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:
Horário de Laboratório:
Docente:
Grupo
Nº202100984, Francisco Silva
Nº202100296, Gonçalo Vieira
Nº202100299, Rui Barroso

2ª Fase Relatório Técnico – Complementos de Bases de Dados

1. Introdução

No âmbito da disciplina de Complementos de Bases de Dados, foi nos pedido para o desenvolvimento de uma base de dados com dados migrados de outra, responsável por gerir uma empresa de vendas.

Neste relatório vamos apresentar a nossa proposta do Modelo Entidade-Relação, o nosso diagrama, e o nosso Modelo relacional, as funcionalidades da nossa base de dados e os detalhes tecnicos.

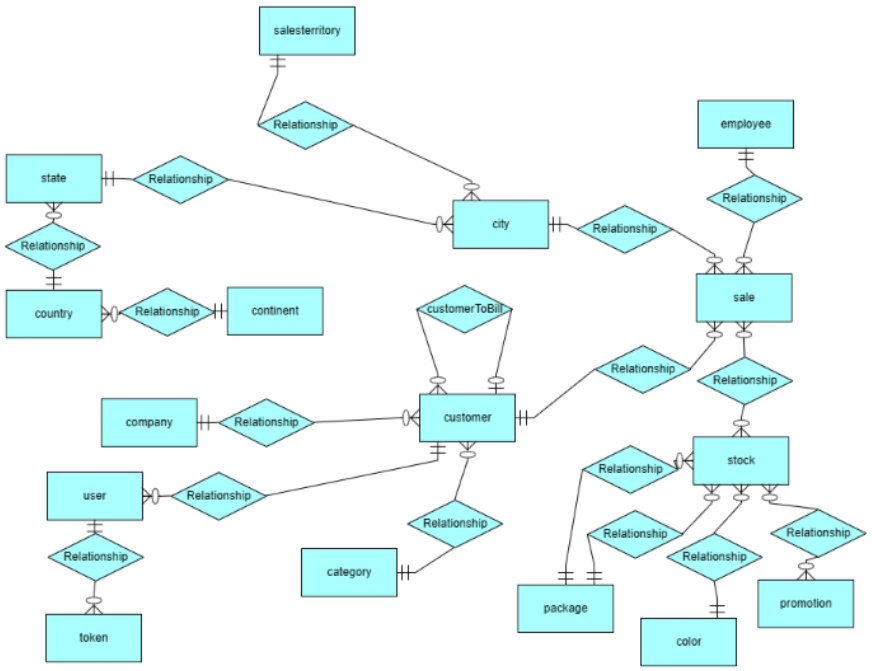
Especificação de Requisitos

Especificar os requisitos funcionais apresentados no enunciado do projeto. Estes requisitos devem incluir restrições de integridade ou regras de validação de informação/processos de negócio. Os requisitos que são propostos como melhoria aos expostos no enunciado devem ser incluídos e identificados por RM##.

