SQL Server 2008
Seguridad

Marta Zorrilla

UNIVERSIDAD DE CANTABRIA

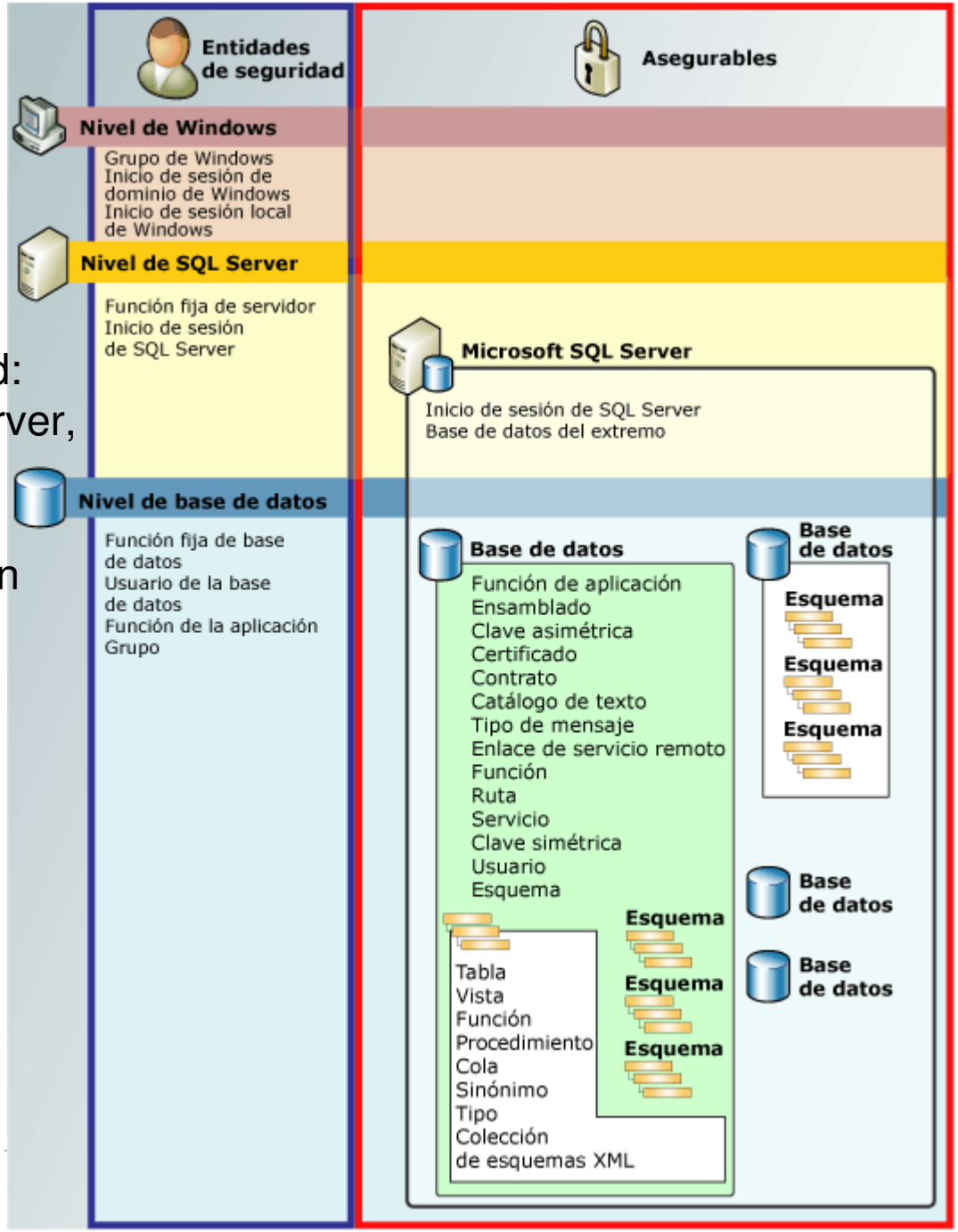
Tabla de contenidos

Modelo de Seguridad en SQL Server

Jerarquía de Seguridad

Principals: entidades de seguridad: Usuarios windows, usuarios sql server, Usuarios de BD

