSDADADAD	wassup	NULL	1
2	616	ADsadAD	ADasdSAD	SDADAsad	睹	NULL	1

1. Não Encriptada 2.Encriptada (Hashing)

Price Encryption:

Descrição: Cria-se uma master key, cria-se um certificado, cria-se uma chave de encriptação com o algoritmo AES_256.
Altera-se depois a tabela produtos para adicionar uma coluna com as chaves (unit price como coluna original).

```
USE WWI
GO
CREATE MASTER KEY ENCRYPTION
BY PASSWORD = 'SADA'
GO
CREATE CERTIFICATE WWI_CERT
WITH SUBJECT = 'Protect Data'
GO
CREATE SYMMETRIC KEY EncryptKey
WITH ALGORITHM = AES_256 ENCRYPTION
BY CERTIFICATE WWI_CERT
```

```
GO
ALTER TABLE ProductsSchema.Product
ADD encryptedUnitPrice VARBINARY(256)
GO
OPEN SYMMETRIC KEY EncryptKey DECRYPTION
BY CERTIFICATE WWI_CERT
UPDATE ProductsSchema.Product
SET encryptedUnitPrice = ENCRYPTBYKEY(KEY_GUID('EncryptKey'), CAST(unitPrice AS VARCHAR(20)))
GO
```

	idProduct	description	colorId	d...	unitPrice	taxRate	brand	size	barCode	recomm...	typi...	lea...	qua...	encryptedUnitPrice
1	45	Bubblewrap dispenser (Blue) 1.5m	2	1	240,00	14	N/A	1.5m	N/A	358,80	10	14	1	0x00A28161E040E846A82C642B3012004002000000DF1007B...
2	47	32 mm Anti static bubble wrap (Blue) 50m	2	1	105,00	14	N/A	50m	N/A	156,98	10	14	10	0x00A28161E040E846A82C642B3012004002000000096327FE...
3	48	32 mm Anti static bubble wrap (Blue) 20m	2	1	48,00	14	N/A	20m	N/A	71,76	6	14	10	0x00A28161E040E846A82C642B3012004002000000CA2C95...
4	49	32 mm Anti static bubble wrap (Blue) 10m	2	1	32,00	14	N/A	10m	N/A	47,84	5	14	10	0x00A28161E040E846A82C642B3012004002000000B479666...
5	50	20 mm Anti static bubble wrap (Blue) 50m	2	1	102,00	14	N/A	50m	N/A	152,49	10	14	10	0x00A28161E040E846A82C642B3012004002000000944215C...
6	51	20 mm Anti static bubble wrap (Blue) 20m	2	1	45,00	14	N/A	20m	N/A	67,28	6	14	10	0x00A28161E040E846A82C642B3012004002000000784167B...
7	52	20 mm Anti static bubble wrap (Blue) 10m	2	1	29,00	14	N/A	10m	N/A	43,36	5	14	10	0x00A28161E040E846A82C642B3012004002000000039A42E5...
8	53	10 mm Anti static bubble wrap (Blue) 50m	2	1	99,00	14	N/A	50m	N/A	148,01	10	14	10	0x00A28161E040E846A82C642B3012004002000000DD1197A...
9	54	10 mm Anti static bubble wrap (Blue) 20m	2	1	42,00	14	N/A	20m	N/A	62,79	6	14	10	0x00A28161E040E846A82C642B3012004002000000F82D1D...
10	55	10 mm Anti static bubble wrap (Blue) 10m	2	1	26,00	14	N/A	10m	N/A	38,87	5	14	10	0x00A28161E040E846A82C642B3012004002000000049DD7C8...
11	103	Superhero action jacket (Blue) 5XL	2	1	34,00	12	N/A	5XL	N/A	50,83	0,4	12	1	0x00A28161E040E846A82C642B3012004002000000396983C...
12	104	Superhero action jacket (Blue) 4XL	2	1	34,00	12	N/A	4XL	N/A	50,83	0,4	12	1	0x00A28161E040E846A82C642B3012004002000000F069669...
13	105	Superhero action jacket (Blue) 3XL	2	1	34,00	12	N/A	3XL	N/A	50,83	0,4	12	1	0x00A28161E040E846A82C642B3012004002000000BEACFE...
14	106	Superhero action jacket (Blue) XXL	2	1	30,00	12	N/A	XXL	N/A	44,85	0,35	12	1	0x00A28161E040E846A82C642B3012004002000000BC8082C...
15	107	Superhero action jacket (Blue) XL	2	1	30,00	12	N/A	XL	N/A	44,85	0,35	12	1	0x00A28161E040E846A82C642B3012004002000000001642DA...

