TECNOLOGIA SETÚBAL
ESCOLA SUPERIOR POLITÉCNICO SETÚBAL

Complementos de Bases de Dados

2024/2025

Licenciatura em Engª. Informática

Fase Final Relatório Técnico

Turma: LEI-04
Horário de Laboratório: Seg. 16:30-18:30
Docente: Luís Damas
Grupo
Nº202001541, João Morais
Nº202100067, Lucas Alexandre

1. Introdução

Este relatório técnico apresenta o desenvolvimento de um projeto prático para a disciplina de Complementos de Bases de Dados, com o objetivo de aplicar tópicos avançados de administração e modelação de bases de dados. O projeto centra-se na reestruturação do sistema de gestão de dados da empresa *AdventureWorks*, especializada em material de ciclismo e integrada no grupo *Adventure*, que se encontra em processo de restruturação organizacional.

Atualmente, a gestão das operações da *AdventureWorks* é realizada por meio de um ERP desatualizado, complementado com ficheiros Excel, o que limita a eficiência, a integração e a escalabilidade dos processos de negócio. Como parte das iniciativas de modernização, foi proposta a implementação de um novo ERP que permita uma gestão centralizada e otimizada de todo o processo de vendas da empresa.

Para suportar o novo ERP, é necessário modelar e integrar uma nova base de dados que consolide os fragmentos de informação extraídos do sistema legado e de aplicações complementares utilizadas até então. Os dados fornecidos, provenientes de diferentes fontes, apresentam-se fracamente relacionados, exigindo um processo de modelação alinhado às boas práticas de normalização e de design de bases de dados, com o objetivo de garantir operações eficientes e escaláveis.

O presente relatório detalha todas as etapas do desenvolvimento do projeto, desde a análise e importação dos dados legados até à criação de um modelo de base de dados relacional, incluindo a definição de layouts, migração de dados e implementação de funcionalidades que respondam aos requisitos propostos.