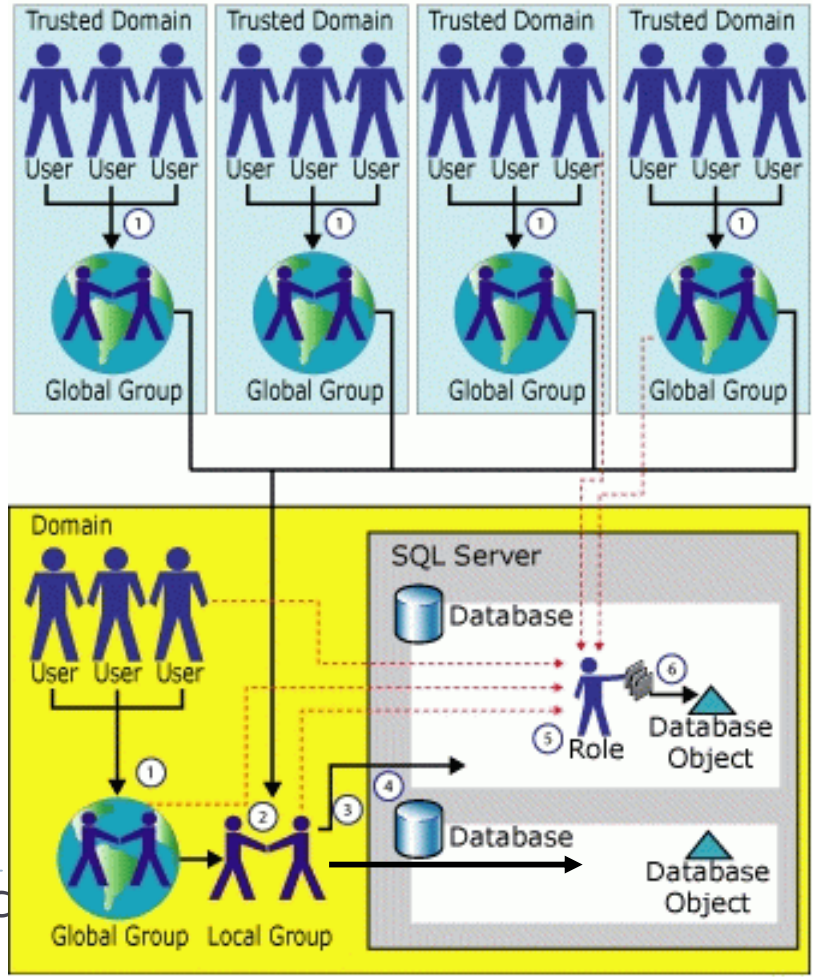
Asegurables: recursos que pueden ser protegidos



Inicios de sesión - Usuarios

- Modo de autentificación (acceso al servidor) **(Inicio de sesión: Login)**
  - Windows (S.O.)
  - Servidor SQL Server

- Acceso y gestión de una BD (autorización: User)
  - Permisos a
    - objetos de BD
    - ejecución de sentencias
  - Permisos a través de roles:
    - del servidor o de BD
    - definidos por el usuario



Inicios de sesión preestablecidos

- Al instalarse SQL Server se crean 2 inicios de sesión:
  - la cuenta de servicio que se utiliza para iniciar el servicio SQL Server. Puede cambiarse sus privilegios.
  - el usuario **sa**. Este no puede eliminarse ni modificarse. No estará disponible si solo está configurada la autentificación de Windows.