12. Controlo de Concorrência

Para esta parte, foi nos pedido definir níveis de isolamento no controlo transacional para várias funcionalidades, estas mesmas relacionadas com a interação de dados nas tabelas de vendas e produtos, algo importante devido à elevada taxa de transação de dados nestes tipos de tabelas.

- Adicionar um produto a uma venda:

Este processo deve usar o nível de isolamento READ COMMITTED. Evita dirty reads, mas permitirá que outras sessões façam alterações nos dados enquanto o processo estiver em execução.

Código:

```
BEGIN TRANSACTION
    SET TRANSACTION ISOLATION LEVEL READ COMMITTED

       EXEC addProductToSale 1, 2, 2, 'adasd', 20, 2, 3, 22, 2, 0

    -- Check if the product was added successfully
    IF @@ROWCOUNT = 0
        ROLLBACK TRANSACTION
    ELSE
        COMMIT TRANSACTION
```

- Atualizar o preço de um produto que não esteja em nenhuma venda ativa:

Este processo deve usar o nível de isolamento SERIALIZABLE. Impede que outras sessões façam alterações nos dados até que o processo seja concluído. Impede também que qualquer nível de isolamento de leitura confirmada e repetível leia os dados que estão a ser atualizados.

Código:

```
BEGIN TRANSACTION
    SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

    DECLARE @productId INT,
       @unitPrice DECIMAL(18, 2)
    SET @productId = 1
    SET @unitPrice = 5.5

    -- Update the price of the product
    UPDATE [ProductsSchema].[Product]
    SET unitPrice = @unitPrice
    WHERE idProduct = @productId
    AND idProduct
    NOT IN (SELECT productId FROM [SalesSchema].[Sale] as s
                                JOIN [SalesSchema].[SaleProduct] as sp ON sp.saleId = s.idSale
                                WHERE deliveryDate > GETDATE())

    -- Check if the product was updated successfully
    IF @@ROWCOUNT = 0
        ROLLBACK TRANSACTION
    ELSE
        COMMIT TRANSACTION
```

- Calcular o total de vendas e a quantidade de produtos:

Este processo deve usar o nível de isolamento REPEATABLE READ. Impede que outras sessões modifiquem os dados enquanto o processo estiver em execução. Garante que os dados lidos pelo processo são consistentes durante a transação.

Código:

```
BEGIN TRANSACTION
    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

    DECLARE @saleId INT,
       @totalSale DECIMAL(18, 2),
       @totalQuantity INT
    SET @saleId = 2

    -- Calculate the total sale and quantity of products in the sale
    SELECT @totalSale = SUM(totalIncludingTax), @totalQuantity = SUM(quantity)
    FROM [SalesSchema].[SaleProduct]
    WHERE saleId = @saleId

    PRINT 'Total Sale: ' + CAST(@totalSale AS VARCHAR(20))
    PRINT 'Total Quantity: ' + CAST(@totalQuantity AS VARCHAR(20))

COMMIT TRANSACTION
```

13. Descrição da Demonstração

13.1 Script de demonstração sobre a base de dados relacional

-- Procedures
-- Person_insertInto

```
EXEC [WWI].[dbo].Person_insertInto 'Luís', 'Luís', 'lui.luis@hotmail.com', '123', null, 1

SELECT * FROM [WWI].[UserSchema].[Person]
WHERE name = 'Luís'
```

	idPerson	name	preferredName	email	password	photo	personTypeId
1	1615	Luís	Luís	lui.luis@hotmail.com	✞	NULL	1