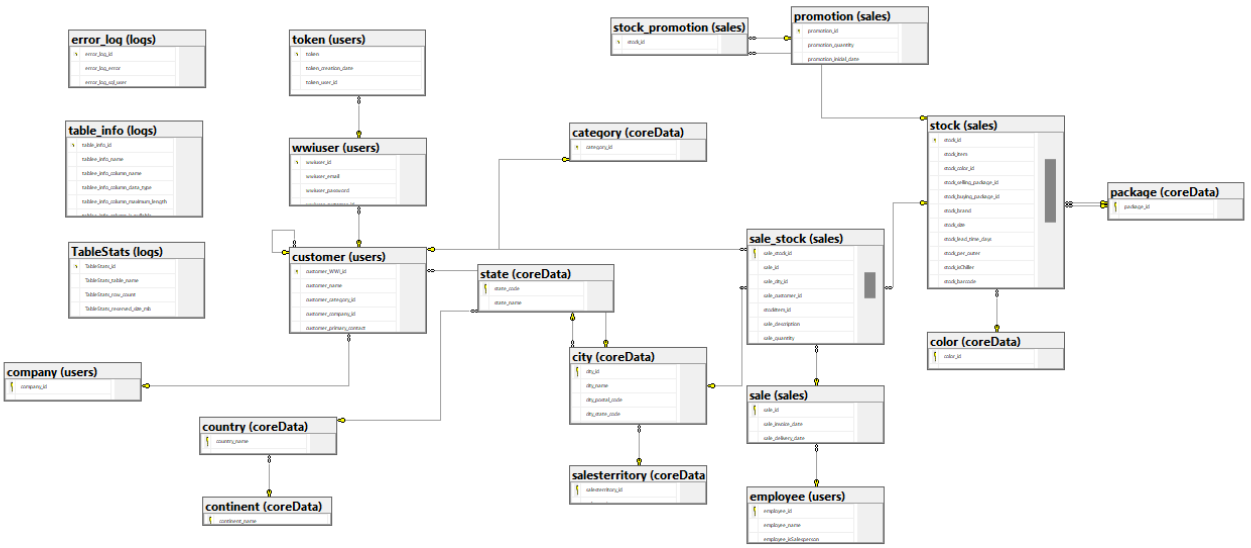
ID	Descrição	Implementado (S/N)
R01	O sistema deverá permitir *gerar automaticamente um token para Recuperar Password*	S
R02	*Definir o conceito de promoção sobre um ou mais produtos*	S
R03	*Alterar a quantidade de um produto numa venda*	S
R04	Criar uma venda	S
R05	Adicionar um produto a uma venda;	S
R06	Remover um produto de uma venda. Recebe um parâmetro que indica se a venda é removida no caso de não ter mais produtos associados;	S
R07	Calcular o preço total de uma venda;	S
R08	Implementar a regra de negócio que verifique se a data de entrega está de acordo com o tempo previsto de entrega de um produto (“Lead Time Days”);	S
R09	Não permitir uma venda conter produtos com e sem “Chiller Stock”.	S

2. Relacional (*Modelo de dados*)

2.1 Diagrama do Modelo Entidade Relação



2.2 Diagrama do Modelo Relacional



2.3 Identificação do espaço ocupado por tabela

Nome Tabela	Dimensão do Registo	Nº de Registos (inicial/final)
coreData.color	31	9
coreData.package	31	6
coreData.category	31	5
coreData.continent	30	1
coreData.country	80	1
coreData.state	102	57
coreData.salesterritory	54	9
coreData.city	98	116294
dbo.sysdiagrams	267	0

users.company	32	2
users.employee	64	212
users.customer	115	402
users.wwiuser	117	1
users.token	21	1
sales.promotion	11	0
dbo.record_size	3	1
sales.stock	229	671
sales.sale	18	70510
sales.stock_promotion	8	0
sales.sale_stock	197	228265
logs.error_log	66	2
logs.TableStats	288	0
logs.table_info	879	0

2.4 Especificação dos Filegroups

Nome Filegroup	Tabelas associadas	Parâmetros
readOnly_filegroup	city continent country category	SIZE = 10000KB

	package color salesterritory state	
intensive_filegroup	sale_stock sale stock stock_promotion promotion	SIZE = 10 MB, FILEGROWTH = 2 MB, MAXSIZE = 100MB
log_filegroup	error_log table_info TableStats	SIZE = 1 MB, FILEGROWTH = 1 MB, MAXSIZE = 10MB
PRIMARY	company employee customer token wwiuser	SIZE = 1 MB, FILEGROWTH = 1 MB, MAXSIZE = 10MB

2.5 Schemas

Nome	Descrição
sales	Contem as tabelas das vendas
users	Contem as tabelas dos utilizadores
coreData	Contem as tabelas de dados mais basicos
log	Contem tabelas de gestão da Base de dados