- Pueden realizar cualquier tarea en SQL Server (pertenecen al rol de servidor **sysadmin**)

Tipos de Inicios de sesión

- Windows
  - Usuario
  - Grupo de usuarios
- SQL Server
- Certificado
- Clave Asimétrica
- Asociados a Credenciales
  - Para acceso a recursos externos

Tipos de Inicios de sesión (y 2)

- Opciones de Administración
  - Podemos forzar el cambio de contraseña en el primer inicio de sesión: MUST_CHANGE
  - Exigir directivas de contraseña
  - Desbloquear Cuentas: UNLOCK
  - Deshabilitar un inicio de sesión: DISABLE
  - Establecer una BD de conexión predeterminada

Crear Inicios de sesión (y 3)

- Gráficamente SSMS
- CREATE LOGIN
  - *CREATE LOGIN Pepe WITH PASSWORD = 'Passwd' MUST_CHANGE*
  - *CREATE LOGIN [UNICAN\pepe] FROM WINDOWS*
- DROP LOGIN / ALTER LOGIN

- La información se almacena en
  - sys.server_principals: Contiene una fila por cada entidad de seguridad del servidor.
  - sys.sql_logins: Devuelve una fila por cada inicio de sesión de SQL.

Roles fijos de Servidor

- Cada rol agrupa un conjunto de permisos
- Facilitan la admón. de seguridad
- Se definen a nivel de servidor. Independiente, por tanto, de las bases de datos
- Un inicio de sesión puede pertenecer a cero o más roles de servidor
- Un inicio de sesión que pertenezca a un rol de servidor adquiere los permisos de ese rol
- Son fijos:
  - No se pueden modificar sus permisos
  - No pueden eliminarse
  - No pueden añadirse nuevos roles de servidor

Roles fijos de Servidor (y 2)

Fixed Server Role	Description
Sysadmin	Performs any activity in SQL Server.
Serveradmin	Configures server-wide configuration options, shuts down the server.
Setupadmin	Adds and removes linked servers, and executes some system stored procedures, such as **sp_serveroption**.
securityadmin	Manages server-wide security settings, including linked servers, and CREATE DATABASE permissions. Resets passwords for SQL Server authentication logins.
processadmin	Terminates processes running in SQL Server.
dbcreator	Creates, alters, drops, and restores any database.
diskadmin	Manages disk files.
Bulkadmin	Allows a non-sysadmin user to run the **bulkadmin** statement.

- sp_addsrvrolemember /sp_dropsrvrolemember
- sys.server_role_members
- sp_srvrolepermission
- sys.server_permissions

Seguridad de base de datos

- Los siguientes inicios de sesión pueden conectarse a una BD:
  - sysadmin
  - Propietario de la BD
  - Usuarios de la BD
  - Cualquier inicio de sesión si existe el usuario guest y tiene permiso

- Usuario de BD
  - Definido a nivel de BD
  - Corresponde con un inicio de sesión

Conceder acceso a una BD

- Pueden conceder permisos:
  - sysadmin
  - Propietario de la BD
  - Usuario con rol db_owner
  - Usuario con rol db_accessadmin
- Al conceder acceso a un inicio de sesión a una base de datos:
  - Se crea el usuario correspondiente en esa BD
  - Pertenece al rol public
- Para conceder permisos → sp_grantdbaccess y sp_adduser
- Para quitar el acceso → sp_revokedbaccess
- Para reasignar inicios de sesión con usuarios → sp_change_users_login
- Ver info de usuarios → sp_helpuser

Usuarios por defecto en una BD

- dbo:
  - Propietario. No puede ser borrado de la BD
- Guest:
  - Permite a usuarios que no tienen cuenta en la BD, que accedan a ella, pero hay que darle permiso explícitamente
- Information_schema
  - Permite ver los metadatos de SQL Server
- sys
  - Permite consultar las tablas y vistas del sistema, procedimientos extendidos y otros objetos del catálogo del sistema

