



JACKSON JÚNIOR

LEONARDO LOPEZ

PROJETO: INTEGRAÇÃO DE FLUXOS DE INFORMAÇÃO (SANITOP 360)

GUIA DE IMPLEMENTAÇÃO

VIANA DO CASTELO

2022

JACKSON BARRETO COSTA JÚNIOR
LEONARDO MARIZ MARTINEZ LOPEZ

PROJETO: INTEGRAÇÃO DE FLUXOS DE INFORMAÇÃO (SANITOP 360)
GUIA DE IMPLEMENTAÇÃO

Guia de implementação sobre o trabalho realizado através do estágio extracurricular na empresa Sanitop no curso de Licenciatura em Engenharia Informática do Instituto Politécnico de Viana do Castelo – IPVC.

Orientação: Professor doutor Pedro Coutinho, Professora doutora Sara Paiva, e Professor doutor Ricardo Freitas.

VIANA DO CASTELO

2022

SUMÁRIO

1.	Introdução	5
2.	Planeamento da arquitetura	5
3.	Implementação	11
3.1.	Configuração dos nodos	12
3.1.1.	engine.name	12
3.1.2.	external.id	12
3.1.3.	group.id	12
3.1.4.	sync.url	12
3.1.5.	registration.url	12
3.1.6.	db.driver	13
3.1.7.	db.url	13
3.1.8.	db.user	13
3.1.9.	db.password	13
3.2.	Configuração da sincronização	14
3.2.1.	Links de comunicação	16
3.2.2.	Definindo Rotas	17
3.2.3.	Canais	18
3.2.4.	Definindo Triggers	19
3.2.5.	Roteamento dos Triggers	20
3.2.6.	Transformações	21
4.	Anexo	25
4.1.	Script SQL de instalação	25

4.1.1. Configuração dos Nodos	25
4.1.2. Configuração dos links.....	26
4.1.3. Configuração do roteamento	27
4.1.4. Configuração dos canais	28
4.1.5. Configuração dos triggers	29
4.1.6. Configuração do roteamento dos triggers.....	30
4.1.7. Transformação de tabelas.....	31
4.1.8. Transformação de colunas.....	42

1. INTRODUÇÃO

O presente guia de implementação tem por objetivo servir como orientação para a implementação do sistema de sincronização das bases de dados da empresa **Sanitop**, através do software **SymmetricDS**.

O guia discorrerá sobre as etapas necessárias para se realizar a integração de fluxos de informação através do SymmetricDS, considerando o contexto do ambiente de produção da Sanitop.

É importante ressaltar que este documento não tem a pretensão de substituir a documentação oficial do SymmetricDS, consistindo apenas em um relatório sobre pesquisas empíricas sob as diretrizes da documentação oficial. Portanto, para esclarecimentos mais pontuais, recomenda-se vivamente revisar a documentação oficial.

2. PLANEAMENTO DA ARQUITETURA

Em uma arquitetura de solução de integração de fluxo de informações, é importante identificar o sentido em que os dados devem fluir; as redes e servidores nos quais as bases de dados estão provisionadas; e as versões dos SGBDs (Sistema de Gerenciamento de Base de Dados) em execução.

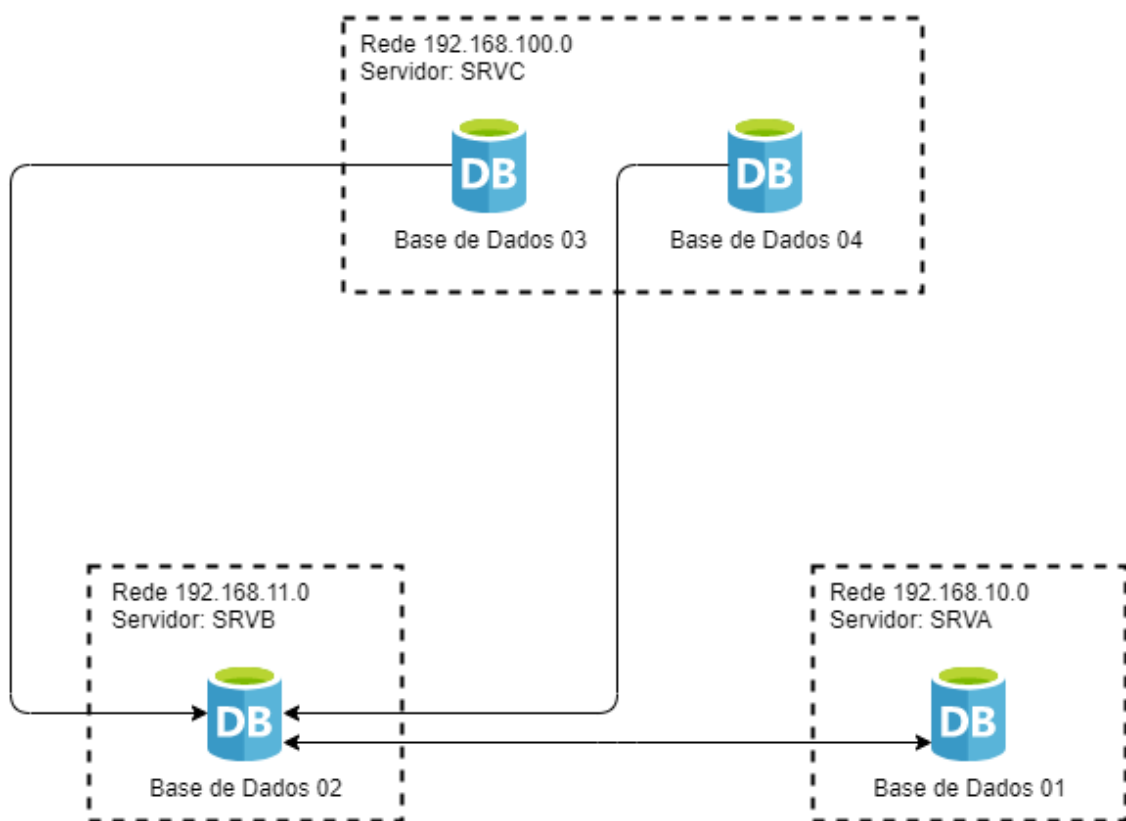


Figura 1 - Topologia e fluxo de informações.