-- Person_update

```
EXEC [WWI].[dbo].Person_update 1615, 'Luís', 'Lui', 'lui.luis@hotmail.com', '123', null, 2

SELECT * FROM [WWI].[UserSchema].[Person]
WHERE name = 'Luís'
```

	idPerson	name	preferredName	email	password	photo	personTypeId
1	1615	Luís	Lui	lui.luis@hotmail.com	123	NULL	2

-- Person_removeFrom

```
EXEC [WWI].[dbo].Person_removeFrom 1615

SELECT * FROM [WWI].[UserSchema].[Person]
WHERE name = 'Luís'
```

idPerson	name	preferredName	email	password	photo	personTypeId

-- newDiscount

```
EXEC [WWI].[dbo].newDiscount 10,'2022-11-20 00:00:00','2022-11-30 00:00:00'

SELECT * FROM [WWI].[SalesSchema].[Discount]
```

	idDiscount	discountPercentage	beginDateGDH	endDateGDH
1	1	0	1753-01-01 00:00:00.000	9999-12-31 23:59:59.000
2	2	25	2022-04-23 06:00:00.000	2022-04-30 20:00:00.000
3	3	50	2022-02-11 06:00:00.000	2022-02-25 20:00:00.000
4	4	10	2022-11-20 00:00:00.000	2022-11-30 00:00:00.000

-- updateDiscountDates

```
EXEC [WWI].[dbo].updateDiscountDates 1 ,'2022-11-25 00:00:00','2022-12-05 00:00:00'

SELECT * FROM [WWI].[SalesSchema].[Discount]
```

	idDiscount	discountPercentage	beginDateGDH	endDateGDH
1	1	0	2022-11-25 00:00:00.000	2022-12-05 00:00:00.000
2	2	25	2022-04-23 06:00:00.000	2022-04-30 20:00:00.000
3	3	50	2022-02-11 06:00:00.000	2022-02-25 20:00:00.000
4	4	10	2022-11-20 00:00:00.000	2022-11-30 00:00:00.000

-- createSale

```
EXEC [WWI].[dbo].createSale 600000, '2022-11-25 00:00:00',2,2,2, 2, 'Notas'

SELECT * FROM [WWI].[SalesSchema].[Sale]
where idSale = 600000
```

	idSale	deliveryDate	employeeId	customerId	invoiceId	cityId	notes
1	600000	2022-11-25 00:00:00.000	2	2	2	2	Notas

-- addProductToSale

```
EXEC [WWI].[dbo].addProductToSale 10, 600000, 5, 5 ,200, 10, 0, 220, 0, 0

SELECT * FROM [WWI].[SalesSchema].[SaleProduct]
WHERE saleId = 600000
```

	idSaleProduct	productId	saleId	quantity	description	totalExcludingTax	taxAmout	profit	totalIncludingTax	totalDryItens	totalChillerItens
1	228268	10	600000	5	5	200,00	10,00	0,00	220,00	0	0

-- updateProductQuantityOnSale

```
EXEC [WWI].[dbo].updateProductQuantityOnSale 228270, 12, 0,0, 0, 0, 0, 0

SELECT * FROM [WWI].[SalesSchema].[SaleProduct]
WHERE saleId = 600000
```

	idSaleProduct	productId	saleId	quantity	description	totalExcludingTax	taxAmout	profit	totalIncludingTax	totalDryItens	totalChillerItens
1	228270	10	600000	12	5	0,00	0,00	0,00	0,00	0	0

-- deleteFromSaleProductTable

```
EXEC [WWI].[dbo].deleteFromSaleProductTable 228268, 600000

SELECT * FROM [WWI].[SalesSchema].[SaleProduct]
WHERE saleId = 600000
```

idSaleProduct	productId	saleId	quantity	description	totalExcludingTax	taxAmout	profit	totalIncludingTax	totalDryItens	totalChillerItens