3. Verificação da migração de dados

3.1 Consultas sobre a base de dados original

```
--Nº de Customers // Antiga

select count(*)

from WWI_DS.dbo.customer


--Nº de Customers por Categoria // Antiga

select Category, count(*) as 'Number of customers'
```

```
from WWI_DS.dbo.customer
group by Category;


--Total de vendas por Employee // Antiga
select e.[Employee Key], count(distinct s.[WWI Invoice ID]) as 'Number of sales'
from WWI_DS.dbo.employee e inner join WWI_DS.dbo.sale s
on e.[Employee Key] = s.[Salesperson Key]
group by e.[Employee Key]
order by e.[Employee Key];

--Total monetário de vendas por StockItem // Antiga
select s.[Stock Item Key], sum(s.Quantity*s.[Unit Price]) as 'Sales Value'
from WWI_DS.dbo.[Stock Item] st inner join WWI_DS.dbo.sale s
on st.[Stock Item Key] = s.[Stock Item Key]
group by s.[Stock Item Key]
order by s.[Stock Item Key];


--Total monetário de vendas por ano por StockItem // Antiga
select s.[WWI Invoice ID], year(s.[Delivery Date Key])as Year, sum(s.Quantity * st.[Unit Price]) as 'Sales Value'
from WWI_DS.dbo.[Stock Item] st inner join WWI_DS.dbo.sale s
on st.[Stock Item Key] = s.[Stock Item Key]
group by year(s.[Delivery Date Key]), s.[WWI Invoice ID]
order by s.[WWI Invoice ID], year(s.[Delivery Date Key]);


--Total monetário de vendas por ano por City // Antiga
select c.City, year([Delivery Date Key]), sum(s.Quantity * st.[Unit Price]) as 'Sales Value'
```

```
from WWI_DS.dbo.[Stock Item] st inner join WWI_DS.dbo.sale s
on st.[Stock Item Key] = s.[Stock Item Key]
inner join WWI_DS.dbo.City as c on c.[City Key] = s.[City Key]
group by year(s.[Delivery Date Key]), c.City
order by c.City, year(s.[Delivery Date Key]);
```

3.2 Consultas sobre a nova base de dados

```
--Number of customer // Nova
select count(*)
from WWIGlobal.users.customer;

--Nº de Customers por Categoria // Nova
select cat.category_name, count(*) as 'Number of customers'
from WWIGlobal.users.customer c inner join WWIGlobal.coreData.category cat
on c.customer_category_id = cat.category_id
group by cat.category_name;


--Total de vendas por Employee // Nova
select e.employee_id ,count(*) as 'Number of sales'
from WWIGlobal.users.employee e inner join WWIGlobal.sales.sale s
on e.employee_id = s.sale_salesperson_id
group by e.employee_id
order by e.employee_id;


--Total monetário de vendas por StockItem // Nova
select stockItem_id, sum(sale_quantity*sale_unit_price) as 'Sales Value'
from WWIGlobal.sales.sale_stock
```

```
group by stockItem_id
order by stockItem_id;


--Total monetário de vendas por ano por StockItem // Nova
select s.sale_id, year(sale_delivery_date) as Year, sum(sale_quantity * sale_unit_price) as 'Sales Value'
from WWIGlobal.sales.sale_stock st inner join WWIGlobal.sales.sale s on st.sale_id = s.sale_id
group by year(sale_delivery_date), s.sale_id
order by s.sale_id, year(sale_delivery_date);


--Total monetário de vendas por ano por City // Nova
select city_name, year(sale_delivery_date), sum(sale_quantity * sale_unit_price) as 'Sales Value'
from WWIGlobal.sales.sale_stock st inner join WWIGlobal.sales.sale s on st.sale_id = s.sale_id
inner join WWIGlobal.coreData.city on city_id = sale_city_id
group by year(sale_delivery_date), city_name
order by city_name, year(sale_delivery_date);
```