A Figura 1 mostra a rede em que cada base de dados está provisionada, assim como o fluxo de informações/dados, isto é, o sentido no qual a informação precisa fluir. Note que entre os servidores SRVA e SRVB há um canal de dados bidirecional, enquanto as duas instâncias de base de dados que estão executando no mesmo servidor SRVC (situação incomum) enviam dados para o SRVB por um canal unidirecional.

A identificação das redes é extremamente útil na fase de implementação para definição das políticas de segurança da rede, como quais portas devem ser abertas nos Firewalls.

Na sequência, deve ser elaborado um mapa de correlação das tabelas a serem sincronizadas, ou seja, identificar as tabelas da origem que precisam ser sincronizadas

com as tabelas do destino. Isto é necessário porque uma tabela de origem pode ter um nome diferente em seu destino e conter apenas algumas das colunas da origem.

CORRELAÇÃO DE TABELAS		
#	Base de Dados 03	Base de Dados 02
01	vendas	vendas_centralizada
02	pedidos	confirmados
03	retomas	retomas

Tabela 1 - Correlação de tabelas

A Tabela 1 exemplifica uma sincronização entre a *base de dados 03* e a *02*, onde as tabelas *vendas* e *pedidos* tem seu nome alterado, enquanto a tabela *retoma* mantém o nome.

CORRELAÇÃO DE COLUNAS		
#	Base de Dados 03 - vendas	Base de Dados 02 – vendas_centralizada
01	qty	despachos
02	date	ocorrencia
#	Base de Dados 03 - pedidos	Base de Dados 02 – confirmados
01	cliente	associado
02	data	data

Tabela 2 - Correlação de colunas

Já a Tabela 2 mapeia as colunas que sofrem alteração de nome ou que devem ser consideradas na operação. Observe que a tabela *retomas* não foi citada, o que

significa que esta tabela será sincronizada integralmente, ou seja, terá todas as suas colunas consideradas.

Outra informação relevante levantada nessa fase preliminar é a versão do SGBD em execução na base de dados. Com essa informação, pode-se definir corretamente os drives JDBC necessários para a implementação.

Cabe salientar que o SymmetricDS vem acompanhado com alguns drives JDBC, entretanto, pode ocorrer alguma incompatibilidade entre as versões em execução e as disponibilizadas pelo aplicativo, como falhas de autenticação ou reconexões excessivas. Justamente por esse possível problema é importante documentar previamente a versão do SGBD para selecionar o drive correto.

REGISTO DE SGBD		
Base de Dados	SGBD	Versão
Base de Dados 01	Microsoft SQL Server 2019	15.2.1
Base de Dados 02	Postgre SQL 9.6	9.6.24
Base de Dados 03	Microsoft SQL Server 2016	12.5.7

Tabela 3 - Registo de SGBD

A Tabela 3 identifica o SGBD e a versão corrente de cada uma das bases de dados que compõem o processo de integração.

Finalmente, deve-se definir um agrupamento transacional, que tem como finalidade identificar todas as tabelas que fazem parte de uma mesma transação, mesmo aquelas que não estão relacionadas por chave estrangeira, mas sim por uma regra de negócio que gera uma transação.

Essa informação é importante para agrupar essas tabelas no mesmo **canal de comunicação**, caso contrário ocorrerão erros no momento de replicar os dados, por inconsistência dos dados.

REGISTO DE TRANSAÇÕES	
Base de dados 01	
Transação 01	
#	Tabelas
01	qty
02	date

Tabela 4 - Registo de transações

A Tabela 4 demonstra que as tabelas *qty* e *date* fazem parte de uma mesma transação, e por isso devem ser sincronizadas em conjunto, de modo a manter o seu relacionamento e integridade dos dados.

Por fim deve ser identificado os agrupamentos de base de dados.

#	Base de dado	Porta	Grupo
01	Filial01	8443	Filial
02	Filial02	8443	Filial
03	Filial03	8443	Filial
04	Filial04	8443	Filial
05	Central	8443	Central
06	financeiro	8443	Financeiro

Tabela 5 - Agrupamento de base de dado

A Tabela 5 permite-nos observar que as bases de dados *filial01*, *filial02*, *filial03* e *filial04* pertencem ao mesmo grupo, enquanto a *central* e *financeiro* estão individualizadas.

As bases de dados das filiais não necessitam de estar no mesmo servidor, para estarem no mesmo agrupamento (como se observa pelo uso da mesma porta TCP). O agrupamento se justifica pelo contexto, ou seja, todas as filiais devem receber as mesmas atualizações de dados da central. A definição, neste caso, é que a central fornece dados para todas as suas filiais.

Reunidas todas estas informações o projeto encontra-se apto a seguir para a fase de implementação.

3. IMPLEMENTAÇÃO

3.1. INSTALAÇÃO

A instalação do SymmetricDS consiste em, após baixado o ficheiro ZIP no site oficial, descompactar do ficheiro na diretoria onde instalar o SymmetricDS.