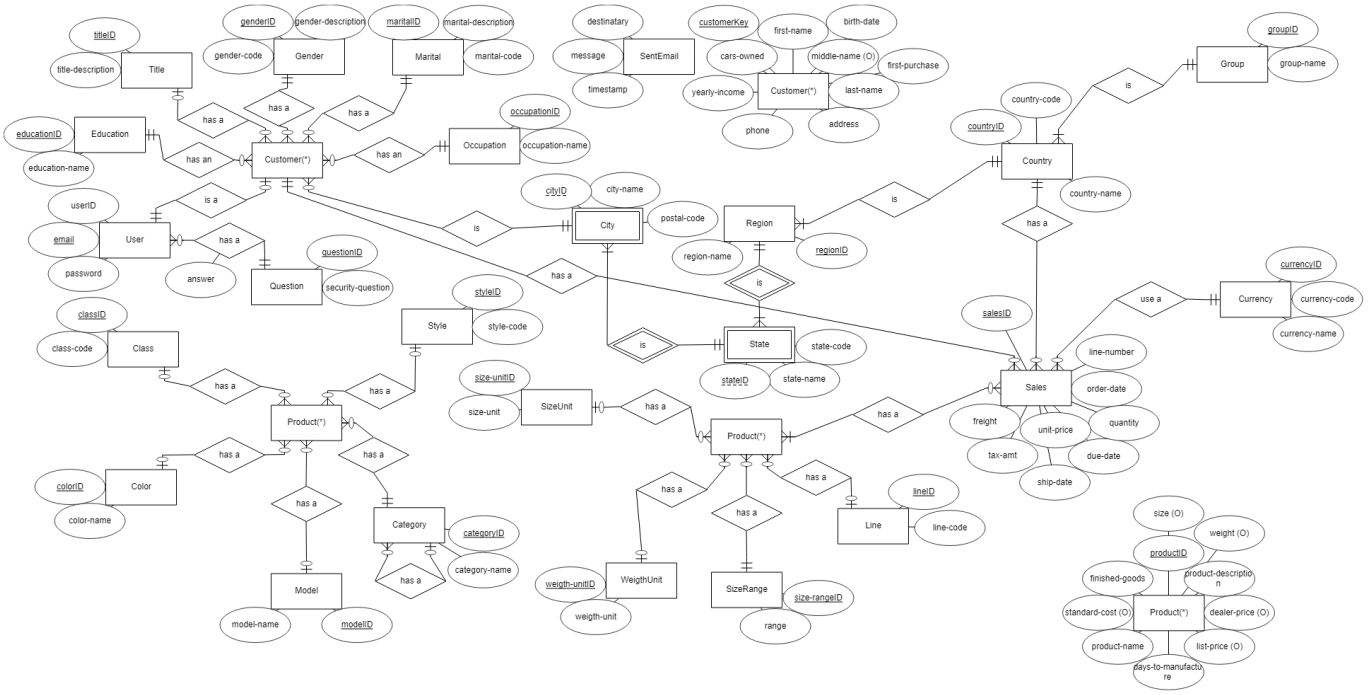
2. Especificação de Requisitos

ID	Descrição	Implementado (S/N)
RQ01	Criar a base de dados “AdventureWorksLegacy” para importar os dados do sistema antigo.	S
RQ02	Analisar os dados importados para inferir entidades de negócio e construir o modelo MER.	S
RQ03	Organizar produtos em subcategorias e categorias gerais para navegação e filtragem.	S
RQ04	Implementar gestão de utilizadores com autenticação por email e senha.	S
RQ05	Suportar recuperação de senha via geração de nova senha e envio simulado de e-mail.	S

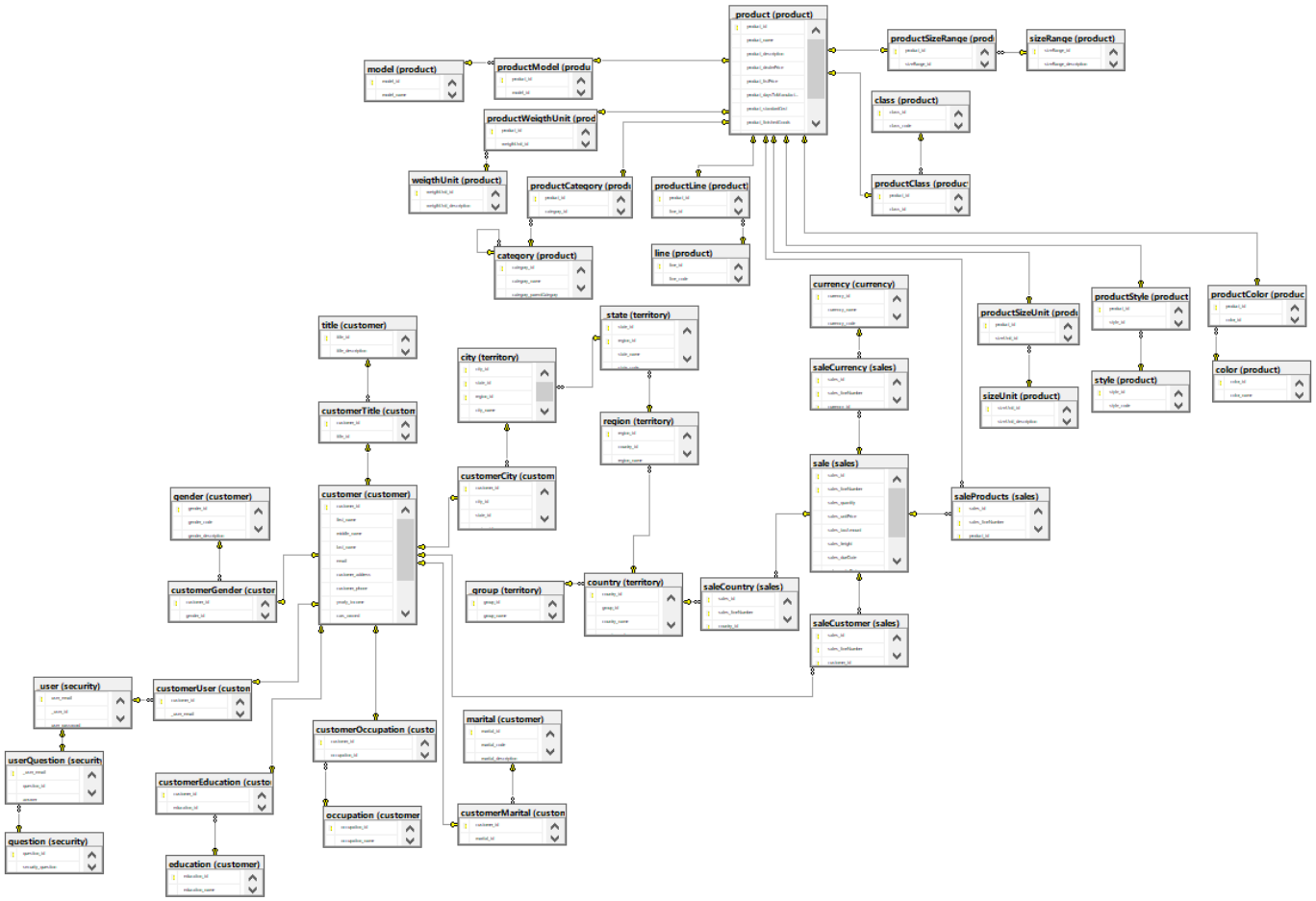
RQ06	Associar uma questão de segurança a cada utilizador para recuperação de senha.	S
RQ07	Criar o diagrama MER usando a ferramenta ERD Plus.	S
RQ08	Converter o MER em um modelo relacional normalizado até a 3ª Forma Normal.	S
RQ09	Determinar o layout da nova base de dados “AdventureWorks” com análise de espaço e uso.	S
RQ10	Criar filegroups com especificações de tamanho inicial, crescimento e limite máximo.	N
RQ11	Migrar os dados da base “AdventureWorksLegacy” para a nova base “AdventureWorks”.	S
RQ12	Garantir que a migração ocorra sem perda de dados.	S
RQ13	Desenvolver queries para validar a conformidade da migração.	S
RQ14	Implementar stored procedures e functions para apoio à migração e gestão de utilizadores e informação de vendas.	S
RQ15	Desenvolver stored procedures para recuperar senha.	S
RQ16	Implementar tratamento de erros centralizado, com logs e mensagens amigáveis.	S
RQ17	Criar a tabela “dbStatistics” para monitorar o número de registos e espaço ocupado.	N
RQ18	Desenvolver a stored procedure “sp_dbstatistics” para manter histórico de estatísticas.	N