- Mostrar usuarios de una base de datos

  Select * from sys.database_principals

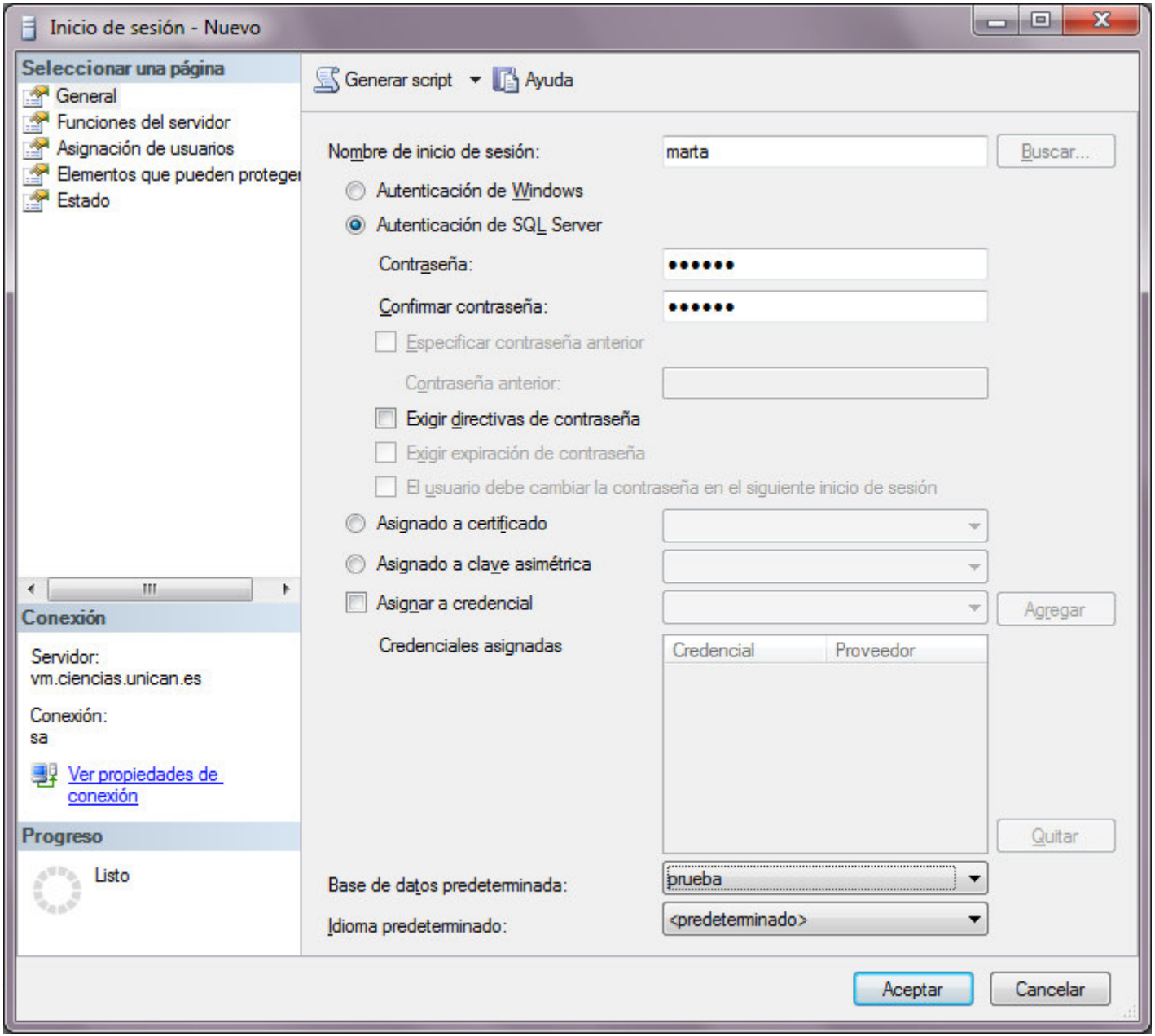
Roles fijos de base de datos

Fixed database role	Description
db_owner	Performs all maintenance and configuration activities in the database.
db_accessadmin	Adds or removes access for Windows users, groups, and SQL Server logins.
db_datareader	Reads all data from all user tables.
db_datawriter	Adds, deletes, or changes data in all user tables.
db_ddladmin	Runs any Data Definition Language (DDL) command in a database.
db_securityadmin	Modifies role membership and manages permissions.
db_backupoperator	Backs up the database.
db_denydatareader	Cannot read any data in user tables within a database.
db_denydatawriter	Cannot add, modify, or delete data in any user tables or views.