Habitualmente a porta TCP padrão, utilizada pelo SymmetricDS, não é utilizada por outras aplicações, em caso de necessidade de alterar a porta TCP basta aceder ao ficheiro `conf/symmetric-server.properties`, a partir da diretoria de instalação.

```
26  # Enable synchronization over HTTP.
27  #
28  http.enable=true
29
30  # Port number for synchronization over HTTP.
31  #
32  http.port=31415
33
```

Figura 2 - Configuração da porta de acesso

A Figura 2 exhibe o atributo para configurar a porta TCP na qual o SymmetricDS irá operar. Destaca-se que a porta TCP definida deve ser informada nos ficheiros de configuração, como parte da URL de registro e sincronização, conforme demonstrado no tópico seguinte (configuração dos nodos).

Por fim, deve-se configurar o SymmetricDS para ser executado como um serviço do Windows. Para tal, deve-se executar, **com permissões de administrador**, o ficheiro `bin/sym_service.bat install`, a partir da diretoria de instalação.

Desta forma, o serviço de sincronização iniciará automaticamente quando o sistema inicializar, sendo executado como um serviço do Windows. Um processo de *wrapper* inicia o SymmetricDS e o monitora, para que possa ser reiniciado se ficar sem memória ou sair inesperadamente.

3.2. CONFIGURAÇÃO DOS NODOS

De modo a obter a melhor performance o SymmetricDS deve ser instalado no mesmo servidor onde está a base de dado a ser monitorada.

Dentro do contexto do SymmetricDS, cada base de dado é representada como um nodo. O nodo é definido por um ficheiro de extensão "*properties*" que deve ser armazenado na diretoria "*engine/*", e possui as seguintes configurações obrigatórias:

3.2.1. **engine.name**

É o nome utilizado para acessar o servidor SymmetricDS deste nodo, através da URL de comunicação.

No caso do *engine.name* ser *DynamamanWMS* a URL de acesso seria:
`http://srvsql:8443/sync/DynamamanWMS`

3.2.2. **external.id**

É um nome lógico, que possui significado no contexto de negócio onde está sendo implementado, como *filial01* ou *financeiro*.

3.2.3. **group.id**

O grupo de nós do qual esse nó é membro, para esta configuração basta recorrer a sua tabela de agrupamento de base de dados (Tabela 5).

3.2.4. **sync.url**

É a URL utilizada pelos servidores SymmetricDS para comunicação. A URL de sincronização tem o formato: `http://{hostname}:{port}/sync/{engine.name}`

3.2.5. **registration.url**

Esta é a URL do servidor de configuração SymmetricDS, que é o servidor de registro onde se encontram as configurações de sincronização de todo o projeto.

Cada nodo se registra neste servidor e passa a receber as configurações e atualizações para o funcionamento do sistema de sincronização.

3.2.6. db.driver

O nome da classe Java do driver JDBC.

Exemplo: com.microsoft.sqlserver.jdbc.SQLServerDriver

3.2.7. db.url

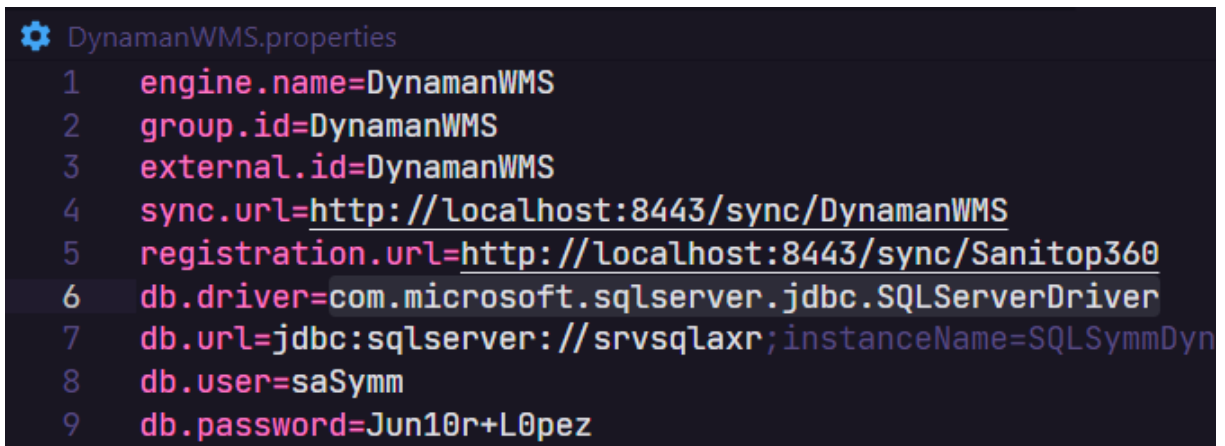
A URL JDBC usada para se conectar ao banco de dados.

3.2.8. db.user

O nome de usuário do banco de dados, que é usado para efetuar login, criar e atualizar tabelas. Recomenda-se que seja criado um utilizador exclusivamente para o SymmetricDS.

3.2.9. db.password

A senha para o usuário do banco de dados.



```
1 engine.name=DynamanWMS
2 group.id=DynamanWMS
3 external.id=DynamanWMS
4 sync.url=http://localhost:8443/sync/DynamanWMS
5 registration.url=http://localhost:8443/sync/Sanitop360
6 db.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
7 db.url=jdbc:sqlserver://srvsq1axr;instanceName=SQLSymmDyn
8 db.user=saSymm
9 db.password=Jun10r+L0pez
```

Figura 3 - Exemplo de configuração de nodo

A Figura 3 apresenta um ficheiro de configuração de um nodo SymmetricDS, com a configuração mínima necessária para sua execução.