3. Modelo Relacional (*Modelo de dados*)

3.1 Diagrama do Modelo Entidade Relação



3.2 Diagrama do Modelo Relacional



4. Definição do Layout

4.1 Identificação do espaço ocupado por tabela

Nome Tabela	Dimensão do Registo	Nº de Registos (inicial/final)
_product	2095Kb	397/397
_group	24Kb	4/3
_state	38	65535/53

_user	60	0/0
category	108	397/41
city	82	6535/336
class	14	4/3
color	104	10/9
country	38	7/6
currency	54	210/210
customer	638	18485/18484
customerCity	16	0/18484
customerEducation	8	0/18484
customerGender	8	0/18484
customerMarital	8	0/18484
customerOccupation	8	0/18484
customerTitle	8	0/101
customerUser	34	0/0
education	34	5/5
errorLog	8220	0/0
gender	29	2/2

line	14	5/4
marital	29	2/2
model	104	119/119
occupation	34	5/5
productCategory	8	0/258
productClass	8	0/312
productColor	8	0/341
productLine	8	0/380
productModel	8	0/397
productSizeRange	8	0/299
productSizeUnit	8	0/253
productStyle	8	0/301
productWeigthUnit	8	0/275
question	204	0/0
region	28	65535/10
sale	61	60398/60398
saleCountry	28	0/60398
saleCurrency	28	0/60398

saleCustomer	28	0/60398
saleProducts	28	0/60398
sentEmail	107	0/0
sizeRange	104	11/10
sizeUnit	14	2/1
style	14	4/3
title	14	7/5
userQuestion	234	0/0
weigthUnit	14	3/2

4.2 Especificação dos FIlegroups

Não foram criados "filegroups" para além dos default da criação da base de dados.

4.3 Schemas

Nome	Descrição
customer	Agrega todas as tabelas relacionadas com customers.
security	Agrega todas as tabelas relacionadas com security.
territory	Agrega todas as tabelas relacionadas com territory.
currency	Agrega todas as tabelas relacionadas com currency.
product	Agrega todas as tabelas relacionadas com products.

sales	Agrega todas as tabelas relacionadas com sales.
error	Agrega todas as tabelas relacionadas com error.