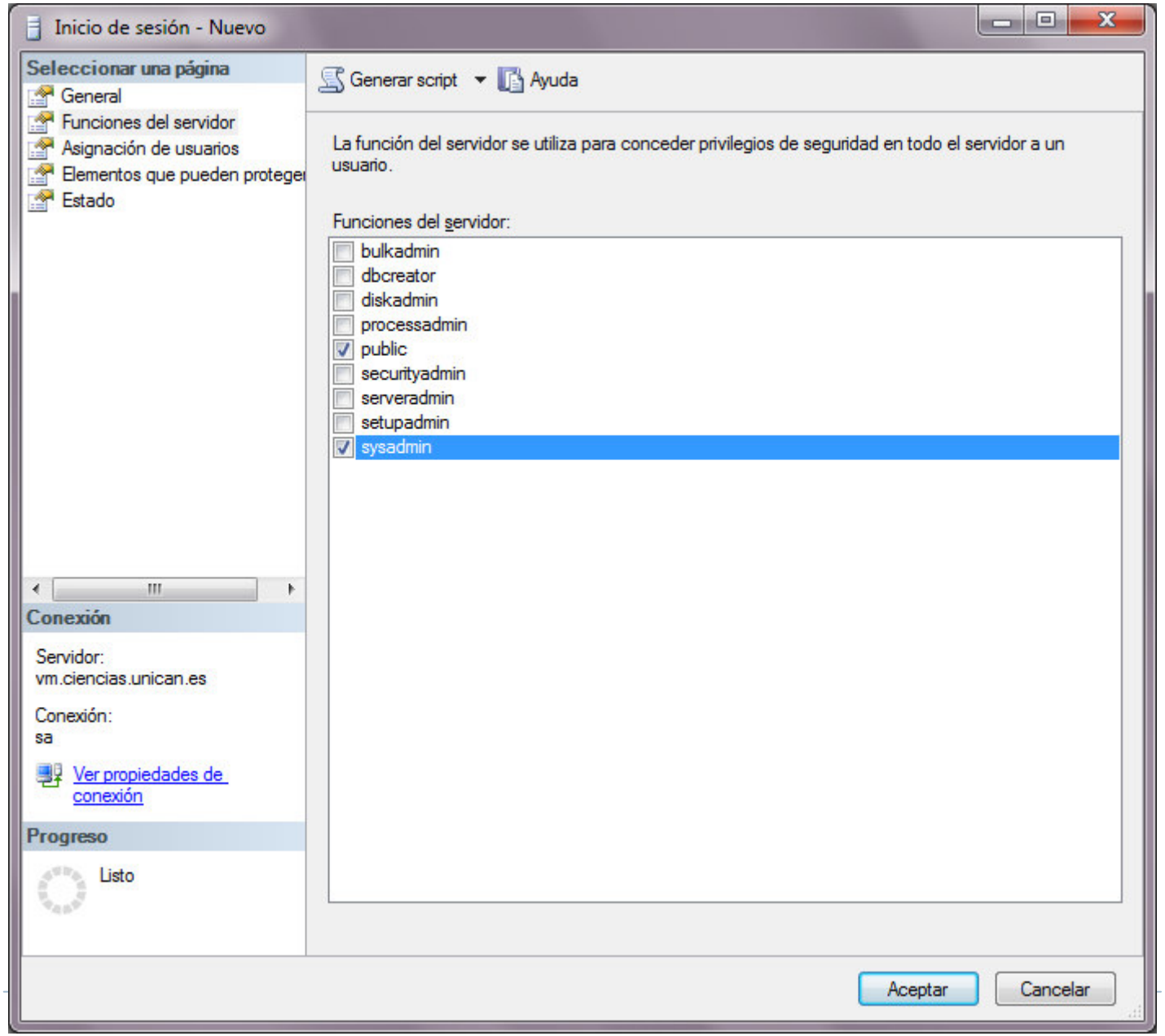
Roles definidos por el usuario

- Agrupan un conjunto de permisos
- No tienen permisos predefinidos
- Los permisos se establecen por:
  - Pertenencia a otros roles
  - Permisos de sentencias
  - Permisos específicos de objetos
- Pueden ser:
  - Rol estándar
  - Rol de aplicación: establecer permisos a una aplicación sobre la BD
- Los pueden gestionar: sysadmin, propietario de BD, db_owner, db_securityadmin