3.3. CONFIGURAÇÃO DA SINCRONIZAÇÃO

O primeiro passo para configurar a sincronização é criar os nodos propriamente ditos na base de dados do servidor de registro. Isso irá permitir que todos os nodos do sistema recebam as configurações que lhes dizem respeito.

Por isso, os scripts SQL que configuram o SymmetricDS devem ser executados na base de dados do nodo mestre (servidor de registro).

```

1 | -- CONFIGURAÇÃO DOS NODOS
2 | INSERT INTO SYM_NODE_GROUP
3 | (
4 |     node_group_id,
5 |     description,
6 |     create_time,
7 |     last_update_time,
8 |     last_update_by
9 | )
10 | VALUES
11 | (
12 |     'DynamanWMS',
13 |     'IBS Dynaman WMS - SQL Server',
14 |     current_timestamp,
15 |     current_timestamp,
16 |     'Installation'
17 | );
18 |
19 | INSERT INTO SYM_NODE_GROUP
20 | (
21 |     node_group_id,
22 |     description,
23 |     create_time,
24 |     last_update_time,
25 |     last_update_by
26 | )
27 | VALUES
28 | (
29 |     'Sanitop360',
30 |     'Sanitop 360 - SQL Server',
31 |     current_timestamp,
32 |     current_timestamp,
33 |     'Installation'
34 | );
35 |

```

Figura 4 - Configuração dos nodos

A Figura 4 exibe um código SQL para configuração dos nodos.

Observe que os campos *create_time*, *last_update_time* e *last_update_by*, apesar de serem opcionais, são de extrema importância para manter o sistema auditável. Em

nosso exemplo utilizamos o valor *"installation"* para representar que estas definições foram realizadas na instalação.

3.3.1. Links de comunicação

Realizada a definição dos nodos é o momento de configurar os links de comunicação entre eles. Nesta etapa se está a implementar as definições estabelecidas na Figura 1, ou seja, os fluxos de comunicação.

```
-- CONFIGURAÇÃO DOS LINKS (CONEXÃO ENTRE GRUPOS)
-- verificar o data_event_action correto para cada link
INSERT INTO SYM_NODE_GROUP_LINK
(
    source_node_group_id,
    target_node_group_id,
    data_event_action,
    create_time,
    last_update_time,
    last_update_by
)
VALUES
(
    'DynamamanWMS',
    'Sanitop360',
    'P',
    current_timestamp,
    current_timestamp,
    'Installation'
);

INSERT INTO SYM_NODE_GROUP_LINK
(
    source_node_group_id,
    target_node_group_id,
    data_event_action,
    create_time,
    last_update_time,
    last_update_by
)
VALUES
(
    'Sanitop360',
    'DynamamanWMS',
    'W',
    current_timestamp,
    current_timestamp,
    'Installation'
);
```

Figura 5 - Configuração dos Links

Os links definem em alto nível como os dados se movem em todo o cenário de sincronização. O link define quais **grupos de nós** sincronizarão dados com outros grupos de nós e, dentro dessa troca, qual grupo de nós iniciará a conversa para essa troca.

Um canal pode estar configurado, em *"data_event_action"*, como "P" para *"push"* e "W" para *"pull"*.

O *"push"* indica que os nodos do grupo de nodos de origem irão iniciar a comunicação, ou seja, enviarão dados para os nodos de destino.

Enquanto que *"pull"* sinaliza que os nodos de origem irão aguardar um nodo de destino iniciar a comunicação, isto também permite que os nodos de destino extraiam dados dos nodos de origem.

3.3.2. Definindo Rotas

```
-- CONFIGURAÇÃO DO ROTEAMENTO (talvez seja necessário configurar o Target Catalog e o Target Schema)
-- Target Table, deve ser definido no caso de o nome da tabela de destino ser diferente da de origem.
INSERT INTO SYM_ROUTER
(
    router_id,
    source_node_group_id,
    target_node_group_id,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'DynamamWMS-2-Sanitop360',
    'DynamamWMS',
    'Sanitop360',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Router DynamamWMS to Sanitop360'
);
```

Figura 6 - Configuração de roteamento

3.3.3. Canais

Depois que os links de grupo e os roteadores são definidos, é o momento de especificar quais dados devem ser sincronizados nesses links e roteadores. Os canais definem os agrupamentos lógicos realizados na Tabela 4, ou seja, os registros de transações.

```
-- CONFIGURAÇÃO DOS CANAIS
INSERT INTO SYM_CHANNEL
(
    channel_id,
    processing_order,
    max_batch_size,
    max_batch_to_send,
    extract_period_millis,
    batch_algorithm,
    enabled,
    reload_flag,
    last_update_time,
    create_time,
    last_update_by,
    description
)
VALUES
(
    'chanelDynaman',
    2,
    1000,
    10,
    0,
    'default',
    1,
    1,
    current_timestamp,
    current_timestamp,
    'Installation',
    'Dynaman WMS data chanel'
);
```

Figura 7 - Configuração dos Canais

Na Figura 7 todos os dados serão transmitidos no mesmo canal. No entanto, quando se possui operações que demandam uma alta carga de dados, como blobs de grandes dimensões, recomenda-se que estes (e os associados à mesma transação) sejam transmitidos em um canal separado, para otimizar a performance de transferência de dados.

