Manual de instalação de conector SQL Server 2008 para o intermediário Kafka

Ivo M. F. Silva and Carlos C. Teixeira

DI–FCUL–TR–2017–03

DOI:10.451/31160

(http://hdl.handle.net/10451/31160)

November 2017

Publication without review in the repository of the Department of Informatics
of the University of Lisbon, Faculty of Sciences
(http://repositorio.ul.pt/handle/10451/12254).

Índice

1. Introdução

Este documento pretende auxiliar a instalação do conector SQL Server que permite obter dados do SGBD, convertê-los para JSON e publicá-los num intermediário Kafka usando o consumidor JSON Kafka-REST-Proxy [3].

O conector foi desenvolvido como parte de um sistema de integração de dados de uma multinacional de telecomunicações no âmbito de um projeto de mestrado do Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa.

2. Requisitos para instalar o conector

1. Permissões de *sysadmin* num servidor SQL Server 2008.

2. Os scripts *usp_InstalationScriptAdminCDC.sql* e *usp_InstalationScriptDBA.sql* .

3. A biblioteca SqlWebRequest.dll compilada para o .NET Framework 2.0 e localizada na diretoria C: (raiz do sistema de ficheiros Windows, para alterar diretoria ver capítulo "Configurações").

4. Acesso a um consumidor JSON para publicar as alterações detetadas nas tabelas.

3. Como instalar o conector

3.1. Instalação normal

1. Aceder ao servidor SQL Server 2008 com um login que tenha permissões de *sysadmin*.

2. Mudar o contexto para a BD desejada (comando *USE*) ou criá-la caso ainda não exista.

3. a) Caso os utilizadores da BD sejam utilizadores pré-existentes ou utilizadores do Windows é necessário alterar o seu nome nos scripts de configuração como descrito no capítulo "Configurações" deste documento.

3. b) Caso se pretendam criar novos utilizadores, o conector assume que já existem os logins e users do SQL Server *UserCDC* e *UserCDCAdmin*. Se estes não existirem é necessário criá-los antes de continuar. Para isto sugerem-se os passos indicados no capítulo seguinte deste documento "Como criar os logins e users necessários".

4. Caso os roles *CDC_admin* e *CDC_user* já existam, seguir o caso especial nº1 presente no final desta secção.

5. Caso o schema *propcdc* já exista, seguir o caso especial nº2 presente no final desta secção.

6. Abrir o script *usp_InstalationScriptDBA.sql* e executar para criar o procedimento *usp_InstalationScriptDBA.*

7. Executar o procedimento recém-criado: (EXEC usp_InstalationScriptDBA ;).

8. Terminar a atribuição de permissões aos logins e users descrita no capítulo deste documento "Como dar as permissões necessárias aos logins e users".

9. Aceder ao servidor SQL Server com o login do *UserCDCAdmin*.

10. Abrir o script *usp_InstalationScriptAdminCDC.sql* e executar para criar o procedimento *usp_InstalationScriptAdminCDC*.

11. Executar o procedimento recém-criado: (EXEC usp_InstalationScriptAdminCDC ;).

12. Caso os comandos sugeridos em seguida não funcionem, seguir o caso especial nº3 presente no final desta secção.

13. Registar as tabelas que o conector deve monitorizar, uma a uma com o procedimento *usp_RegisterTableInCDC* seguido do nome da tabela.

(EXEC usp_RegisterTableInCDC @tableName = N'tabelaDeExemplo' ;).

14. Caso seja necessário cancelar o registo de alguma tabela, seguir o caso especial nº4 presente no final desta secção.

15. Iniciar a captura de alterações nas tabelas registadas executando o procedimento *usp_StartGetCDCAlterations*.

(EXEC usp_StartGetCDCAlterations ;).

16. Para terminar a conversão e envio de alterações para o consumidor JSON, executar o procedimento *usp_StopGetCDCAlterations*.

(EXEC usp_StopGetCDCAlterations ;).

17. Quando se alterar alguma tabela (ALTER TABLE), seguir o caso especial nº5 presente no final desta secção.

18. Caso se altere a maioria das tabelas, pode ser mais conveniente seguir o caso nº6 presente no final desta secção.

3.2. Casos especiais da instalação

1. Caso os roles *CDC_admin* e *CDC_user* já existam é necessário remover todos os seus membros antes de removê-los:

(EXEC sp_droprolemember 'CDC_admin', 'UserCDCAdmin';)

(EXEC sp_droprolemember 'CDC_user', 'UserCDC';)

e em seguida já é possível removê-los:

(DROP ROLE CDC_admin;)

(DROP ROLE CDC_user;).

2. Caso o schema *propcdc* já exista, também é necessário removê-lo:

(DROP SCHEMA propcdc;).

3. Caso os comandos sugeridos depois do ponto 12 da instalação normal não funcionem, deve ser necessário executar os procedimentos chamando pelo schema.nome_do_procedimento, dependendo da forma como foram criados os utilizadores.

4. Caso seja necessário cancelar o registo de alguma tabela, executar o procedimento *usp_UnregisterTableInCDC* seguido do nome da tabela.

(EXEC usp_UnregisterTableInCDC @tableName = N'tabelaDeExemplo' ;).

5. Quando se alterar alguma tabela (ALTER TABLE), é necessário executar um procedimento de sincronização seguido do nome da tabela alterada:

(EXEC usp_SyncCDCTable @TableName = N'tabelaDeExemplo' ;).

6. Se a maioria das tabelas forem alteradas, em vez de indicar o nome de cada uma delas pode ser mais conveniente executar um procedimento que sincronize todas as tabelas alteradas:

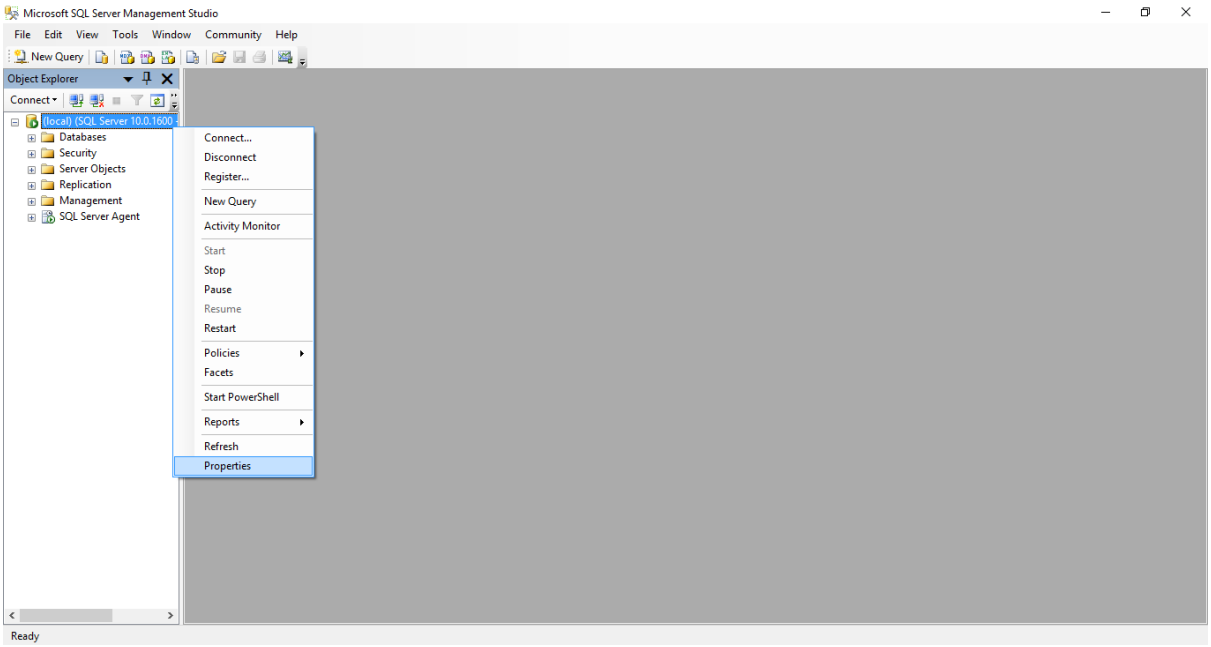
(EXEC usp_SyncAllCDCTables ;).