-- totalSalePriceProcedure

```
EXEC [WWI].[dbo].totalSalePriceProcedure 600000

SELECT * FROM [WWI].[SalesSchema].[SaleProduct]
WHERE saleId = 600000
```

	Sale ID	Total Sale Price Taxed	Total Price NotTax
1	600000	220,00	200,00

-- authenticateUser

```
EXEC [WWI].[dbo].authenticateUser 'lui.luis@hotmail.com', '123'
```

```
Aunthentication valid!

(1 row affected)

Completion time: 2022-11-21T19:31:42.0180708+00:00
```

```
SELECT * FROM [WWI].[UserSchema].Person
WHERE email = 'lui.luis@hotmail.com'

SELECT * FROM [WWI].[UserSchema].AuthenticationEvent
```

	idPerson	name	preferredName	isSalesPerson	email	password	photo	isActive	personTypeId
1	1233	Luís	Luís	0	lui.luis@hotmail.com	123	NULL	0	2

	idAuthenticationEvent	personId	authenticationGDH	ipAddress	actionResult
1	1	1233	2022-11-21 19:30:05.187	0.0.0.0	1
2	2	1233	2022-11-21 19:31:41.147	0.0.0.0	1

-- tokenGeneration

```
EXEC [WWI].[dbo].tokenGeneration 'lui.luis@hotmail.com'

SELECT * FROM [WWI].[UserSchema].TokenPassword
```

	idTokenPassword	personId	token	generationGDH
1	1	1616	7BE33937-243A-4460-A7A1-2D0BC1DD76FE	2023-01-20 20:31:49.763

-- Functions
-- returnIfSaleEmpty

```
select [WWI].[dbo].returnIfSaleEmpty (600000)
```

	(No column name)
1	0

----> not empty

-- totalSalePriceFunction

```
select [WWI].[dbo].totalSalePriceFunction (600000) [Total Price Including Tax]
```

	Total Price Including Tax
1	220

-- verifyLeadTimeDaysFunction

```
select [WWI].[dbo].verifySaleContainsChillerStockFunction (600000)
```

	(No column name)
1	Lead Time Days doesnt correspond with delivery date!

-- verifySaleContainsChillerStockFunction

```
select [WWI].[dbo].verifySaleContainsChillerStockFunction (600000)
```

	(No column name)
1	0

```
SELECT * FROM [WWI].[SaleSchema].[SaleProduct]
join [WWI].[SaleSchema].[Sale] as sp on sp.idSale = saleId
WHERE saleId = 600000
```

	idSaleProduct	productId	saleId	quantity	description	totalExcludingTax	taxAmout	profit	totalIncludingTax	totalDryItens	totalChillerItens	idSale	deliveryDate	employeeId	notes
1	228267	10	600000	5	5	200,00	10,00	0,00	220,00	0	0	600000	2022-11-25 00:00:00.000	2	Notas

--Triggers
-- dbo_Insert_User_Email_TriggerDuplicate

```
INSERT INTO [WWI].[UserSchema].[Person] VALUES ('Guilherme', 'Gui', 0, 'guib@hotmail.com', '123',
null, 0, 1)
SELECT * FROM [WWI].[UserSchema].[Person]
WHERE name ='Guilherme'
```

	idPerson	name	preferredName	isSalesPerson	email	password	photo	isActive	personTypeId
1	616	Guilherme	Gui	0	guib@hotmail.com	123	NULL	0	1

--> 1º Insert

```
Msg 50000, Level 16, State 1, Procedure dbo_Insert_User_Email_TriggerDuplicate, Line 8 [Batch Start Line 81]
Email alreay exists!
Msg 3609, Level 16, State 1, Line 82
The transaction ended in the trigger. The batch has been aborted.
```