3.3.4. Definindo Triggers

Quando os triggers são definidos no SymmetricDS, eles são automaticamente gerados na base de dados correspondente.

```
-- CONFIGURAÇÃO DOS TRIGGERS (TABELAS A SEREM MONITORADAS)
INSERT INTO SYM_TRIGGER
(
    trigger_id,
    source_table_name,
    channel_id,
    last_update_time,
    create_time,
    last_update_by,
    description
)
VALUES
(
    'tablesDynaman',
    'task_detail,stock_product,picklist_detail_order,Container_Inventory,outbound_order_header,c',
    'chanelDynaman',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Dynaman WMS table mapping.'
);
```

Figura 8 - Configuração dos Triggers

A Figura 8 demonstra uma configuração de *triggers* para várias tabelas de uma só vez. Todas as alterações nestas tabelas serão capturadas, ainda que uma coluna tenha seu valor atualizado para o mesmo valor, basta que ocorra uma operação de "update" para que o *trigger* seja disparado, embora esse comportamento padrão possa ser alterado.

3.3.5. Roteamento dos Triggers

A configuração de gatilhos e roteadores é uma relação muitos-para-muitos que define quais combinações específicas de gatilhos e roteadores são necessárias para a configuração. Essa tabela serve como uma tabela de junção para definir quais combinações são válidas, bem como para definir as configurações disponíveis no nível de granularidade do roteador do gatilho.

```
-- CONFIGURAÇÃO DO ROTEAMENTO DOS TRIGGERS
-- Atenção ao initial_load_order: Posição numérica destas tabelas no carregamento inicial,
-- enviada em ordem numérica crescente.
-- Quando dois valores numéricos são iguais, a ordem é baseada em restrições de chave estrangeira.
-- Use um número negativo para excluir a tabela do carregamento inicial.
INSERT INTO SYM_TRIGGER_ROUTER
(
    trigger_id,
    router_id,
    initial_load_order,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'tablesDynaman',
    'DynamanWMS-2-Sanitop360',
    1,
    current_timestamp,
    current_timestamp,
    'Installation',
    'router for Dynaman WMS tables.'
);
```

Figura 9 – Roteamento dos triggers

3.3.6. Transformações

Em alguns casos, pode ser necessário realizar algumas transformações, como alterar o nome da tabela de origem no destino ou montar a tabela de destino baseada em mais de uma tabela de origem.

```
--TRANSFORME
-- (altera o nome da tabela do WMS para as do 360)

INSERT INTO sym_transform_table
(
    transform_id,
    source_node_group_id,
    target_node_group_id,
    transform_point,
    source_table_name,
    target_table_name,
    delete_action,
    update_action,
    column_policy,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'taskDetailChangeName',
    'DynamamWMS',
    'Sanitop360',
    'EXTRACT',
    'task_detail',
    'Dyn56_task_detail',
    'DEL_ROW',
    'UPD_ROW',
    'IMPLIED',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Changes the name of the source table on the target.'
);
```

Figura 10 - Alterando o nome da tabela

A Figura 10 demonstra a configuração para alterar o nome de uma tabela em seu destino. A coluna "*transform_point*" define se a transformação proposta acontece na origem ou no destino, a princípio esta transformação ocorrerá na origem, segundo a metodologia clássica ETL (Extract Transform Load).

```
INSERT INTO SYM_TRANSFORM_COLUMN
(
    transform_id,
    include_on,
    target_column_name,
    source_column_name,
    pk,
    transform_type,
    transform_expression,
    transform_order,
    last_update_time,
    last_update_by,
    create_time,
    description
)
VALUES
(
    'Container_InventoryChangeName',
    '*',
    'handling_qty',
    'handling_qty',
    0,
    'copy',
    '',
    1,
    current_timestamp,
    'Installation',
    current_timestamp,
    'simple copy from source column to destination column.'
);
```

Figura 11 - transformação de coluna

A Figura 11 demonstra a transformação de uma coluna onde simplesmente se copiam os dados de uma coluna de origem para a coluna de destino, sem nenhuma alteração. Este procedimento é necessário quando se copia apenas alguma das colunas de uma tabela e não todas elas implicitamente.

```
INSERT INTO SYM_TRANSFORM_COLUMN
(
    transform_id,
    include_on,
    target_column_name,
    source_column_name,
    pk,
    transform_type,
    transform_expression,
    transform_order,
    last_update_time,
    last_update_by,
    create_time,
    description
)
VALUES
(
    'Container_InventoryChangeName',
    '*',
    'item_code',
    '',
    0,
    'lookup',
    'SELECT item_code from stock_product WHERE stock_product_id = :stock_product_id',
    1,
    current_timestamp,
    'Installation',
    current_timestamp,
    'simple copy from source column to destination column.'
);
```

Figura 12 - Transformação de coluna de tabela terceira

Já a Figura 12 demonstra que a coluna de destino não será copiada de nenhuma coluna na tabela atual, mas sim de uma tabela "lookup" e o dado a ser selecionado nesta tabela é obtido através da instrução SQL definida no campo "transform_expression".

```

INSERT INTO SYM_TRANSFORM_COLUMN
(
    transform_id,
    include_on,
    target_column_name,
    source_column_name,
    pk,
    transform_type,
    transform_expression,
    transform_order,
    last_update_time,
    last_update_by,
    create_time,
    description
)
VALUES
(
    'Container_InventoryChangeName',
    '*',
    'StockType',
    '',
    0,
    'lookup',
    'SELECT CASE WHEN Location_code IN (''1BRICO'', ''OF'', ''KITOF'', ''BLISPACK'', ''BL
1,
current_timestamp,
'Installation',
current_timestamp,
'simple copy from source column to destination column.'
);

```

Figura 13 - Expressões complexas em transformação

A Figura 13 demonstra que expressões SQL diversas podem ser utilizadas, de modo a decompor um script em várias instruções menores.