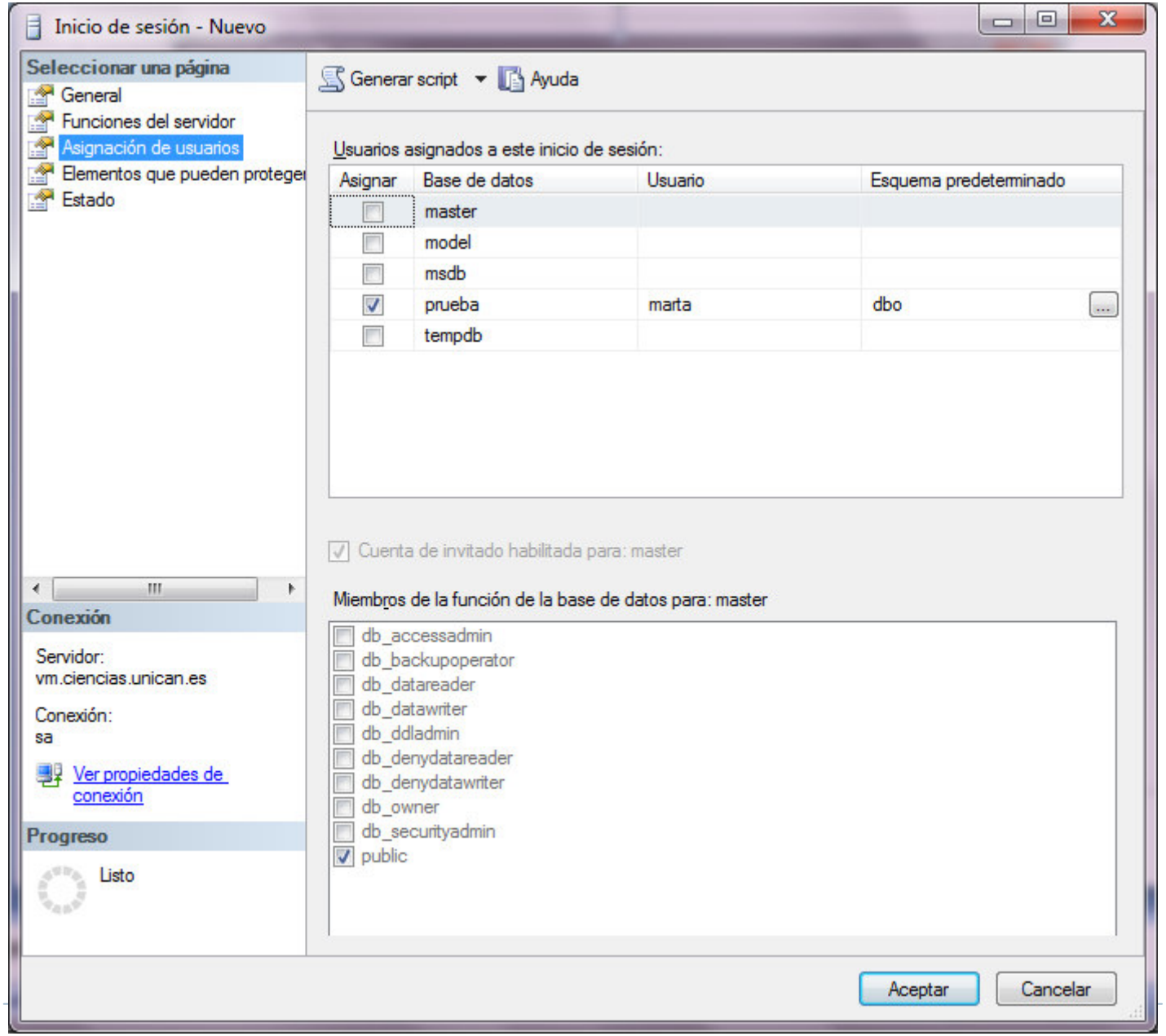
Creación de un inicio de sesión