-->2º Insert

-- Metadata/Catalog Procedures
--Insert Generator

```
EXEC [WWI].[dbo].sp_insertProcedure 'Color'
```

```
CREATE PROCEDURE Color_Insert (@idColor int,@name nvarchar(100))
    AS BEGIN
        INSERT INTO Color (idColor,name)
        VALUES (@idColor,@name)
    END

Completion time: 2022-11-20T23:16:20.4664990+00:00
```

```
EXEC Color_Insert '10', 'NavyBlue'
SELECT * FROM [WWI].[ProductsSchema].[Color]
```

	idColor	name
1	0	N/A
2	1	Black
3	2	Blue
4	3	Gray
5	4	Light Brown
6	5	Red
7	6	Steel Gray
8	7	White
9	8	Yellow
10	10	NavyBlue

--Update Generator

```
EXEC [WWI].[dbo].sp_updateProcedure 'Color'
```

```
CREATE PROCEDURE Color_Update (@idColor int,@name nvarchar(100))
    AS BEGIN
        UPDATE Color
        SET idColor=@idColor,name=@name
        WHERE (idColor = @idColor)
    END

Completion time: 2022-11-20T23:17:34.2665737+00:00
```

```
EXEC Color_Update '10', 'LightBlue'
SELECT * FROM [WWI].[ProductsSchema].[Color]
```

	idColor	name
1	0	N/A
2	1	Black
3	2	Blue
4	3	Gray
5	4	Light Brown
6	5	Red
7	6	Steel Gray
8	7	White
9	8	Yellow
10	10	LightBlue

-- Remove Generator

```
EXEC [WWI].[dbo].sp_deleteProcedure 'Color'
```

```
CREATE PROCEDURE Color_Delete (@idColor int)
       AS BEGIN DELETE FROM Color
       WHERE (idColor = @idColor)
   END

Completion time: 2022-11-20T23:18:05.7229621+00:00
```

```
EXEC [WWI].[dbo].sp_deleteProcedure 'Color'
EXEC Color_Delete '10'
SELECT * FROM [WWI].[ProductsSchema].[Color]
```

	idColor	name
1	0	N/A
2	1	Black
3	2	Blue
4	3	Gray
5	4	Light Brown
6	5	Red
7	6	Steel Gray
8	7	White
9	8	Yellow

-- Record Database

```
EXEC [WWI].[dbo].sp_RecordOfDatabase
```

	TABLE_CATALOG	TABLE_SCHEMA	COLUMN_NAME	COLLATION_NAME	IS_NULLABLE	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME
1	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_AuthenticationEvent_Person
2	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_AuthenticationEvent_Person
3	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_AuthenticationEvent_Person
4	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_AuthenticationEvent_Person
5	WWI	dbo	stateId	Latin1_General_CI_AS	NO	varchar	2	WWI	dbo	FK_City_State
6	WWI	dbo	continentId	NULL	NO	int	NULL	WWI	dbo	FK_Country_Continent
7	WWI	dbo	buyingGroupId	NULL	NO	int	NULL	WWI	dbo	FK_Customer_BuyingGroup
8	WWI	dbo	categoryId	NULL	NO	int	NULL	WWI	dbo	FK_Customer_Category
9	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_Customer_Person

	TABLE_CATALOG	TABLE_SCHEMA	COLUMN_NAME	COLLATION_NAME	IS_NULLABLE	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME
1	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_AuthenticationEvent_Person
2	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_AuthenticationEvent_Person
3	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_AuthenticationEvent_Person
4	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_AuthenticationEvent_Person
5	WWI	dbo	stateId	Latin1_General_CI_AS	NO	varchar	2	WWI	dbo	FK_City_State
6	WWI	dbo	continentId	NULL	NO	int	NULL	WWI	dbo	FK_Country_Continent
7	WWI	dbo	buyingGroupId	NULL	NO	int	NULL	WWI	dbo	FK_Customer_BuyingGroup
8	WWI	dbo	categoryId	NULL	NO	int	NULL	WWI	dbo	FK_Customer_Category
9	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_Customer_Person


```
SELECT * FROM ViewRecordOfWWI
```