4. ANEXO

4.1. SCRIPT SQL DE INSTALAÇÃO

4.1.1. Configuração dos Nodos

```
-- CONFIGURAÇÃO DOS NODOS
INSERT INTO SYM_NODE_GROUP
(
    node_group_id,
    description,
    create_time,
    last_update_time,
    last_update_by
)
VALUES
(
    'DynamamWMS',
    'IBS Dynamam WMS - SQL Server',
    current_timestamp,
    current_timestamp,
    'Installation'
);

INSERT INTO SYM_NODE_GROUP
(
    node_group_id,
    description,
    create_time,
    last_update_time,
    last_update_by
)
VALUES
(
    'Sanitop360',
    'Sanitop 360 - SQL Server',
    current_timestamp,
    current_timestamp,
    'Installation'
);
```

4.1.2. Configuração dos links

```
INSERT INTO SYM_NODE_GROUP_LINK
(
    source_node_group_id,
    target_node_group_id,
    data_event_action,
    create_time,
    last_update_time,
    last_update_by
)
VALUES
(
    'DynamamWMS',
    'Sanitop360',
    'P',
    current_timestamp,
    current_timestamp,
    'Installation'
);

INSERT INTO SYM_NODE_GROUP_LINK
(
    source_node_group_id,
    target_node_group_id,
    data_event_action,
    create_time,
    last_update_time,
    last_update_by
)
VALUES
(
    'Sanitop360',
    'DynamamWMS',
    'W',
    current_timestamp,
    current_timestamp,
    'Installation'
);
```

4.1.3. Configuração do roteamento

```
INSERT INTO SYM_ROUTER
(
    router_id,
    source_node_group_id,
    target_node_group_id,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'DynamanWMS-2-Sanitop360',
    'DynamanWMS',
    'Sanitop360',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Router DynamanWMS to Sanitop360'
);
```

4.1.4. Configuração dos canais

```
INSERT INTO SYM_CHANNEL
(
    channel_id,
    processing_order,
    max_batch_size,
    max_batch_to_send,
    extract_period_millis,
    batch_algorithm,
    enabled,
    reload_flag,
    last_update_time,
    create_time,
    last_update_by,
    description
)
VALUES
(
    'chanelDynaman',
    2,
    1000,
    10,
    0,
    'default',
    1,
    1,
    current_timestamp,
    current_timestamp,
    'Installation',
    'Dynaman WMS data chanel'
);
```

4.1.5. Configuração dos triggers

```
INSERT INTO SYM_TRIGGER
(
    trigger_id,
    source_table_name,
    channel_id,
    last_update_time,
    create_time,
    last_update_by,
    description
)
VALUES
(
    'tablesDynamaman',
    'task_detail,stock_product,picklist_detail_order,Container_Inventor
y,outbound_order_header,outbound_Order_Detail,inbound_Order_Header,arri
val_detail,arrival_header,inbound_Order_detail,outbound_Order_text',
    'chanelDynamaman',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Dynamaman WMS table mapping.'
);
```

4.1.6. Configuração do roteamento dos triggers

```
INSERT INTO SYM_TRIGGER_ROUTER
(
    trigger_id,
    router_id,
    initial_load_order,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'tablesDynaman',
    'DynamanWMS-2-Sanitop360',
    1,
    current_timestamp,
    current_timestamp,
    'Installation',
    'router for Dynaman WMS tables.'
);
```

4.1.7. Transformação de tabelas

```
INSERT INTO sym_transform_table
(
    transform_id,
    source_node_group_id,
    target_node_group_id,
    transform_point,
    source_table_name,
    target_table_name,
    delete_action,
    update_action,
    column_policy,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'taskDetailChangeName',
    'DynamamanWMS',
    'Sanitop360',
    'EXTRACT',
    'task_detail',
    'Dyn56_task_detail',
    'DEL_ROW',
    'UPD_ROW',
    'IMPLIED',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Changes the name of the source table on the target.'
);
```

```

INSERT INTO sym_transform_table
(
    transform_id,
    source_node_group_id,
    target_node_group_id,
    transform_point,
    source_table_name,
    target_table_name,
    delete_action,
    update_action,
    column_policy,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'stockProductChangeName',
    'DynamamanWMS',
    'Sanitop360',
    'EXTRACT',
    'stock_product',
    'Dyn56_stock_product',
    'DEL_ROW',
    'UPD_ROW',
    'IMPLIED',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Changes the name of the source table on the target.'
);

```



```

INSERT INTO sym_transform_table
(
    transform_id,
    source_node_group_id,
    target_node_group_id,
    transform_point,
    source_table_name,
    target_table_name,
    delete_action,
    update_action,
    column_policy,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'picklistDetailOrderChangeName',
    'DynamamanWMS',
    'Sanitop360',
    'EXTRACT',
    'picklist_detail_order',
    'Dyn56_picklist_detail_order',
    'DEL_ROW',
    'UPD_ROW',
    'IMPLIED',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Changes the name of the source table on the target.'
);

```

```

INSERT INTO sym_transform_table
(
    transform_id,
    source_node_group_id,
    target_node_group_id,
    transform_point,
    source_table_name,
    target_table_name,
    delete_action,
    update_action,
    column_policy,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'outboundOrderDetailChangeName',
    'DynamamanWMS',
    'Sanitop360',
    'EXTRACT',
    'outbound_Order_detail',
    'Dyn56_Outbound_order_Detail',
    'DEL_ROW',
    'UPD_ROW',
    'IMPLIED',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Changes the name of the source table on the target.'
);