4. Como criar os logins e users necessários

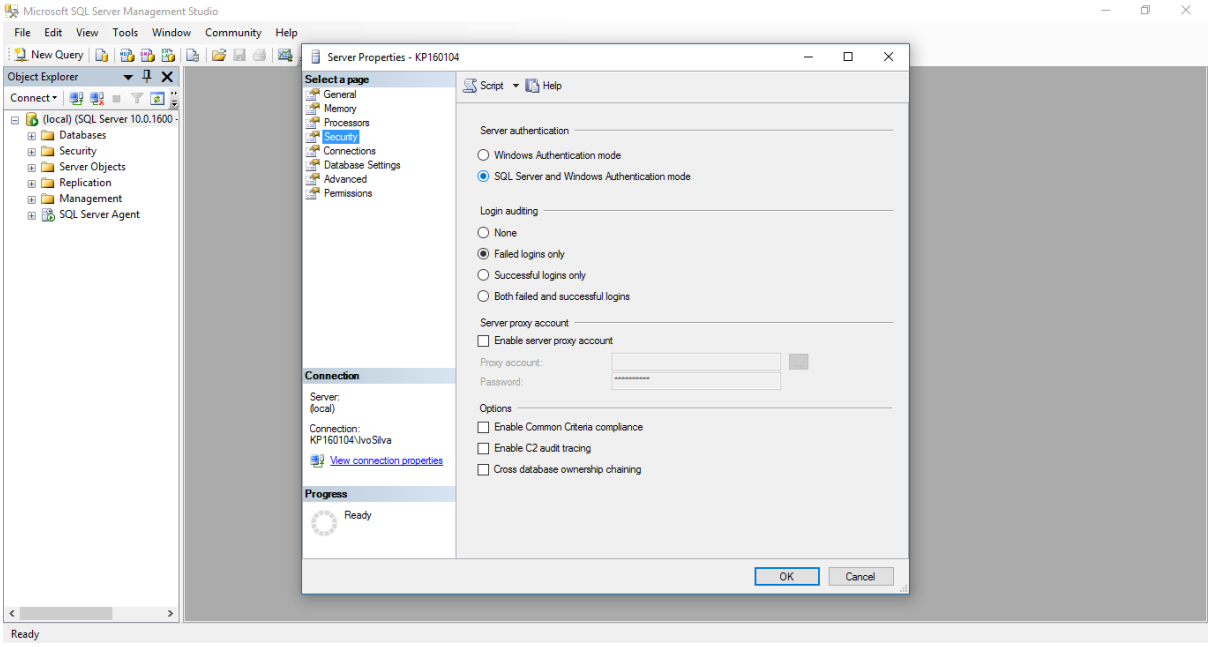
Caso o utilizador *sa* esteja desativado, é necessário ativá-lo com uma password à escolha:

(ALTER LOGIN sa ENABLE; ALTER LOGIN sa WITH PASSWORD='ABCD').

Caso o modo de autenticação esteja restrito nas configurações da BD é necessário alterá-lo para permitir autenticação SQL Server, em Server->Properties:

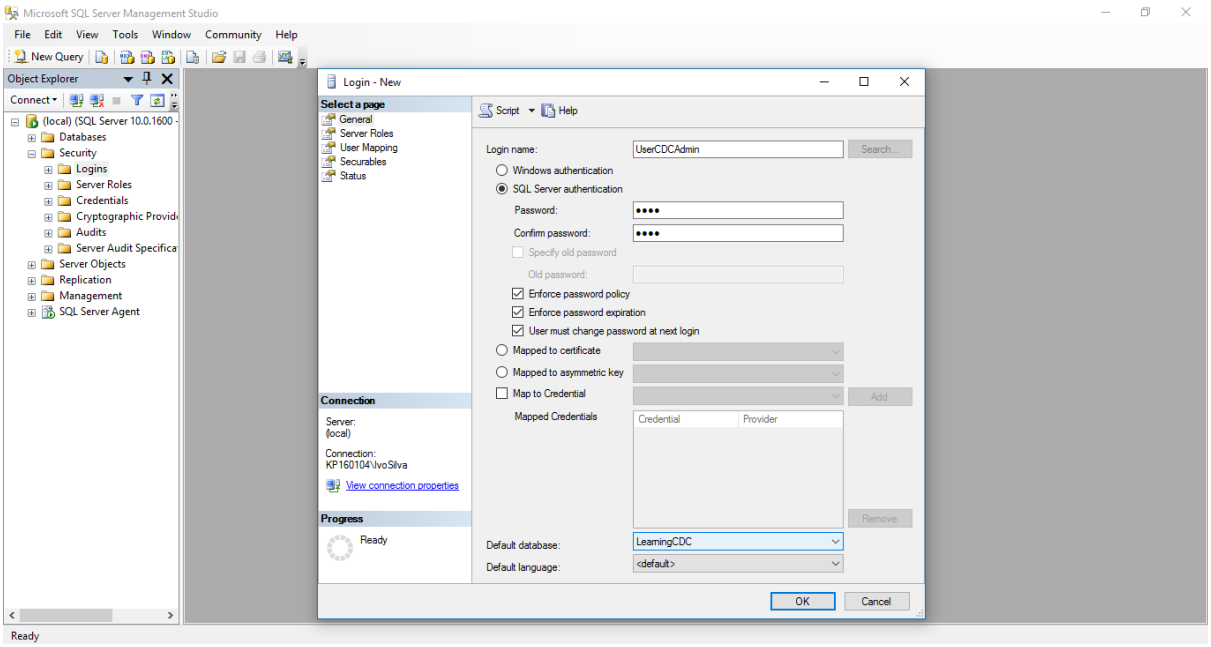


No separador Security selecionar “SQL Server and Windows Authentication mode”:

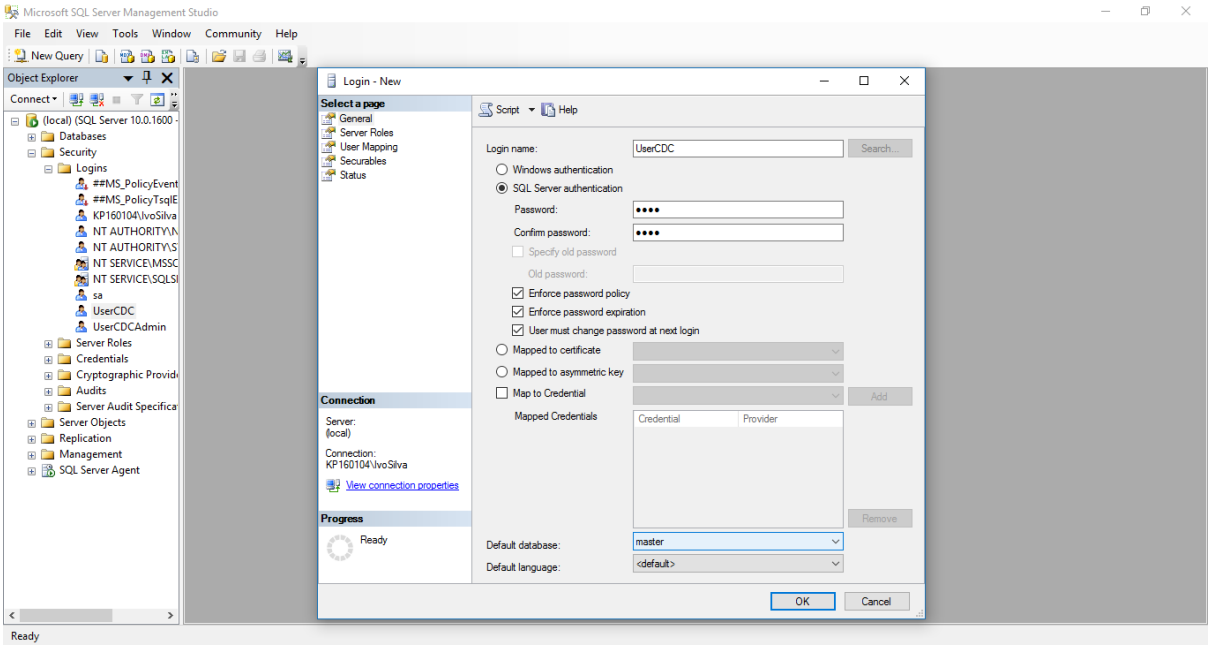


Para criar os logins é necessário selecionar Server->Security->Logins->New Login:

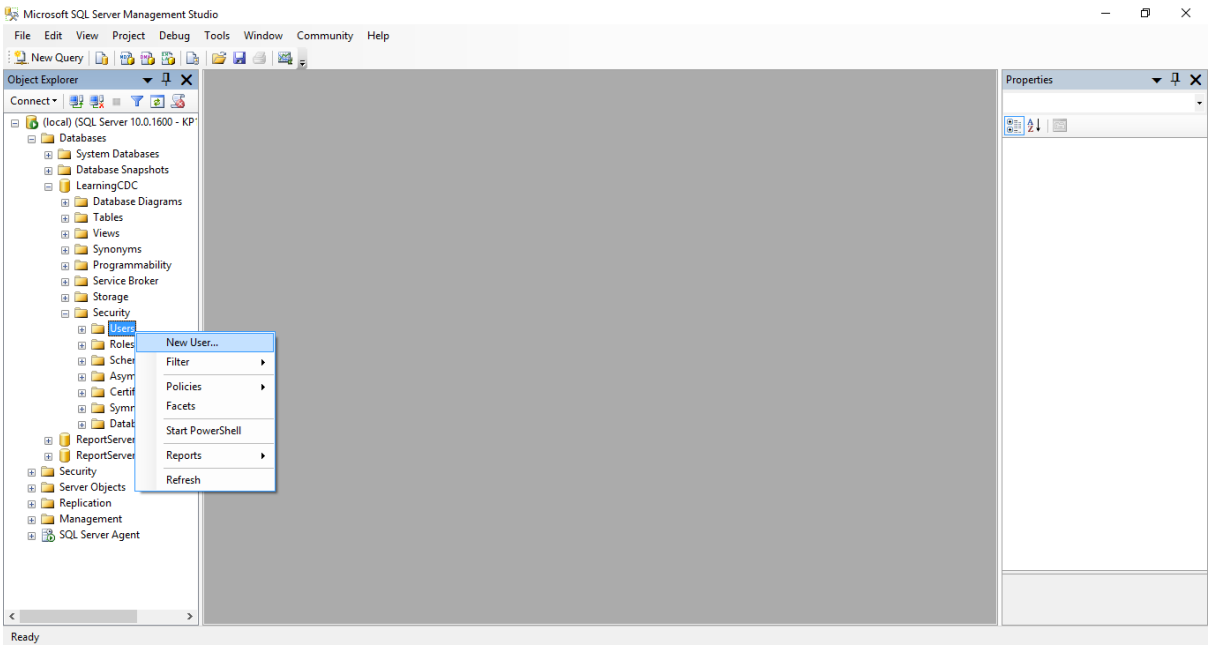
Para criar o login *UserCDCAdmin*:



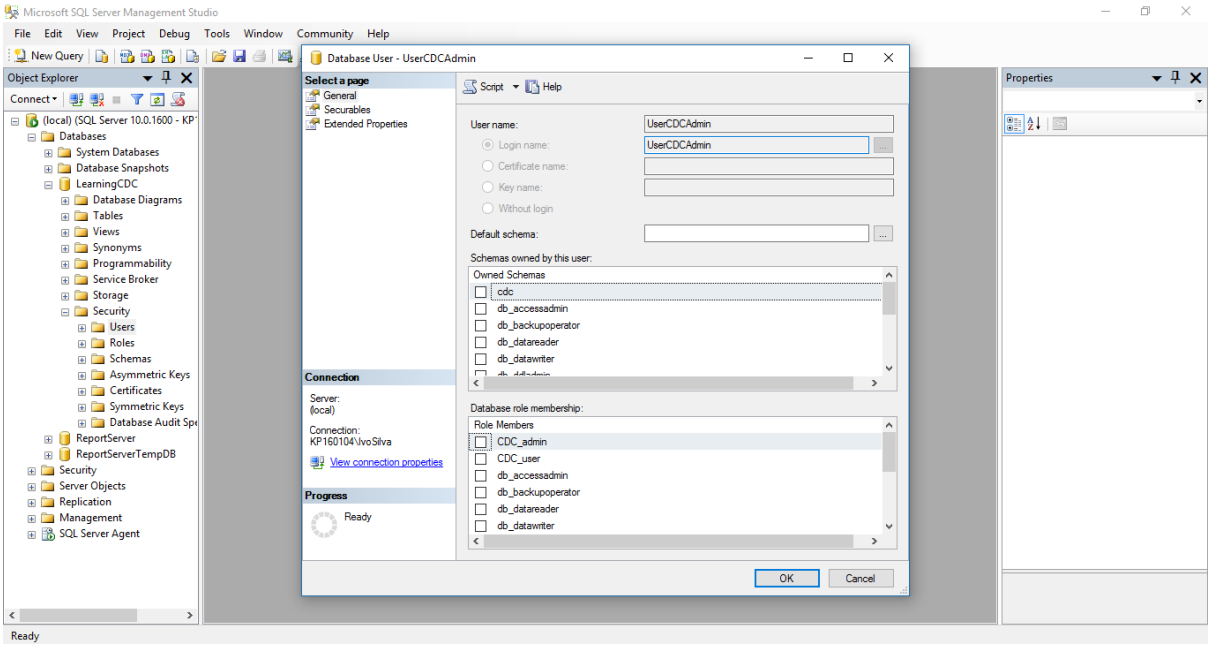
Para criar o login *UserCDC*:



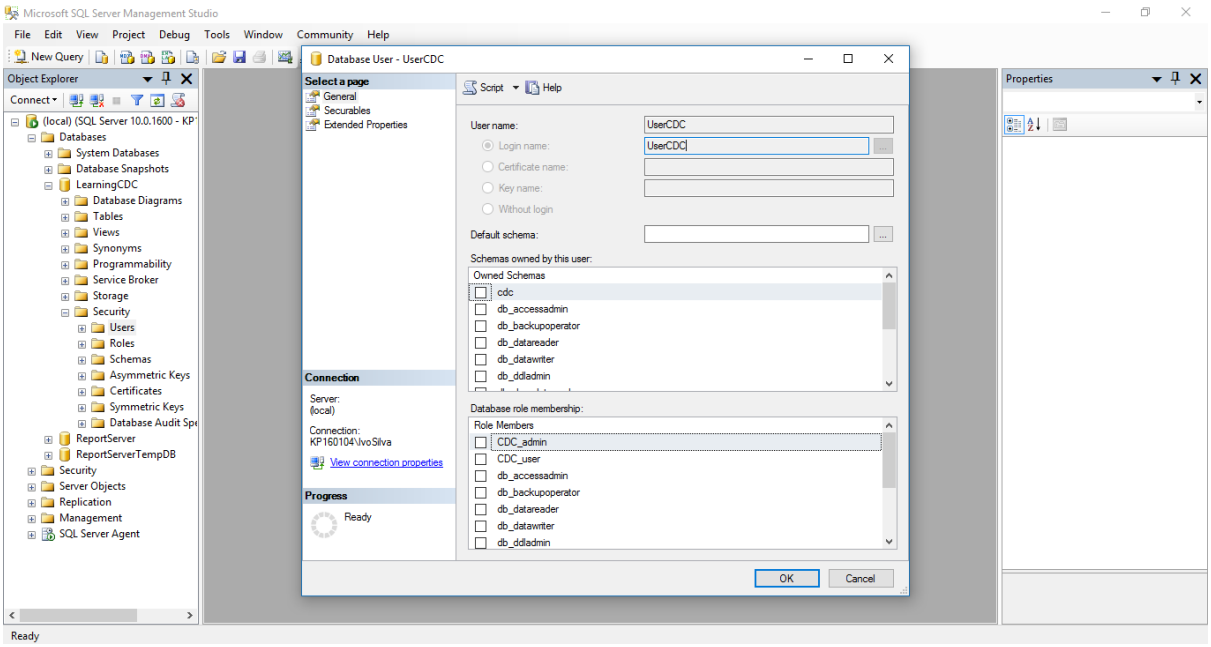
Para criar os users selecionar Server->Database->NomedaBD->Security->Users->New User:



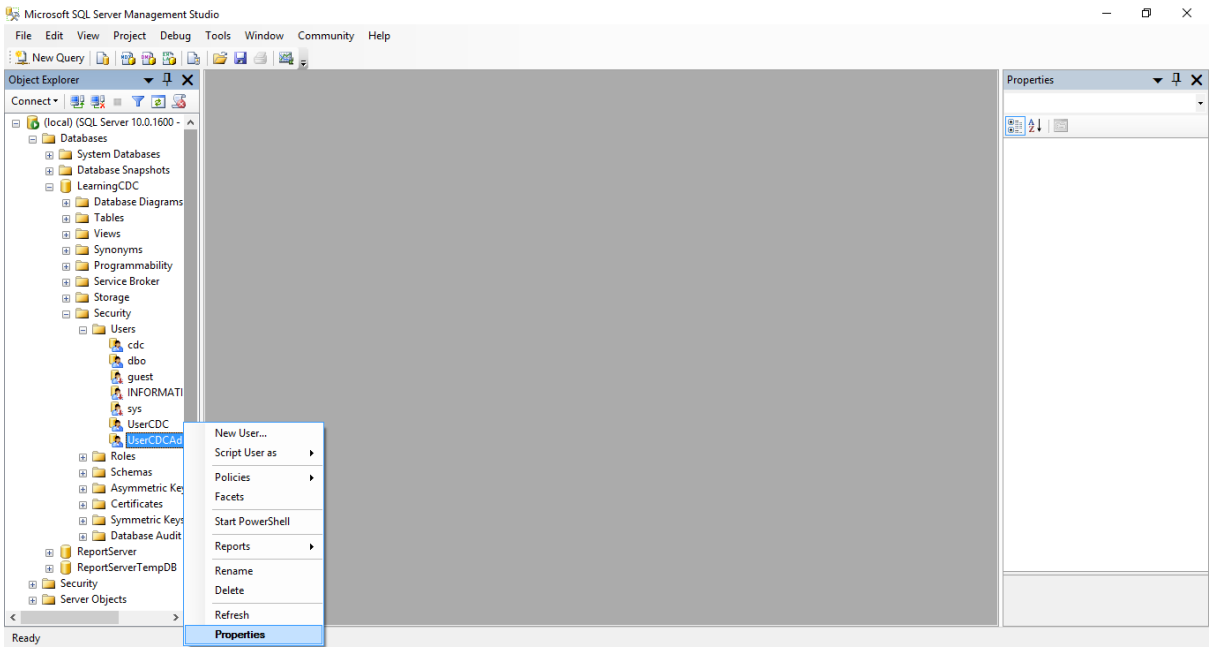
Para criar o user *UserCDCAdmin*:



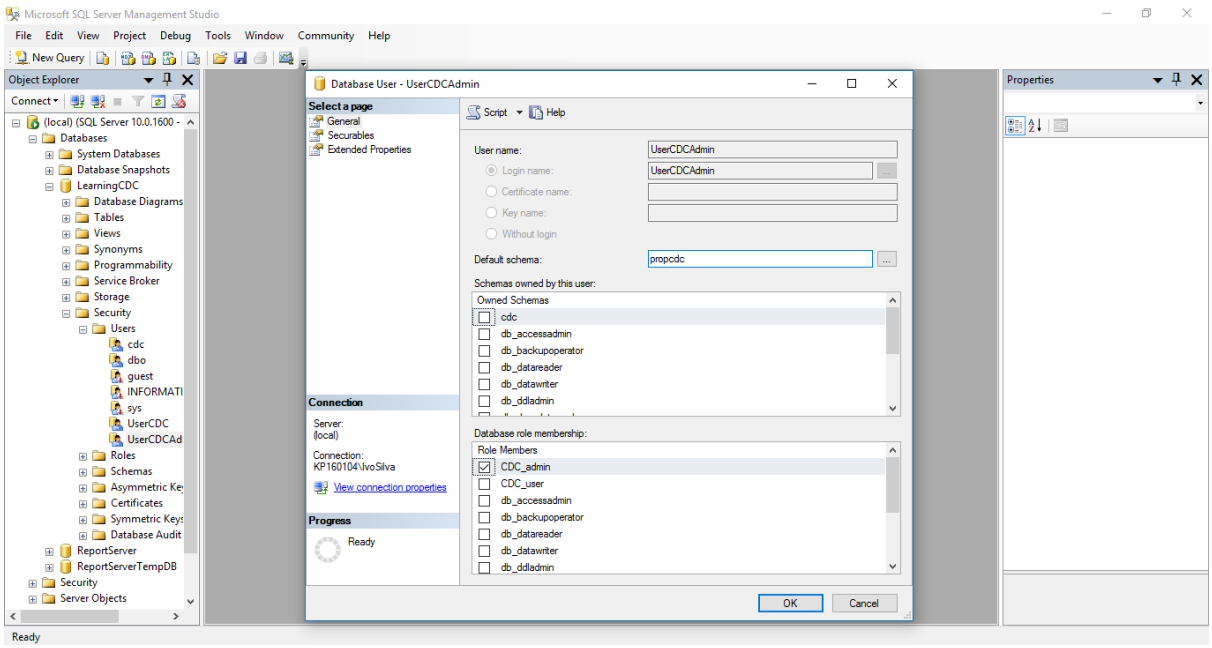
Para criar o user *UserCDC*:



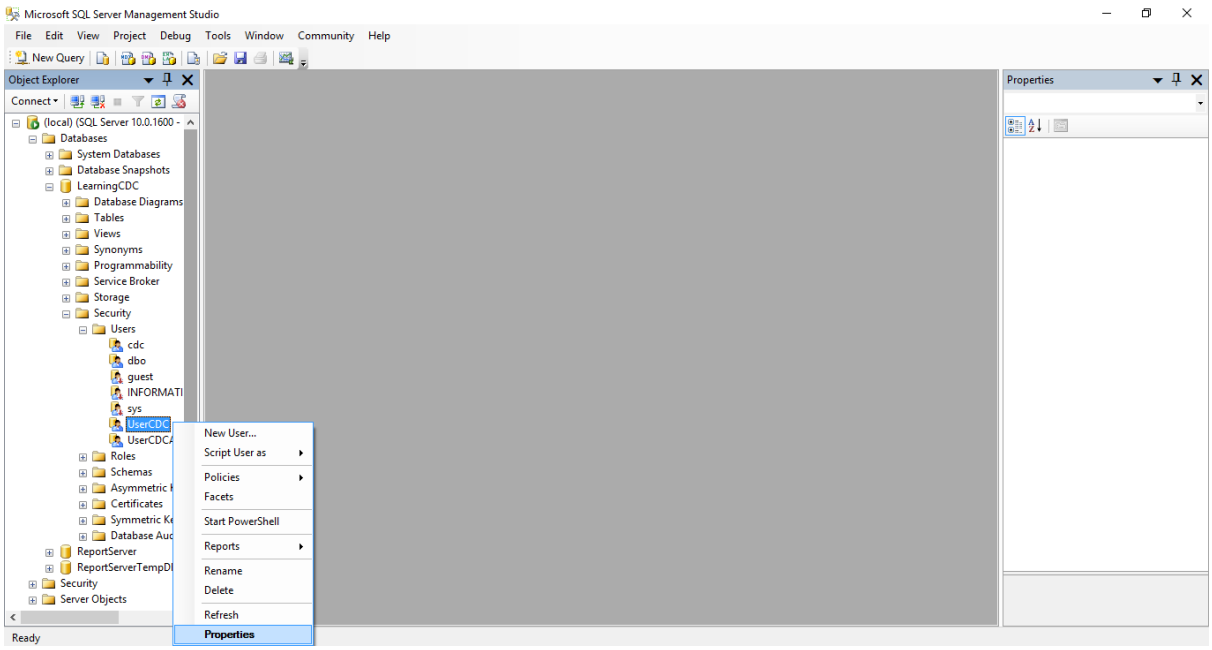
Abrir as propriedades do user *UserCDCAdmin*, em Server->Databases->NomedaBD->Security->Users->UserCDCAdmin->Properties:



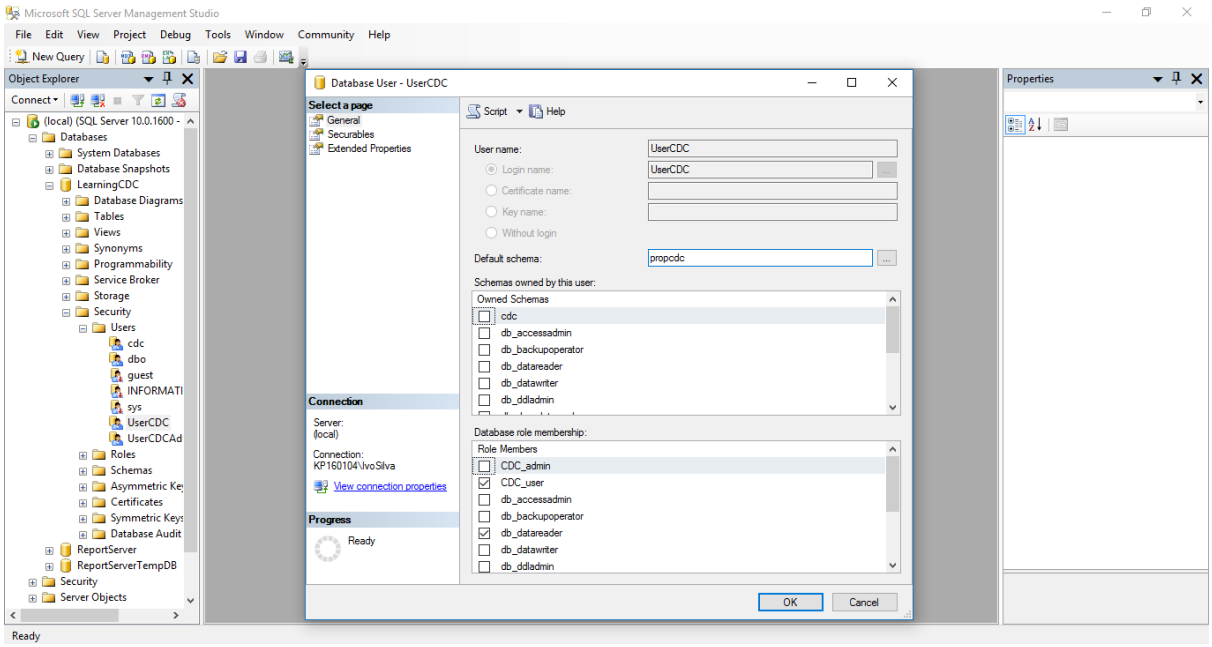
Alterar o schema pré-definido do user *UserCDCAdmin* para *propcdc*:



Abrir as propriedades do user *UserCDC*, em Server->Databases->NomedaBD->Security->Users->UserCDC->Properties:



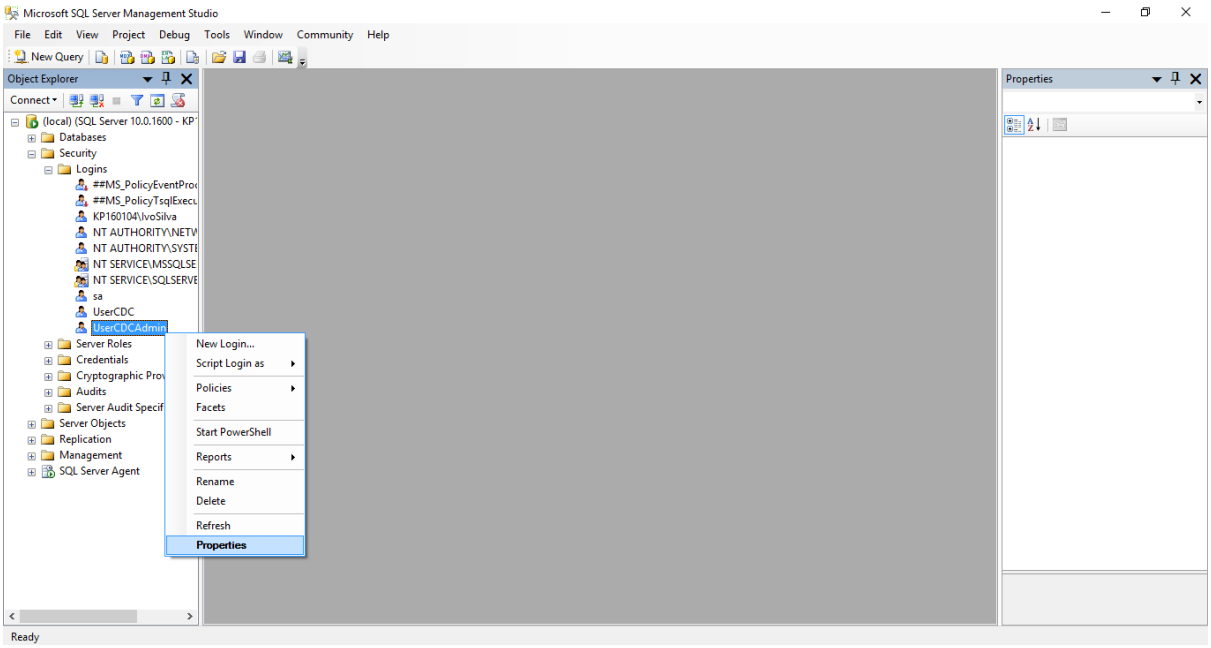
Alterar o schema pré-definido do user *UserCDC* para *propcdc*:



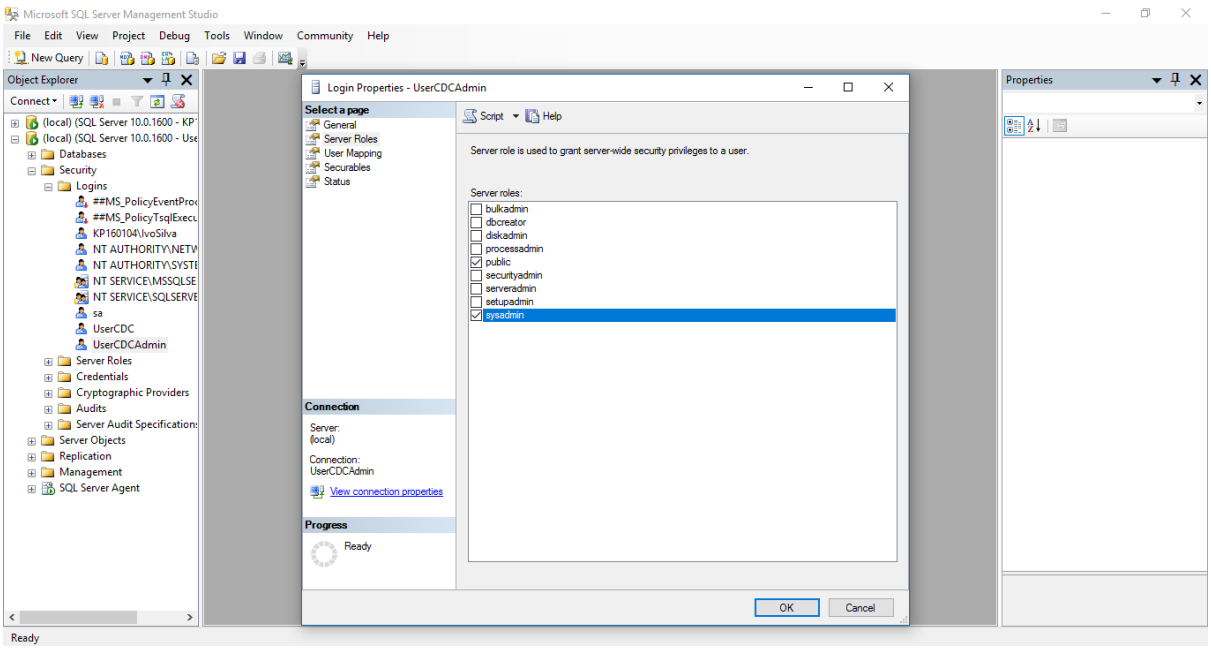
5. Como dar as permissões necessárias aos logins e users

O script *usp_InstalationScriptDBA.sql* configura a maioria das permissões necessárias para os users, mas como também é necessário dar algumas permissões aos logins e aos users não só na BD utilizada como também na *msdb*, o que também implica criar users na *msdb,* é mais simples terminar este processo utilizando os menus do Microsoft SQL Server Management Studio.

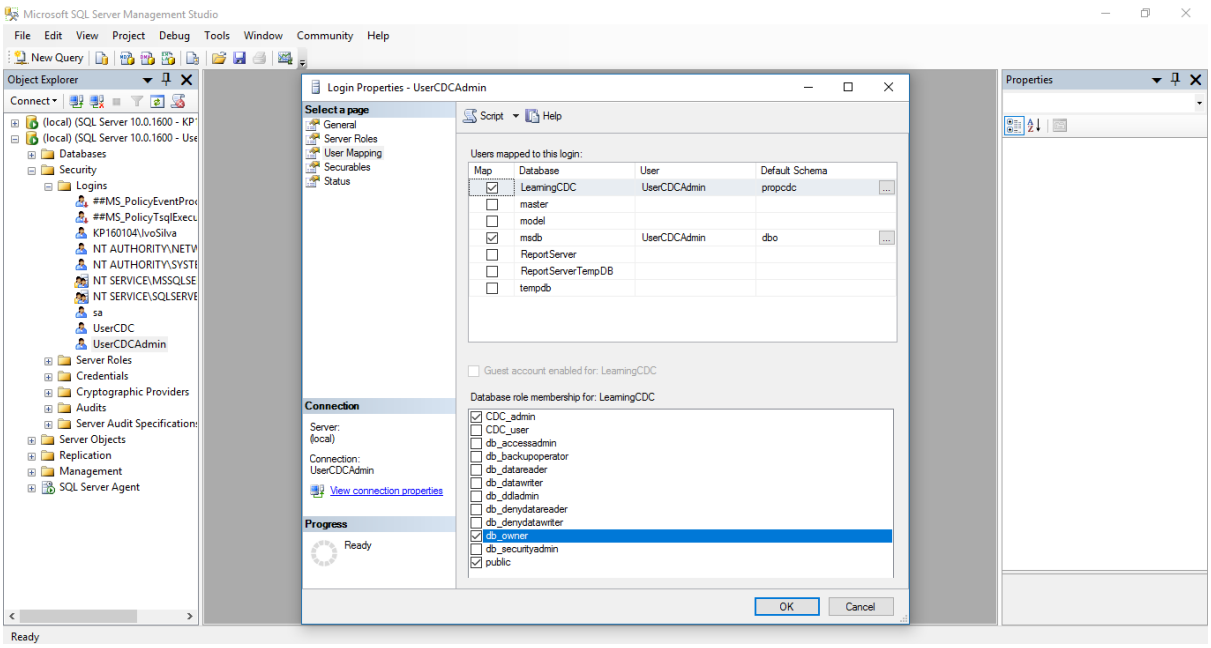
Alterar as permissões do login *UserCDCAdmin*, em Server->Databases->Security->Logins->*UserCDCAdmin*->Properties:



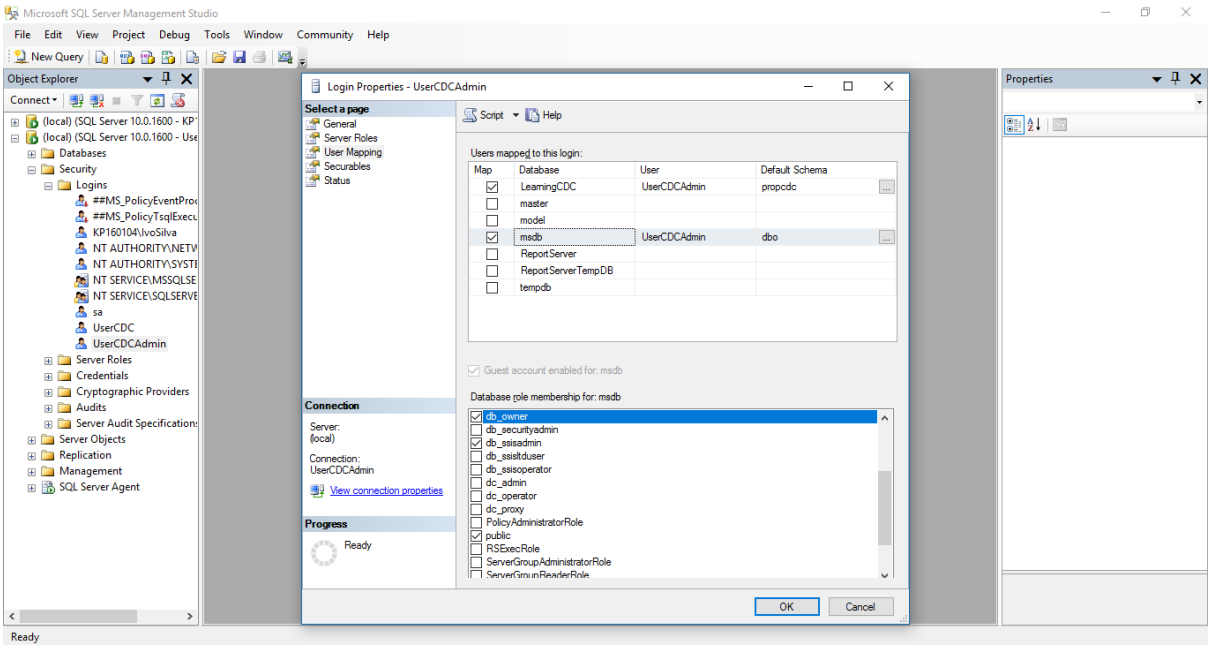
Na página de Server Roles, selecionar *sysadmin*:



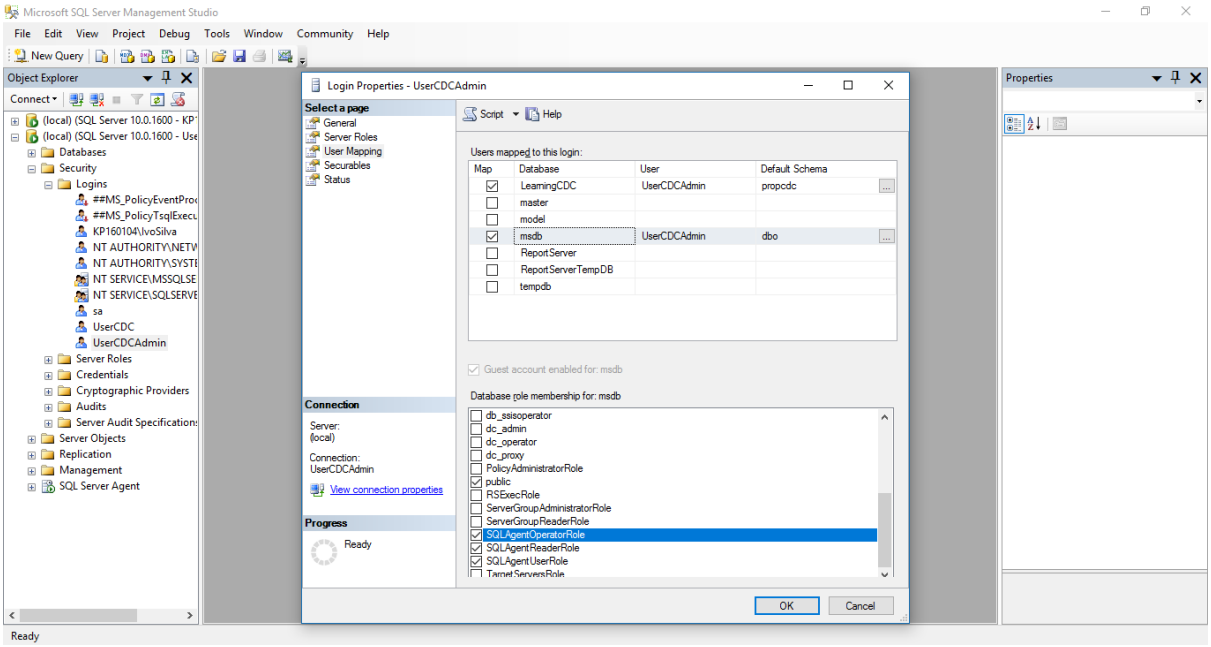
Na página User Mapping, selecionar a BD de exemplo em cima e *db_owner* em baixo:



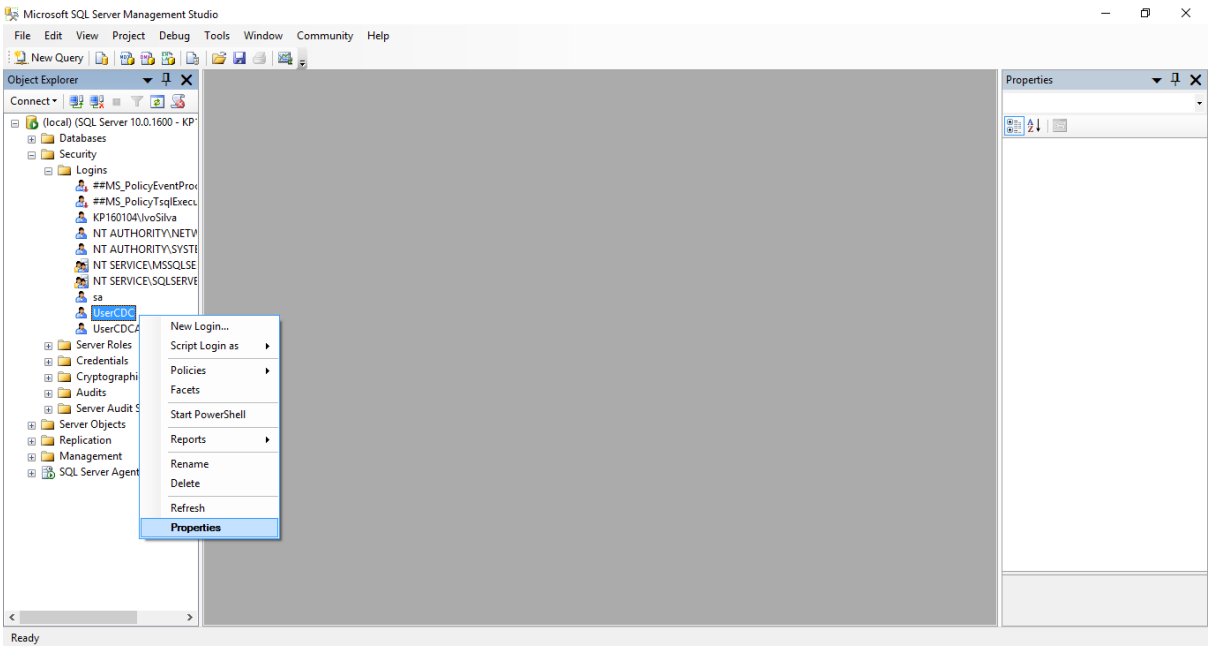
Na mesma página User Mapping, selecionar a BD *msdb* em cima e em baixo *db_owner* e *db_ssisadmin*:



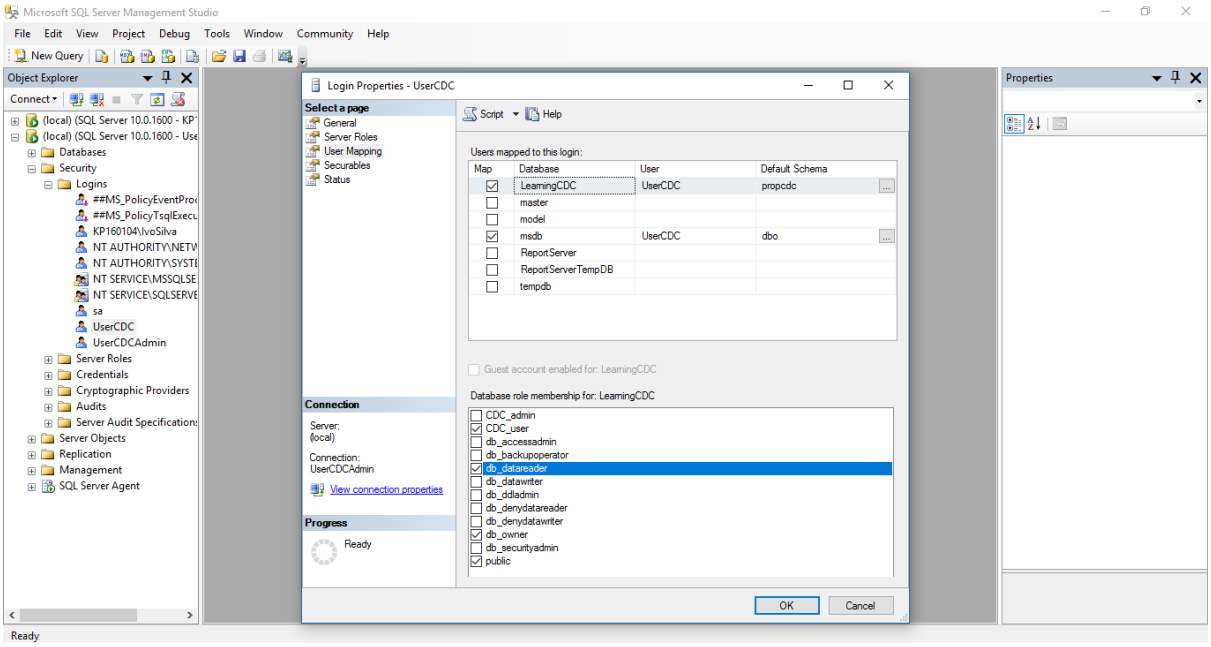
Na mesma página User Mapping e na mesma BD *msdb*, selecionar também *SQLAgentOperatorRole*:



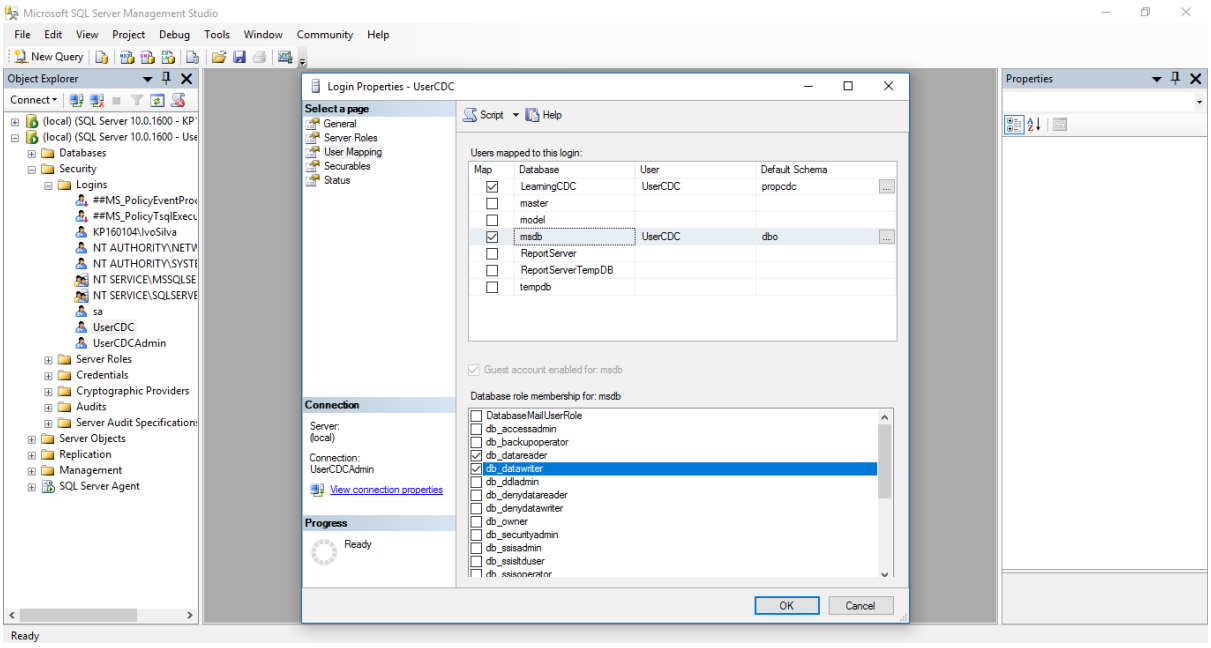
Alterar as permissões do login *UserCDC*, em Server->Databases->Security->Logins->*UserCDC*->Properties:



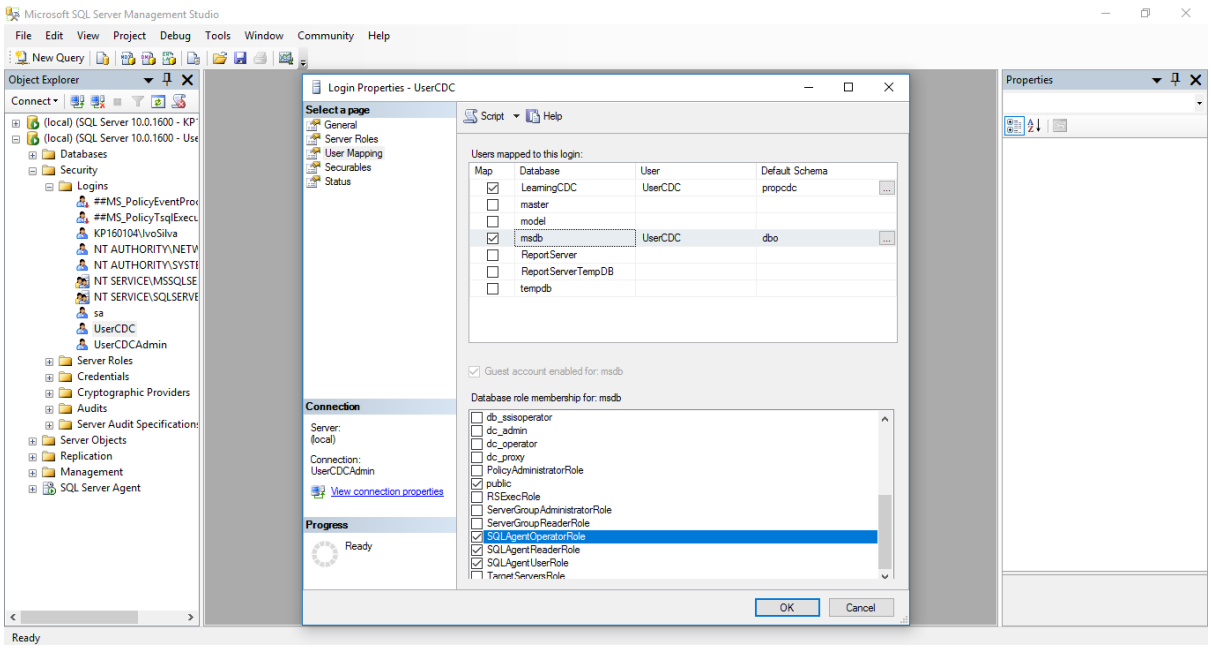
Na página User Mapping, selecionar a BD de exemplo em cima e em baixo *db_datareader* e *db_owner*.



Na mesma página User Mapping, selecionar a BD *msdb* em cima e em baixo *db_datareader* e *db_datawriter*.



Na mesma página User Mapping e na mesma BD *msdb* em cima, também selecionar em baixo *SQLAgentOperatorRole*.



6. Descrição do script de instalação para DBA

O script está no ficheiro *usp_InstalationScriptDBA.sql* e necessita de privilégios de *sysadmin* para ser executado.

6.1. Operações realizadas pelo script de instalação para DBA

1. Verifica se a BD tem um nível de compatibilidade superior ao SQL Server 2008 para garantir o funcionamento do conector.

2. Ativa o mecanismo CDC do SQL Server na BD em que é executado.

3. Define a BD atual como *Trustworthy* para permitir a utilização do SQLCLR (explicada neste documento em secção própria).

4. Cria o schema *propCDC* onde serão criados todos os procedimentos, funções, tabelas e assemblys deste conector.

5. Cria as roles *CDC_admin* e *CDC_user* e adiciona-lhes os utilizadores *UserCDCAdmin* e *UserCDC*:

6. Adiciona o utilizador *UserCDCAdmin*:

   - à role *db_owner* para lhe permitir gerir a BD.
   - *Alter Settings* para este poder alterar as definições da BD.

7. Dá permissões (*Grant*) à role *CDC_admin*:

   - *Create Procedure e Function* para este poder criar os procedimentos e funções que compõem este conector.
   - *View Definition* para poder consultar as tabelas de sistema.
   - *View Change Tracking* e *Control* apenas no schema *propCDC*.

8. Adiciona o utilizador *UserCDC* à role *CDC_user.*

9. Dá permissões (*Grant*) à role *CDC_user:*

   - *Create Table* para poder criar tabelas no seu esquema pré-definido.
   - *Execute* no schema *propCDC* para os seus membros poderem executar todos os procedimentos, funções e assemblys.
   - *Select* no schema *cdc* para poder aceder às alterações registadas e executar os procedimentos *usp_StartGetCDCAlterations*, *usp_StopGetCDCAlterations* e *usp_GetLastRows*.
   - *Control* no schema *dbo* para poder manipular todas as suas tabelas.

10. Elimina e cria as funções e o assembly do SQLCLR que utilizam a biblioteca .dll para exportar as alterações para o consumidor JSON. O script assume que a biblioteca se chama SqlWebRequest.dll e está na diretoria C: (raiz do sistema Windows que poder ser alterada da forma descrita no capítulo “Configurações” deste documento).

7. Descrição do script de instalação para o UserCDCAdmin

Apenas necessita das permissões que lhe são atribuídas pelo script do DBA.

O utilizador UserCDCAdmin deverá conseguir realizar uma gestão autónoma do conector e do mecanismo CDC do SQL Server após a instalação. Para isso deve ser o dono da base de dados onde o conector é instalado e o dono do schema *propCDC* podendo adicionar novos utilizadores do conector caso seja necessário. Este utilizador deve ter permissões para executar qualquer procedimento ou função deste conector.

7.1. Operações realizadas pelo script de instalação para o UserCDCAdmin

1. À semelhança do script para o DBA também verifica se a BD tem um nível de compatibilidade superior ao SQL Server 2008 para garantir o funcionamento do conector.

2. Declara e define variáveis para todos os valores inseridos na tabela de configuração *ConfigCDCTable* no schema *propCDC* que é utilizada pelos procedimentos do conector para obter os dados necessários ao seu funcionamento (o significado dos valores inseridos na tabela está descrito adiante neste documento).

3. Com recurso ao *sp_configure* altera os valores:

   - *show advanced options* para 1 de forma a permitir alterar as opções seguintes;
   - *clr enabled* para 1 de forma a permitir a criação do assembly da biblioteca .dll para exportar as alterações registadas pelo conector;
   - *user connections* para 0 de forma a colocar o limite de utilizadores ligado à BD no máximo suportado.

4. Remove e cria todas as funções e procedimentos que compõem o conector SQL Server.

5. Cria a tabela auxiliar CDC onde o conector regista as tabelas em que deve obter alterações com o nome que será definido na tabela de configuração na linha com a key *CDCauxTableName*.

6. Cria a tabela de configuração com os valores definidos nas variáveis no início deste script.

8. Lista de procedimentos para o UserCDCAdmin executar

1. O *usp_StartGetCDCAlterations* inicia o schedule que periodicamente recolhe as alterações das tabelas registadas no conector, formata o JSON e envia-as para o consumidor JSON.