5. Verificação da migração de dados

Foram agregados os pontos 5.1 e 5.2.

```sql
--N◆ de ◆Products◆;
select count(ProductKey) as 'N◆ de produtos' from AdventureWorksLegacy.dbo.Products;
select count(product_id) as 'N◆ de produtos' from AdventureWorks.product._product;

--N◆ de ◆Sales◆;
select count(distinct SalesOrderNumber) as 'N◆ de sales' from AdventureWorksLegacy.dbo.Sales;
select count(distinct sales_id) as 'N◆ de sales' from AdventureWorks.sales.sale;

--Total de vendas por ◆Customer◆;
select CustomerKey as 'Customer', count(distinct SalesOrderNumber) as 'N◆ de sales' from AdventureWorksLegacy.dbo.Sales group by CustomerKey;
select s.customer_id as 'Customer', count(distinct s.sales_id) as 'N◆ de sales' from AdventureWorks.sales.saleCustomer s group by s.customer_id;

--Total monet◆rio de vendas por ano;
select sum(s.OrderQuantity*s.UnitPrice) as 'Total monet◆rio de vendas por ano'
from AdventureWorksLegacy.dbo.Sales s
group by YEAR(s.OrderDate);

select sum(sales_quantity*sales_unitPrice) as 'Total monet◆rio de vendas por ano'
from AdventureWorks.sales.sale group by YEAR(sales_orderDate);

--Total monet◆rio de vendas por ano e por ◆Product◆.
select
s.ProductKey as 'Produto',
sum(s.OrderQuantity*s.UnitPrice) as 'Total monet◆rio de vendas por ano e produto'
from AdventureWorksLegacy.dbo.Sales s
group by YEAR(s.OrderDate), s.ProductKey;


select
sp.product_id as 'Produto',
sum(s.sales_quantity*s.sales_unitPrice) as 'Total monet◆rio de vendas por ano e produto'
from AdventureWorks.sales.sale s
inner join AdventureWorks.sales.saleProducts sp on sp.sales_id = s.sales_id and sp.sales_lineNumber = s.sales_lineNumber
group by YEAR(sales_orderDate), sp.product_id;
```

	Nº de produtos
1	397

	Nº de produtos
1	397

	Nº de sales
1	27659

	Nº de sales
1	27659

	Customer	Nº de sales
1	11000	3
2	11014	2
3	11017	3
4	11020	1
5	11023	2
6	11026	3
7	11029	3
8	11032	3

	Customer	Nº de sales
1	22814	1
2	19897	1
3	14324	3
4	28387	1
5	11407	1
6	15675	3
7	24165	2
8	27036	1

	Total monetário de vendas por ano
1	95421,82
2	48626679,0599996
3	138391,330000001
4	21183296,4099996
5	17582255,46

	Total monetário de vendas por ano
1	95421,82
2	48626679,0599996
3	138391,330000001
4	21183296,4099996
5	17582255,46

Produto	Total monetário de vendas por ano e produto
584	4319,92
483	7800
390	626353,909999999
536	96507,8200000003
378	24433,5
225	233,74
474	60401,37
570	95763,15

Produto	Total monetário de vendas por ano e produto
374	14660,1
578	374173,8
479	224,75
537	420
358	18441,9
584	521630,339999999
381	3001,32
484	21027,75

6. Programação

6.1 Views

Nome	Descrição
product.CategoriasProdutos	Permite listar as categorias e subcategorias de cada produto
product.ListarProdudos	Lista os produtos

6.2 Functions

Não foram desenvolvidas quaisquer funções para a primeira fase.

6.3 Stored procedures

Nome	Atributos	Requisito	Descrição
security.sp_logError	@ErrorMessage nvarchar(4000) @ErrorNumber int @ErrorSeverity int	RQ16	Permite logar um erro
security.sp_addUser	@email char(100) @password char(100) @securityQuestion char(200) @answer varchar(200)	RQ14	Permite criar um user.
security.sp_editUser	@email char(100) @newPassword char(100) @newSecurityQuestion char(200) @newAnswer char(200)	RQ14	Permite atualizar um user.
security.sp_removeUser	@email char(100)	RQ14	Permite apagar um user.
security.sp_receivePass	@email char(100) @securityAnswer char(200)	RQ15	Permite gerar uma nova palavra-passe para um determinado user.