```

```

INSERT INTO sym_transform_table
(
    transform_id,
    source_node_group_id,
    target_node_group_id,
    transform_point,
    source_table_name,
    target_table_name,
    delete_action,
    update_action,
    column_policy,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'outboundOrderHeaderChangeName',
    'DynamamanWMS',
    'Sanitop360',
    'EXTRACT',
    'outbound_order_header',
    'Dyn56_Outbound_order_header',
    'DEL_ROW',
    'UPD_ROW',
    'IMPLIED',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Changes the name of the source table on the target.'
);

```

```

INSERT INTO sym_transform_table
(
    transform_id,
    source_node_group_id,
    target_node_group_id,
    transform_point,
    source_table_name,
    target_table_name,
    delete_action,
    update_action,
    column_policy,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'inboundOrderHeaderChangeName',
    'DynamamanWMS',
    'Sanitop360',
    'EXTRACT',
    'inbound_Order_Header',
    'Dyn56_Inbound_order_Header',
    'DEL_ROW',
    'UPD_ROW',
    'IMPLIED',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Changes the name of the source table on the target.'
);

```

```

INSERT INTO sym_transform_table
(
    transform_id,
    source_node_group_id,
    target_node_group_id,
    transform_point,
    source_table_name,
    target_table_name,
    delete_action,
    update_action,
    column_policy,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'arrivalDetailChangeName',
    'DynamamanWMS',
    'Sanitop360',
    'EXTRACT',
    'arrival_detail',
    'Dyn56_Arrival_Detail',
    'DEL_ROW',
    'UPD_ROW',
    'IMPLIED',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Changes the name of the source table on the target.'
);

```

```

INSERT INTO sym_transform_table
(
    transform_id,
    source_node_group_id,
    target_node_group_id,
    transform_point,
    source_table_name,
    target_table_name,
    delete_action,
    update_action,
    column_policy,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'arrivalHeaderChangeName',
    'DynamamanWMS',
    'Sanitop360',
    'EXTRACT',
    'arrival_header',
    'Dyn56_Arrival_Header',
    'DEL_ROW',
    'UPD_ROW',
    'IMPLIED',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Changes the name of the source table on the target.'
);

```

```

INSERT INTO sym_transform_table
(
    transform_id,
    source_node_group_id,
    target_node_group_id,
    transform_point,
    source_table_name,
    target_table_name,
    delete_action,
    update_action,
    column_policy,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'inboundOrderDetailChangeName',
    'DynamamanWMS',
    'Sanitop360',
    'EXTRACT',
    'inbound_Order_detail',
    'Dyn56_Inbound_Order_Detail',
    'DEL_ROW',
    'UPD_ROW',
    'IMPLIED',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Changes the name of the source table on the target.'
);

```

```

INSERT INTO sym_transform_table
(
    transform_id,
    source_node_group_id,
    target_node_group_id,
    transform_point,
    source_table_name,
    target_table_name,
    delete_action,
    update_action,
    column_policy,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'outboundOrderTextChangeName',
    'DynamamWMS',
    'Sanitop360',
    'EXTRACT',
    'outbound_Order_text',
    'Dyn56_Outbound_Order_Text',
    'DEL_ROW',
    'UPD_ROW',
    'IMPLIED',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Changes the name of the source table on the target.'
);

```



```

INSERT INTO sym_transform_table
(
    transform_id,
    source_node_group_id,
    target_node_group_id,
    transform_point,
    source_table_name,
    target_table_name,
    delete_action,
    update_action,
    column_policy,
    create_time,
    last_update_time,
    last_update_by,
    description
)
VALUES
(
    'Container_InventoryChangeName',
    'DynamamanWMS',
    'Sanitop360',
    'EXTRACT',
    'Container_Inventory',
    'Dyn56_Container_Inventory',
    'DEL_ROW',
    'UPD_ROW',
    'SPECIFIED',
    current_timestamp,
    current_timestamp,
    'Installation',
    'Changes the name of the source table on the target.'
);