2. O *usp_StopGetAlterations* elimina o schedule e pára o funcionamente do conector, mas não impede o mecanismo CDC do SQL Server de continuar a registar as alterações ocorridas nas tabelas.

3. O *usp_RegisterTableInCDC* regista uma tabela no conector e no mecanismo CDC do SQL Server para que este capture as alterações e o conector formate e exporte o JSON.

4. O *usp_UnregisterTableInCDC* elimina o registo de uma tabela no conector e no mecanismo CDC. Após a sua execução deixa de se capturar as alterações que ocorram na tabela.

5. O *usp_SyncCDCTable* repõe o funcionamento normal da captura de alterações de uma tabela após um *Alter Table Add Column*.

6. O *usp_SyncAllCDCTables* repõe o funcionamento normal da captura de alterações de todas as tabelas registadas no conector que tenham sofrido alterações.

7. O *usp_GetLastRows* permite ao consumidor JSON pedir as alterações de uma tabela após um timestamp.

9. Lista de procedimentos para o UserCDC executar

O *usp_StartGetCDCAlterations, o usp_StopGetAlterations* e o *usp_GetLastRows* que permitem realizar as operações descritas no capítulo anterior.

10. Lista de procedimentos, funções e tabelas auxiliares

10.1. Procedimentos

- usp_AnalyseStrings ;
- usp_GetCustomResults ;
- usp_ConvertJSONType ;
- usp_FetchTableAlterations ;
- usp_DescribeTable ;
- usp_MakeJSON ;
- usp_CleanupCDCAlterations ;
- usp_SendJSONByHTTPWithSQLCLR ;
- usp_GetTableColsList ;

10.2. Funções

- fn_get_webrequest
- fn_post_webrequest
- GetJSONDataType
- GetEquivalentDataType ;
- GetLiteralOperationType ;
- GetOnlyTableName ;

10.3. Tabelas

- ConfigCDCTableName: tabela de configuração do conector;
- CDCAuxTableRegistry: tabela de registo das tabelas cujas alterações o conector deve considerar.

11. Configurações

11.1. Na tabela de configuração

Esta tabela organiza-se em duas colunas *configKey* e *configValue* com uma configuração por linha:

- *roleName* é o nome da role do utilizador CDC *UserCDC;*
- *CDCschemaName* é o nome do schema criado pelo script *usp_InstalationScriptDBA.sql* e utilizado pelo *usp_InstalationScriptAdminCDC.sql* para criar as funções, procedimentos e tabelas;
- *consumerJSONip* é o endereço IP do consumidor JSON;
- *consumerJSONport* é o porto do consumidor JSON;
- *consumerJSONtopic* é o nome do tópico do consumidor JSON onde se publicam as alterações;
- *CDCauxTableName* é o nome da tabela auxiliar onde o conector regista as tabelas do CDC para obter as alterações;
- *sendCDCAlterationsJobName* é o nome do job que exporta as alterações e as apaga quando são publicadas;
- *scheduleCDCName* é o nome do schedule que executa o job anterior;
- *searchAlterationsScheduleFrequencyInMinutes* é o número de minutos entre cada execução do schedule.

Todos os parâmetros presentes na tabela de configuração *ConfigCDCTable* podem ser alterados nas definições das variáveis com o mesmo nome no início do script *usp_InstalationScriptAdminCDC.sql.*

11.2. Na tabela de configuração

configKey	***configValue***
roleName	CDC_user
CDCschemaName	propcdc
consumerJSONip	10.64.104.4
consumerJSONport	8082
consumerJSONtopic	IVO
CDCauxTableName	CDCAuxTableRegistry
sendCDCAlterationsJobName	send_json_by_http_with_sqlclr
scheduleCDCName	scheduled_get_alterations_in_json
searchAlterationsScheduleFrequencyInMinutes	1

11.3. Incluídas no código T-SQL

O nome e a localização da biblioteca *.dll* que fornece as funções para exportar o JSON com recurso ao SQLCLR pode ser alterado no script *usp_InstalationScriptAdminCDC.sql* pesquisando por 'CREATE ASSEMBLY SqlWebRequest'. O nome da biblioteca deve ser antecedido pelo caminho completo até à mesma e finalizado com a sua extensão. O *Content-Type* JSON da mensagem POST HTTP enviada para o consumidor JSON pré-definido é "application/vnd.kafka.json.v1+json" mas pode ser alterado no código C# pesquisando por “req.ContentType”.

O nome dos logins, utilizadores, schema e role *CDC_admin* pode ser alterado pesquisando no script *usp_InstalationScriptDBA.sql* e substituindo todas as suas ocorrências.

O nome da role *CDC_user* pode ser redefinido alterando a variável *@propRole* no início do script *usp_InstalationScriptAdminCDC.sql* e pesquisando no script *usp_InstalationScriptDBA.sql* para substituir todas as suas ocorrências.

O nome da tabela de configuração *ConfigCDCTable* pode ser alterado substituindo a sua definição nas variáveis *@ConfigCDCTableName* no início do script *usp_InstalationScriptAdminCDC.sql* e dos procedimentos *usp_StartGetCDCAlterations, usp_StopGetCDCAlterations, usp_RegisterTableInCDC, usp_UnregisterTableInCDC, usp_SyncCDCTable, usp_SyncAllCDCTables, usp_GetLastRows, usp_CleanupCDCAlterations.*

O nome das colunas da tabela de configuração só pode ser alterado pesquisando por *configKey* e *configValue* no script *usp_InstalationScriptAdminCDC.sql* e substituindo todas as suas ocorrências nos procedimentos.

12. Código C# para exportação do JSON

Para gerar a biblioteca *SqlWebRequest.dll* basta colocar este código no Microsoft Visual Studio e compilá-lo usando o .NET Framework 2.0 [1].

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Collections;
using System.Globalization;

// For the SQL Server integration
using Microsoft.SqlServer.Server;

// Other things we need for WebRequest
using System.Net;
using System.Text;
using System.IO;
public partial class Functions
{
    // Function to return a web URL as a string value.
    [Microsoft.SqlServer.Server.SqlFunction(DataAccess =
DataAccessKind.Read)]
    public static SqlString GET(SqlString uri, SqlString username,
SqlString passwd)
    {
        // The SqlPipe is how we send data back to the caller
        SqlPipe pipe = SqlContext.Pipe;
        SqlString document;

        // Set up the request, including authentication
        WebRequest req = WebRequest.Create(Convert.ToString(uri));
        if (Convert.ToString(username) != null &
Convert.ToString(username) != "")
```

```
        {
            req.Credentials = new NetworkCredential(
                Convert.ToString(username),
                Convert.ToString(passwd));
        }
        ((HttpWebRequest)req).UserAgent = "CLR web client on SQL
Server";

        // Fire off the request and retrieve the response.
        // We'll put the response in the string variable "document".
        WebResponse resp = req.GetResponse();
        Stream dataStream = resp.GetResponseStream();
        StreamReader rdr = new StreamReader(dataStream);
        document = (SqlString)rdr.ReadToEnd();

        // Close up everything...
        rdr.Close();
        dataStream.Close();
        resp.Close();

        // .. and return the output to the caller.
        return (document);
    }

    // Function to submit a HTTP POST and return the resulting
output.
    [Microsoft.SqlServer.Server.SqlFunction(DataAccess =
DataAccessKind.Read)]
    public static SqlString POST(SqlString uri, SqlString postData)
    {
        SqlPipe pipe = SqlContext.Pipe;
        SqlString document;
        byte[] postByteArray =
Encoding.UTF8.GetBytes(Convert.ToString(postData));
```

```
         // Set up the request, including authentication,
        // method=POST and encoding:
        WebRequest req = WebRequest.Create(Convert.ToString(uri));
        ((HttpWebRequest)req).UserAgent = "CLR web client on SQL
Server";
        req.Method = "POST";
        req.ContentType = "application/vnd.kafka.json.v1+json";

        // Submit the POST data
        Stream dataStream = req.GetRequestStream();
        dataStream.Write(postByteArray, 0, postByteArray.Length);
        dataStream.Close();

        // Collect the response, put it in the string variable
"document"
        WebResponse resp = req.GetResponse();
        dataStream = resp.GetResponseStream();
        StreamReader rdr = new StreamReader(dataStream);
        document = (SqlString)rdr.ReadToEnd();

        // Close up and return
        rdr.Close();
        dataStream.Close();
        resp.Close();

        return (document);
    }
}
```

13. Conclusão

Seguindo os passos de instalação descritos neste documento é possível colocar o conector SQL Server em funcionamento numa base de dados em operação utilizando o mecanismo CDC integrado para detetar alterações para garantir um impacto reduzido no desempenho da mesma.

Com a instalação deste conector é possível integrar e persistir dados de sistemas em operação publicando-os num intermediário Kafka que depois pode alimentar subscritores com requisitos muito distintos, permitindo inclusive analisar posteriormente os dados recolhidos.

14. Bibliografia

1. Web requests using a CLR procedure | sqlsunday.com, https://sqlsunday.com/2013/03/03/web-requests-using-clr-proc/
2. Documentação Microsoft SQL Server, https://docs.microsoft.com/en-us/sql/sql-hub-menu
3. GitHub - confluentinc/kafka-rest: Confluent REST Proxy for Kafka, https://github.com/confluentinc/kafka-rest