4. Programação

4.1 Stored procedures

Nome	Atributos	Requisito	Descrição
sp_new_password_token_generator	@id int	R01	Gerar um token
sp_create_sale	@invoice_date date, @delivarie_date date, @employee_id	R04	Criar uma venda

	int, @sale_id int OUT		
sp_insert_product_to_sale	@sale_id int, @sale_city_id int, @sale_customer varchar(50), @stockItem_id int, @sale_description varchar(150), @sale_quantity int, @sale_unit_price float, @sale_tax_rate float, @sale_stock_id int out	R05	Adicionar um stock item a uma venda
sp_alter_sale_stock_quantity	@id int, @new_quantity int	R03	alterar a quantidade de um stock item em uma venda
sp_calc_total_price	@sale_id int	R07	Calcular o preço total de uma venda
sp_check_business_rule	@sale_id int, @isInTime bit out	R08	Implementar a regra de negócio que verifique se a data de entrega está de acordo com o tempo previsto de entrega de um produto (“Lead Time Days”) ‘
sp_sale_stock_remove_product	@id int, @removeSale bit	R06	Remover um produto de uma venda. Recebe um parâmetro que indica se a venda é removida no caso de não ter mais produtos associados

4.2 Triggers

Nome	Tipo	Tabela	Requisito	Descrição
trigger_check_promotion_is_active	AFTER INSERT, UPDATE	sales.stock_promotion	R02	Verifica se a promoção está ativa
tr_check_sale_stock_chiller	AFTER INSERT, UPDATE	sales.sale_stock	R09	Verifica se a venda contem produtos com e sem “Chiller Stock”

5. Catálogo/Metadados

5.1 Geradores

Nome	Atributos	Descrição
generate_insert_sp	@table_name varchar(255) @schema_name varchar(255)	Implementa o procedimento para inserir registos numa tabela
generate_update_sp	@table_name varchar(255), @schema_name varchar(255)	Implementa o procedimento para atualizar registos numa tabela
generate_delete_sp	@table_name varchar(255), @schema_name varchar(255	Implementa o procedimento para remover registos numa tabela
GenerateStoredProcedures		Usa as 3 outras procedures generaters

5.2 Monitorização

Nome	Atributos	Descrição
sp_generate_table_info		Recorre ao catalogo para gerar entradas numa tabela(s) dedicada(s) onde deve constar a seguinte informação relativa à base de dados:

		todos os campos de todas as tabelas, com os seus tipos de dados, tamanho respetivo e restrições associadas
sp_TableStats		Regista em tabela dedicada, por cada tabela da base de dados o seu número de registos e estimativa mais fiável do espaço ocupado.

6. Índices

6.1 Views

Nome	Descrição
v_sale_quantity_by_year_and_category	Esta view permite a ver a quantidade de vendas de cada ano, por categoria de cliente
v_sale_quantity_by_year_and_category_old	Esta view permite a ver a quantidade de vendas de cada ano, por categoria de cliente da velha base de dados

6.2 Índices

Designação	Tabela	Justificação/Consultas
city_index	city	Uma tabela que não tem muitas inserções e romossões e usada muitas vezes em condições where
color_index	color	Uma tabela que não tem muitas inserções e romossões e usada muitas vezes em condições where

7. Backup e Recuperação

Modelo de Recuperação

Inicialmente começamos com um Backup completo e depois fazemos todas as semanas para não acumular demasias diferenças no Backup diferencial.

Todos os dias fazemos um Backup diferencial guardando as transações diminuindo o número de Backup completos que demoram mais tempo.

Backup dos logs transacionais de hora em hora.

Caso haja um crash

Fazemos o Backup do tail de logs, recuperamos o Backup completo, de seguida recuperamos o Backup diferencial, depois o Backup dos logs transacionais e por ultimo o Backup do tail de logs.

8. Segurança e Controlo de Acessos

Definição de Utilizadores, Roles, Schemas e Encriptação.