Customer.sp_saleInformation	@orderDate date @customerID int	RQ14	Mostrar toda a informação de uma venda de um cliente, pelo identificador do cliente e a data do pedido

6.4 Triggers

Não foram implementados quaisquer triggers para a primeira fase.

7. Descrição da Demonstração

7.1 Script de demonstração

- Criar base de dados legacy;
- Importar datasets para a legacy;
- Correr script creates.sql;
- Correr script populates.sql;
- Correr script development.sql;
- Correr script queries.sql (opcional);

8. Índices

8.1 Índices

Designação	Tabela	Justificação/Consultas
idx_city_state_id	territory.city	Consulta pesquisa de vendas por cidade
idx_state_region_id	territory._state	Consulta pesquisa de vendas por cidade
idx_region_country_id	territory.region	Consulta pesquisa de vendas por cidade
idx_country_id	territory.country	Consulta pesquisa de vendas por cidade
idx_saleCountry	sales.saleCountry	Consulta pesquisa de vendas por cidade
idx_sale	sales.sale	Consulta pesquisa de produtos associados a vendas com valor total superior a 1000

8.2 Otimização e Execução de Consultas

Pesquisa de vendas por cidade

```
#1    normalizada sem indices
   CPU time = 234 ms,  elapsed time = 452 ms.
   CPU time = 187 ms,  elapsed time = 297 ms.
   CPU time = 203 ms,  elapsed time = 287 ms.

#2    normalizada com indices
    CPU time = 343 ms,  elapsed time = 270 ms.
    CPU time = 342 ms,  elapsed time = 215 ms.
    CPU time = 376 ms,  elapsed time = 243 ms.
```

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

```
#1    normalizada sem indices
    CPU time = 47 ms,  elapsed time = 52 ms.
    CPU time = 31 ms,  elapsed time = 73 ms.
    CPU time = 47 ms,  elapsed time = 50 ms.

#2    normalizada com indices
    CPU time = 31 ms,  elapsed time = 53 ms.
    CPU time = 31 ms,  elapsed time = 54 ms.
    CPU time = 16 ms,  elapsed time = 65 ms.
```

Número de produtos vendidos por categoria

```
#1    normalizada sem indices
    CPU time = 31 ms,  elapsed time = 39 ms.
    CPU time = 31 ms,  elapsed time = 134 ms.
    CPU time = 32 ms,  elapsed time = 32 ms.

#2    normalizada com indices
    CPU time = 16 ms,  elapsed time = 43 ms.
    CPU time = 15 ms,  elapsed time = 56 ms.
    CPU time = 0 ms,  elapsed time = 51 ms.
```

9. Backup e Recuperação

Para garantir a integridade dos dados e minimizar a perda de informação em caso de falhas, o modelo de recuperação escolhido foi o Completo (Full Recovery Model). Este modelo permite:

- Registar todas as transações no log, permitindo uma recuperação precisa até ao momento exato de uma falha.
- Ideal para sistemas críticos onde a perda de dados é inaceitável.

9.1. Tipos de Backup e Estratégia de Armazenamento

Para otimizar os tempos de recuperação e o consumo de recursos, propomos uma estratégia combinada de backup integral, diferencial e do log de transações, com uma política de rotação bem definida:

9.1.1. Backup Integral

- Periodicidade: Uma vez por semana (domingo à noite).
- Objetivo: Criar uma cópia completa dos dados, servindo como base para backups diferenciais e de logs.
- Local de Armazenamento:
    - Armazenamento primário local (servidor dedicado).
    - Armazenamento secundário remoto (cloud ou outro datacenter).

9.1.2. Backup Diferencial

- Periodicidade: Diariamente (exceto no dia do backup integral).
- Objetivo: Capturar apenas as alterações desde o último backup integral, reduzindo o tempo de backup e o espaço necessário.
- Vantagem: Permite uma recuperação rápida ao combinar o último backup integral com o mais recente backup diferencial.

9.1.3. Backup do Log de Transações

- Periodicidade: A cada 15 minutos durante o horário de operação e a cada hora fora desse horário.
- Objetivo: Garantir a possibilidade de recuperação até ao ponto de falha.
- Vantagem: Minimiza a perda de dados em caso de falhas inesperadas.