```

4.1.8. Transformação de colunas

```
INSERT INTO SYM_TRANSFORM_COLUMN
(
    transform_id,
    include_on,
    target_column_name,
    source_column_name,
    pk,
    transform_type,
    transform_expression,
    transform_order,
    last_update_time,
    last_update_by,
    create_time,
    description
)
VALUES
(
    'Container_InventoryChangeName',
    '*',
    'location_code',
    'location_code',
    0,
    'copy',
    '',
    1,
    current_timestamp,
    'Installation',
    current_timestamp,
    'simple copy from source column to destination column.'
);
```

```
INSERT INTO SYM_TRANSFORM_COLUMN
(
    transform_id,
    include_on,
    target_column_name,
    source_column_name,
    pk,
    transform_type,
    transform_expression,
    transform_order,
    last_update_time,
    last_update_by,
    create_time,
    description
)
VALUES
(
    'Container_InventoryChangeName',
    '*',
    'Container_code',
    'container_code',
    1,
    'copy',
    '',
    1,
    current_timestamp,
    'Installation',
    current_timestamp,
    'simple copy from source column to destination column.'
);
```

```
INSERT INTO SYM_TRANSFORM_COLUMN
(
    transform_id,
    include_on,
    target_column_name,
    source_column_name,
    pk,
    transform_type,
    transform_expression,
    transform_order,
    last_update_time,
    last_update_by,
    create_time,
    description
)
VALUES
(
    'Container_InventoryChangeName',
    '*',
    'inventory_qty',
    'inventory_qty',
    0,
    'copy',
    '',
    1,
    current_timestamp,
    'Installation',
    current_timestamp,
    'simple copy from source column to destination column.'
);
```

```
INSERT INTO SYM_TRANSFORM_COLUMN
(
    transform_id,
    include_on,
    target_column_name,
    source_column_name,
    pk,
    transform_type,
    transform_expression,
    transform_order,
    last_update_time,
    last_update_by,
    create_time,
    description
)
VALUES
(
    'Container_InventoryChangeName',
    '*',
    'qty_on_picklist',
    'qty_on_picklist',
    0,
    'copy',
    '',
    1,
    current_timestamp,
    'Installation',
    current_timestamp,
    'simple copy from source column to destination column.'
);
```

```
INSERT INTO SYM_TRANSFORM_COLUMN
(
    transform_id,
    include_on,
    target_column_name,
    source_column_name,
    pk,
    transform_type,
    transform_expression,
    transform_order,
    last_update_time,
    last_update_by,
    create_time,
    description
)
VALUES
(
    'Container_InventoryChangeName',
    '*',
    'qty_not_on_picklist',
    'qty_not_on_picklist',
    0,
    'copy',
    '',
    1,
    current_timestamp,
    'Installation',
    current_timestamp,
    'simple copy from source column to destination column.'
);
```

```
INSERT INTO SYM_TRANSFORM_COLUMN
(
    transform_id,
    include_on,
    target_column_name,
    source_column_name,
    pk,
    transform_type,
    transform_expression,
    transform_order,
    last_update_time,
    last_update_by,
    create_time,
    description
)
VALUES
(
    'Container_InventoryChangeName',
    '*',
    'picked_qty',
    'picked_qty',
    0,
    'copy',
    '',
    1,
    current_timestamp,
    'Installation',
    current_timestamp,
    'simple copy from source column to destination column.'
);
```

```
INSERT INTO SYM_TRANSFORM_COLUMN
(
    transform_id,
    include_on,
    target_column_name,
    source_column_name,
    pk,
    transform_type,
    transform_expression,
    transform_order,
    last_update_time,
    last_update_by,
    create_time,
    description
)
VALUES
(
    'Container_InventoryChangeName',
    '*',
    'handling_qty',
    'handling_qty',
    0,
    'copy',
    '',
    1,
    current_timestamp,
    'Installation',
    current_timestamp,
    'simple copy from source column to destination column.'
);
```



```

INSERT INTO SYM_TRANSFORM_COLUMN
(
    transform_id,
    include_on,
    target_column_name,
    source_column_name,
    pk,
    transform_type,
    transform_expression,
    transform_order,
    last_update_time,
    last_update_by,
    create_time,
    description
)
VALUES
(
    'Container_InventoryChangeName',
    '*',
    'item_code',
    '',
    0,
    'lookup',
    'SELECT item_code from stock_product WHERE stock_product_id =
:stock_product_id',
    1,
    current_timestamp,
    'Installation',
    current_timestamp,
    'simple copy from source column to destination column.'
);

```

```

INSERT INTO SYM_TRANSFORM_COLUMN
(
    transform_id,
    include_on,
    target_column_name,
    source_column_name,
    pk,
    transform_type,
    transform_expression,
    transform_order,
    last_update_time,
    last_update_by,
    create_time,
    description
)
VALUES
(
    'Container_InventoryChangeName',
    '*',
    'equipment_zone_code',
    '',
    0,
    'lookup',
    'SELECT equipment_zone_code from location WHERE warehouse_code
= :warehouse_code AND location_code = :location_code',
    1,
    current_timestamp,
    'Installation',
    current_timestamp,
    'simple copy from source column to destination column.'
);

```

```

INSERT INTO SYM_TRANSFORM_COLUMN
(
    transform_id,
    include_on,
    target_column_name,
    source_column_name,
    pk,
    transform_type,
    transform_expression,
    transform_order,
    last_update_time,
    last_update_by,
    create_time,
    description
)
VALUES
(
    'Container_InventoryChangeName',
    '*',
    'StockType',
    '',
    0,
    'lookup',
    'SELECT CASE WHEN Location_code IN (''1BRICO'', ''OF'',
''KITOF'', ''BLISPACK'', ''BLISPACK0'', ''PINT'', ''PVNOVA'',
''PVNOVA0'') THEN ''OF'' WHEN Location_code IN (''LOST'', ''UNDO'')
THEN ''LOST'' WHEN Location_code LIKE (''GER%'') THEN ''LOST'' WHEN
Location_code LIKE (''SGUR%'') THEN ''LOST'' WHEN Location_code LIKE
(''AGUARD%'') THEN ''WAIT'' WHEN Location_code LIKE (''4CAIS%'') THEN
''WAIT'' WHEN Location_code IN (''01W0'', ''01W1'', ''02W0'', ''02W1'',
''02W2'', ''04W0'', ''04W1'', ''6P01W0'', ''06P02W0'') THEN ''CAIS''
WHEN Location_code IN (''04DOC01'', ''04DOC02'', ''04DOC03'',
''04DOC04'', ''04DOC05'', ''04DOCS0S'', ''01DOC01'', ''01DOC02'',
''01DOCNEI'', ''01DOCSAV'') THEN ''CAIS'' WHEN Location_code =
''1REC01'' THEN ''REC'' WHEN Location_code LIKE (''1CAI%'') THEN
''CAIS'' ELSE ''STOCK'' END AS StockType from location WHERE
warehouse_code = :warehouse_code AND location_code = :location_code',
    1,
    current_timestamp,
    'Installation',
    current_timestamp,
    'simple copy from source column to destination column.'
);

```

Cofinanciado por:



UNIÃO EUROPEIA
Fundo Europeu
de Desenvolvimento Regional