8.1 Níveis de acesso à informação

Criação de roles:

CREATE ROLE Admin;

CREATE ROLE EmployeeSalesPerson;

CREATE ROLE SalesTerritory;

Criação dos utilizadores:

CREATE USER AdminUser Without LOGIN;

CREATE USER EmployeeSalesPersonUser Without LOGIN;

CREATE USER SalesTerritoryUser Without LOGIN;

Administrador: Tem acesso a toda a informação:

GRANT SELECT, UPDATE, DELETE, EXECUTE, VIEW DEFINITION, ALTER, CONTROL, TAKE OWNERSHIP TO Admin;

EmployeeSalesPerson: Tem acesso total às tabelas de suporte às vendas, e apenas acesso em modo de consulta às restantes tabelas:

GRANT SELECT, UPDATE, DELETE, EXECUTE, VIEW DEFINITION, ALTER, CONTROL, TAKE OWNERSHIP ON SCHEMA::sales TO EmployeeSalesPerson;

```
GRANT SELECT ON coreData.color TO EmployeeSalesPerson;
GRANT SELECT ON coreData.package TO EmployeeSalesPerson;
GRANT SELECT ON coreData.category TO EmployeeSalesPerson;
GRANT SELECT ON coreData.continent TO EmployeeSalesPerson;
GRANT SELECT ON coreData.country TO EmployeeSalesPerson;
GRANT SELECT ON coreData.state TO EmployeeSalesPerson;
GRANT SELECT ON coreData.salesterritory TO EmployeeSalesPerson;
GRANT SELECT ON coreData.city TO EmployeeSalesPerson;
GRANT SELECT ON users.company TO EmployeeSalesPerson;
GRANT SELECT ON users.employee TO EmployeeSalesPerson;
GRANT SELECT ON users.customer TO EmployeeSalesPerson;
GRANT SELECT ON users.wwiuser TO EmployeeSalesPerson;
GRANT SELECT ON users.token TO EmployeeSalesPerson;
GRANT SELECT ON logs.error_log TO EmployeeSalesPerson;
GRANT SELECT ON logs.TableStats TO EmployeeSalesPerson;
GRANT SELECT ON logs.table_info TO EmployeeSalesPerson;
```

SalesTerritory: Apenas pode consultar a informação relativa ao seu território. Considere apenas o território "Rocky Mountain":

```
go
DROP VIEW IF EXISTS viewAux
go
CREATE VIEW viewAux AS
SELECT salesterritory_name
FROM coreData.salesterritory
WHERE salesterritory_name = 'Rocky Mountain';
go

GRANT SELECT ON viewAux TO SalesTerritory;
```

8.2 Encriptação

As passwords são encriptadas através da função hash.

Os preços e os tracking number dos transportes são encriptados com um par de chaves.

9. Controlo de Concorrência

• Adicionar Produto a uma Venda;

Usamos REPEATABLE READ;

• Atualizar preço de um produto, garantindo que o preço do produto nas vendas por finalizar não é alterado;

Usamos REPEATABLE READ

• Calcular o total da venda e/ou a quantidade de produtos na venda sem permitir adição ou remoção de produtos na venda.

Usamos SERIALIZABLE;

10. MongoDB

Fizemos querys para calcular os valore requisitados

Para Importar os dados usamos o mongoDB compass

Para exportar os dados exportamos os resultados da seguinte query para um file json

db.transports.aggregate([

```
  {
    $unwind: "$transport"

  },{
    $addFields: {
      "transport.name": "$name"
    }
  },
```

```
  {
    $replaceRoot: {
      newRoot: "$transport"
    }
  }
])
```

11. Descrição da Demonstração

11.1 Script de demonstração sobre a base de dados relacional

createDataBase.sql

tablesCreation.sql

migration.sql

verification.sql

tablesSize.sql

catalog.sql

procedures.sql

indices.sql

backup.sql

roles.sql

concorrencia.sql

export_json.sql

encryption.sql

import_json.sql

test.sql

11.2 Script de demonstração sobre a base de dados NoSQL

mongoDB_query

12. Conclusões

Após a conclusão do projecto, tivemos alguma dificuldade com a MongoDB, mais na parte das queries, e do catálogo, tivemos também dificuldade ao criar as procedures que geram código, a transformação das tabelas para JSON.

Sentimos que estamos mais aptos e confortáveis a trabalhar com base de dados, pois conseguimos ultrapassar vários desafios e melhoramos na sua utilização.