	TABLE_CATALOG	TABLE_SCHEMA	COLUMN_NAME	COLLATION_NAME	IS_NULLABLE	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME
1	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_AuthenticationEvent_Person
2	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_AuthenticationEvent_Person
3	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_AuthenticationEvent_Person
4	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_AuthenticationEvent_Person
5	WWI	dbo	stateId	Latin1_General_CI_AS	NO	varchar	2	WWI	dbo	FK_City_State
6	WWI	dbo	continentId	NULL	NO	int	NULL	WWI	dbo	FK_Country_Continent
7	WWI	dbo	buyingGroupId	NULL	NO	int	NULL	WWI	dbo	FK_Customer_BuyingGroup
8	WWI	dbo	categoryId	NULL	NO	int	NULL	WWI	dbo	FK_Customer_Category
9	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_Customer_Person
10	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_Customer_Person
11	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_Customer_Person
12	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_Customer_Person
13	WWI	dbo	productId	NULL	NO	int	NULL	WWI	dbo	FK_Discount_Product
14	WWI	dbo	productId	NULL	NO	int	NULL	WWI	dbo	FK_Discount_Product
15	WWI	dbo	productId	NULL	NO	int	NULL	WWI	dbo	FK_Discount_Product
16	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_Employee_Person
17	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_Employee_Person
18	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_Employee_Person
19	WWI	dbo	personId	NULL	NO	int	NULL	WWI	dbo	FK_Employee_Person

-- Record number of records per table

```
EXEC [WWI].[dbo].sp_NumberOfRecords
```

	TableName	Row_Count	previousRowCount
1	AuthenticationEvent	0	0
2	BuyingGroup	2	2
3	Category	5	5
4	City	116294	116294
5	Color	9	9
6	Continent	7	7
7	Country	249	249
8	Customer	402	402
9	Discount	1	0

	TableName	NewCount
1	AuthenticationEvent	0
2	BuyingGroup	2
3	Category	5
4	City	116294
5	Color	9
6	Continent	7
7	Country	249
8	Customer	402
9	Discount	1

```
SELECT * FROM ViewNumberOfRecords
```

	TableName	Row_Count	previousRowCount
1	AuthenticationEvent	0	0
2	BuyingGroup	2	2
3	Category	5	5
4	City	116294	116294
5	Color	9	9
6	Continent	7	7
7	Country	249	249
8	Customer	402	402
9	Discount	1	0
10	Employee	212	212
11	Invoice	70510	70510
12	InvoiceCustomer	228265	228265
13	NumberOfRecords	25	25
14	Package	6	6
15	Person	615	614
16	PersonType	2	2
17	Product	671	671
18	RecordOfWWI	75	75
19	RecordOfWWI_Previous	75	75

14. Conclusões

Com este relatório pode-se concluir o desenvolvimento e a administração de uma Base de Dados, destinada a uma empresa que trabalha sobre o tema de importações e exportações. Com este relatório pode-se concluir o desenvolvimento e a administração de uma Base de Dados, destinada a uma empresa que trabalha sobre o tema de importações e exportações.

Pode se concluir também, ao longo do projeto, como uma base de dados deve ser organizada em termos de Layouts e ficheiros, assim como devem ser desenvolvidas as suas stored procedures, triggers etc. Pode se concluir também, ao longo do projeto, como uma base de dados deve ser organizada em termos de Layouts e ficheiros, assim como devem ser desenvolvidas as suas stored procedures, triggers etc.

No decorrer da primeira fase o grupo encontrou diversas dificuldades tais como:

- Qual seria a melhor abordagem para o desenvolvimento do modelo de dados;
- Que Stored Procedures seriam significativas para o cliente;
- Quais as Views que seriam pertinentes para o cliente;
- Que tamanho deveriam limitar Files e Filesgroups.

No decorrer da segunda fase, o grupo ainda encontrou alguns desafios que envolveram um desenvolvimento à qualidade da base de dados, e implementação de novas funcionalidades tais como:

- Indexação
- Encriptação de passwords e preços de produtos
- Backups
- Melhoria da base de dados

Estas dificuldades listadas foram algumas de muitas, aos quais o grupo através de pesquisa, e assistência de colegas/professores, conseguiram ultrapassar.