1. Planeamento da arquitetura

Dentre os pontos que devem ser identificados, destaca-se: 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 em execução.

Diagrama

Descrição gerada automaticamente

Figura 1 - Topologia e fluxo de informações.

A Figura 1 deixa claro em que rede cada base de dados está, bem como o fluxo de dados, isto é, o sentido no qual a informação deve fluir. Observe que entre os servidores SRVA e SRVB ocorre um canal de dados bidirecional, enquanto as duas instâncias de base de dados que correm no mesmo servidor SRVC (situação incomum), enviam dados para o SRVB por um canal unidirecional.

A identificação das redes se demonstra de extrema utilidade na fase de implementação sobretudo para definição das políticas de segurança de rede, como as portas que devem ser liberadas nos Firewalls.

Na sequência deve ser elaborado um mapa de correlação das tabelas a sincronizar, ou seja, identificar as tabelas da origem que devem ser sincronizadas com as tabelas do destino. Isto torna-se necessário pelo fato de uma tabela de origem poder ter um nome distinto em seu destino, bem como contar com 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 nesta fase preliminar é a versão do SGBD em execução na base de dados. Com esta informação se pode definir corretamente os drives JDBC necessários para a implementação.

Cumpre salientar que o SymmetricDS vem acompanhado com alguns drives JDBC, entretanto pode ocorrer alguma incompatibilidade entre as versões em execução e as disponibilizadas pela aplicação, como falhas de autenticação ou reconexões excessivas. Justamente por esta possibilidade que se faz necessário ter documentado a versão do SGBD para selecionar previamente 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.

Por fim deve se definir um agrupamento transacional, que tem por 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.

Esta informação é importante para agrupar estas tabelas em um 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). 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.

1. Implementação
   1. 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:

engine.name

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

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

external.id

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

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).

sync.url

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

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.

db.driver

O nome da classe Java do driver JDBC.

Exemplo: com.microsoft.sqlserver.jdbc.SQLServerDriver

db.url

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

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.

db.password

A senha para o usuário do banco de dados.

Tela preta com letras brancas

Descrição gerada automaticamente

Figura 2 - Exemplo de configuração de nodo

A Figura 2 apresenta um ficheiro de configuração de um nodo SymmetricDS, com a configuração mínima necessária para sua execução.

* 1. Configuração da sincronização

O script SQL deve ser implementado na base de dados do nodo mestre (servidor de registro).

Desta forma, todos os nodos do sistema irão receber as configurações que lhe dizem respeito.

O primeiro a se configurar são os nodos propriamente ditos, que até então só existem em separados e não no sistema.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Figura 3 - Configuração dos nodos

A Figura 3 exibe um código SQL para configuração dos nodos.

Observe que o 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.

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.

Texto

Descrição gerada automaticamente

Figura 4 - 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.

Definindo Rotas

Texto

Descrição gerada automaticamente

Figura 5 - Configuração de roteamento

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.

Texto

Descrição gerada automaticamente

Figura 6 - Configuração dos Canais

Na Figura 6 todos os dados serão transmitidos no mesmo canal. Quando se possui operações que demandam de alta carga de dados, como blobs de grandes dimensões, recomenda-se que estes (e os associados a mesma transação) sejam transmitidos em um canal separado, para otimizar a performance de transferência de dados.

Definindo Triggers

Ao definir os triggers no SymmetricDS eles serão automaticamente gerados na base de dado correspondente.

Texto

Descrição gerada automaticamente

Figura 7 - Configuração dos Triggers

A Figura 7 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.

Roteamento dos Triggers

Esta configuração serve para definir quais combinações específicas de gatilhos e roteadores são necessárias para sua configuração. A relação entre gatilhos e roteadores é muitos-para-muitos, portanto, essa tabela serve como a tabela de junção para definir quais combinações são válidas, bem como para definir as configurações disponíveis no nível de granularidade do roteador do gatilho.

Texto

Descrição gerada automaticamente

Figura 8 – Roteamento dos triggers

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.

Texto

Descrição gerada automaticamente

Figura 9 - Alterando o nome da tabela

A Figura 9 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).

Texto

Descrição gerada automaticamente

Figura 10 - transformação de coluna

A Figura 10 demonstra a transformação de uma coluna, onde simplesmente está se copiando os dados de uma coluna de origem para a coluna de destino, sem nenhuma alteração. Estre procedimento se faz necessário quando se está copiando apenas alguma das colunas de uma tabela e não todas implicitamente.

Texto

Descrição gerada automaticamente

Figura 11 - Transformação de coluna de tabela terceira

Já a Figura 11 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*”.

Texto

Descrição gerada automaticamente

Figura 12 - Expressões complexas em transformação

A Figura 12 demonstra que expressões SQL diversas podem ser utilizadas, de modo a decompor um script em várias instruções menores.