9.1.4. Política de Rotação

- Armazenamento local: Retenção de backups integrais e diferenciais por 4 semanas.
- Armazenamento remoto: Retenção de backups integrais por 3 meses.
- Logs de transações: Retenção por 1 semana.

9.2. Cenários de Recuperação e Procedimentos

9.2.1. Cenário 1: Falha no Disco Principal do Servidor

Solução:

1. Restaurar o último backup integral.
2. Aplicar o backup diferencial mais recente.
3. Repor os logs de transações sequencialmente até ao momento da falha.

9.2.2. Cenário 2: Eliminação Acidental de Dados

Solução:

4. Identificar o momento da eliminação com base nos logs de transações.
5. Restaurar o último backup integral numa instância secundária.
6. Aplicar os backups diferenciais e logs de transações até ao momento imediatamente anterior à eliminação.

9.2.3. Cenário 3: Corrupção dos Ficheiros de Dados

Solução:

7. Substituir os ficheiros corrompidos pelo último backup integral.
8. Aplicar o backup diferencial mais recente.
9. Repor os logs de transações.

9.3. Otimização do Plano de Backup

Para as tabelas menos alteradas, como _state, _group, city, country, region, currency e gender:

- Realizar um backup inicial integral.
- Utilizar backups diferenciais apenas quando houver alterações.
- Programar um novo backup integral destas tabelas a cada 3 meses, salvo alterações significativas.

10. Segurança e Controlo de Acessos

Foram criados três utilizadores conforme pedido no enunciado.

a. Níveis de acesso à informação

```sql
-- CRIAR CONTAS
create login adminUser with password = 'adworksADM';
create user adminUser for login adminUser;

create login salesUser with password = 'adworksSales';
create user salesUser for login salesUser;

create login territoryUser with password = 'adworksTerritory';
create user territoryUser for login territoryUser;
```

b. Encriptação

```sql
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'AdventureWorksPassword';

CREATE CERTIFICATE RecoveryCert
WITH SUBJECT = 'encriptacao de dados sensiveis';

CREATE SYMMETRIC KEY RecoveryKey
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE RecoveryCert;
```

```sql
drop procedure if exists security.sp_encript;
go
create procedure security.sp_encript
@value char(100),
@hashedValue VARBINARY(MAX) OUTPUT
as
  begin
    begin try
      declare @salt varbinary(16) = cast(100 as varbinary(16)); -- para efeitos demonstrativos usar um salt estatico
      set @hashedValue = hashbytes('SHA2_256', @value + cast(@salt as nvarchar(max)));
    end try
    begin catch
      declare @ErrorMessage nvarchar(4000) = ERROR_MESSAGE();
      declare @ErrorNumber int = ERROR_NUMBER();
      declare @ErrorSeverity int = ERROR_SEVERITY();

      exec security.sp_logError @ErrorMessage, @ErrorNumber, @ErrorSeverity;

      print 'Ocorreu um erro ao encriptar o valor.';
    end catch
  end;
go
```

```
go
CREATE PROCEDURE sp_getRecoveryQuestion (
    @userEmail char(30)
)
AS
BEGIN
    -- Abrir a chave simétrica
    OPEN SYMMETRIC KEY RecoveryKey DECRYPTION BY CERTIFICATE RecoveryCert;

    -- Recuperar e descriptografar os dados
    SELECT
        CONVERT(NVARCHAR(255), DECRYPTBYKEY(q.security_question)) AS Question,
        CONVERT(NVARCHAR(255), DECRYPTBYKEY(uq.answer)) AS Answer
    FROM security._user u
  inner join security.userQuestion uq on u.user_email = uq._user_email
  inner join security.question q on uq.question_id = q.question_id
    WHERE u.user_email = @userEmail;

    -- Fechar a chave simétrica
    CLOSE SYMMETRIC KEY RecoveryKey;
END;
```

11. Controlo de Concorrência

```sql
-- Adicionar produto a uma venda
set transaction isolation level read committed; -- garante que a transa��o n�o leia dados n�o confirmados, evitando problemas como dirty reads.
begin transaction;
  declare @sale_id varchar(20);
  declare @sale_lineNumber int;
  declare @product_id int;

  -- ler de um servi�o web ou aplica��o
  -- neste caso, fazer uma transa��o com dados mock
  set @sale_id = 'SO51176'
  set @sale_lineNumber = (select top 1 (sales_lineNumber + 1) from sales.saleProducts where sales_id = @sale_id
            order by sales_lineNumber DESC);
  insert into sales.sale (sales_id, sales_lineNumber, sales_quantity, sales_unitPrice, sales_taxAmount, sales_freight, sales_dueDate, sales_orderDate, sales_shipDate)
  values (
    @sale_id,
    @sale_lineNumber,
    1,
    10.0,
    0.5,
    0,
    GETDATE(),  -- devia-se ir buscar as datas anteriores e/ou alterar por datas novas
    GETDATE(),  -- devia-se ir buscar as datas anteriores e/ou alterar por datas novas
    GETDATE() -- devia-se ir buscar as datas anteriores e/ou alterar por datas novas
  );

  set @product_id = 212;
  insert into sales.saleProducts (sales_id, sales_lineNumber, product_id)
  values (@sale_id, @sale_lineNumber, @product_id);

commit transaction;
```

```sql
-- Atualizar o pre�o de um produto
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE; -- garante que nenhuma outra transa��o leia ou altere o pre�o do produto enquanto a transa��o est� em andamento.
BEGIN TRANSACTION;
  declare @oldPrice float;
  declare @newPrice float;
  declare @productId int;

  set @productId = 212; -- mock produto
  set @oldPrice = (select p.product_standardCost from product._product p where p.product_id = @productId)
  set @newPrice = @oldPrice * 1.5; -- atualizar o preco

-- Atualizar o pre�o do produto
UPDATE product._product
SET product_standardCost = @newPrice
WHERE product_id = @ProductID;

COMMIT TRANSACTION;
```

```sql
-- Calcular o total de vendas no ano corrente
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ; -- garante que todas as leituras feitas durante a transa��o permane�am consistentes e n�o sejam afetadas por inserts ou updates de outras transa��es.
BEGIN TRANSACTION;
select sum(p.product_listPrice)as 'total de vendas no ano corrente' from sales.saleProducts sp
  inner join sales.sale s on sp.sales_id = s.sales_id and sp.sales_lineNumber = s.sales_lineNumber
  inner join product._product p on sp.product_id = p.product_id
where year(s.sales_orderDate) = year(GETDATE());
COMMIT TRANSACTION;
```

```sql
-- Evitar inser��es duplicadas
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE; -- garante que nenhuma outra transa��o crie um user enquanto esta est� em andamento.

BEGIN TRANSACTION;
  declare @email char(30);

  --set @email = 'jon24@adventure-works.com'; -- email mock, de user que j� existe
  set @email = 'pablo@adventure-works.com'; -- email mock, de user que n�o existe


IF NOT EXISTS (
    SELECT 1 FROM security._user u WHERE u.user_email = @email
)
BEGIN
    exec security.sp_addUser @email, 'umaSenh@MuitoFuerte!', 'qual o nome do meu gato?', 'kiara';
END
COMMIT TRANSACTION;
```

12. Descrição da Demonstração

a. Script de demonstração

Execução de scripts:

- creates.sql
- populates.sql
- seguranca.sql
- criptografia.sql
- transactions.sql
- development.sql
- testing.sql
- queries.sql
- queries_fase_2.sql

13. Conclusões

Este projeto permitiu aplicar conceitos avançados de modelação e administração de bases de dados no contexto da reestruturação da AdventureWorks e implementação de um novo ERP. Desde a integração e migração de dados

legados até à criação de um modelo relacional normalizado, as soluções desenvolvidas garantem eficiência, escalabilidade e suporte às operações empresariais.

Adicionalmente, foi implementada uma estratégia robusta de backup e recuperação, que assegura a integridade e disponibilidade dos dados, mesmo em cenários de falha. O controlo de transações e os níveis de isolamento definidos garantem a consistência dos dados em situações de acesso concorrente, enquanto as políticas de encriptação aplicadas reforçam a segurança da informação sensível, como passwords e dados de recuperação.

As funcionalidades implementadas, como gestão de utilizadores, monitorização e tratamento de erros, em conjunto com uma abordagem otimizada para a segurança e recuperação de dados, reforçam a confiabilidade e resiliência do sistema. Assim, este projeto demonstra a importância de bases de dados bem projetadas, acompanhadas de boas práticas de administração e segurança, para atender às necessidades de um ambiente empresarial moderno, competitivo e em constante evolução.