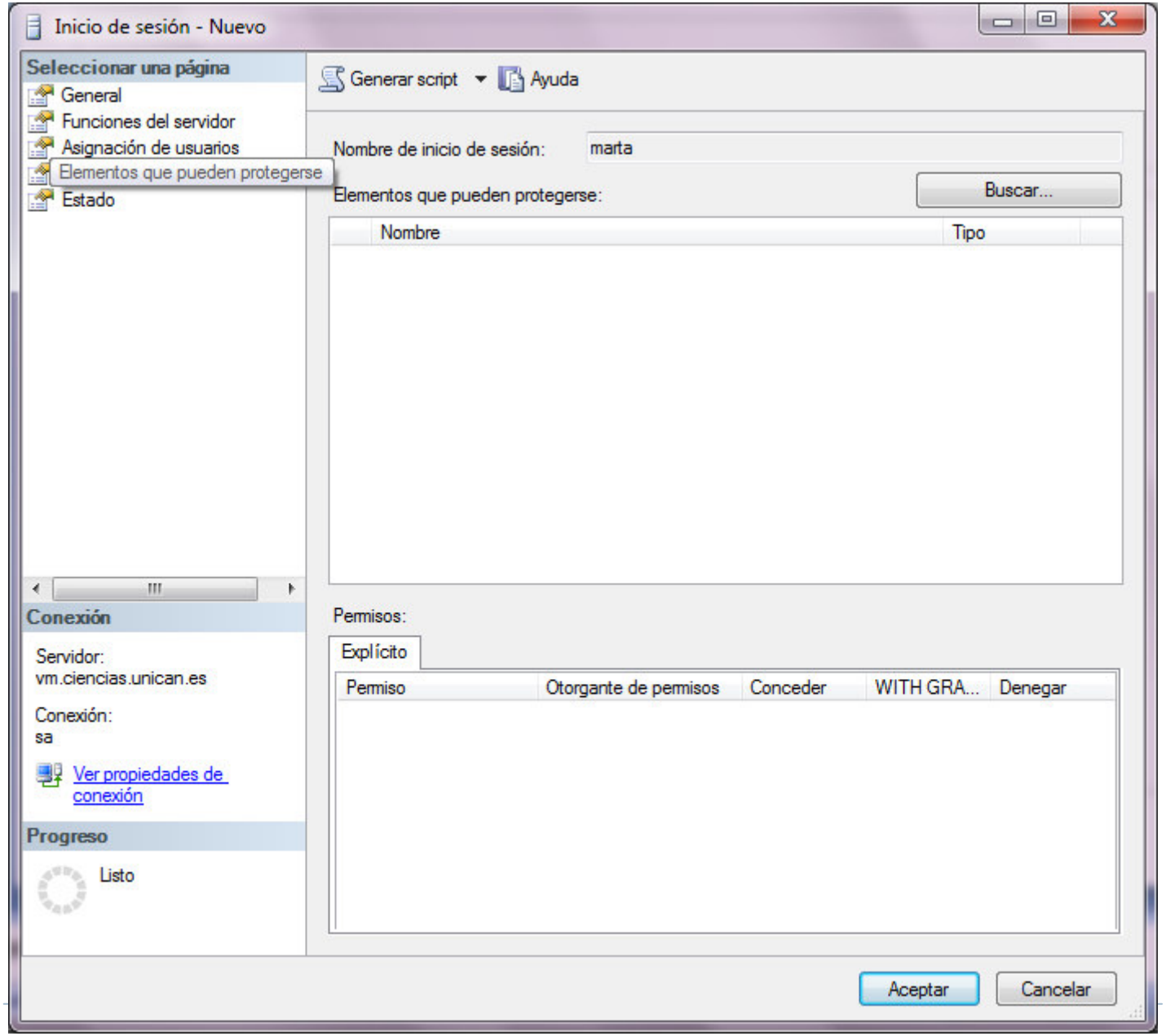
Asignación de un rol de servidor



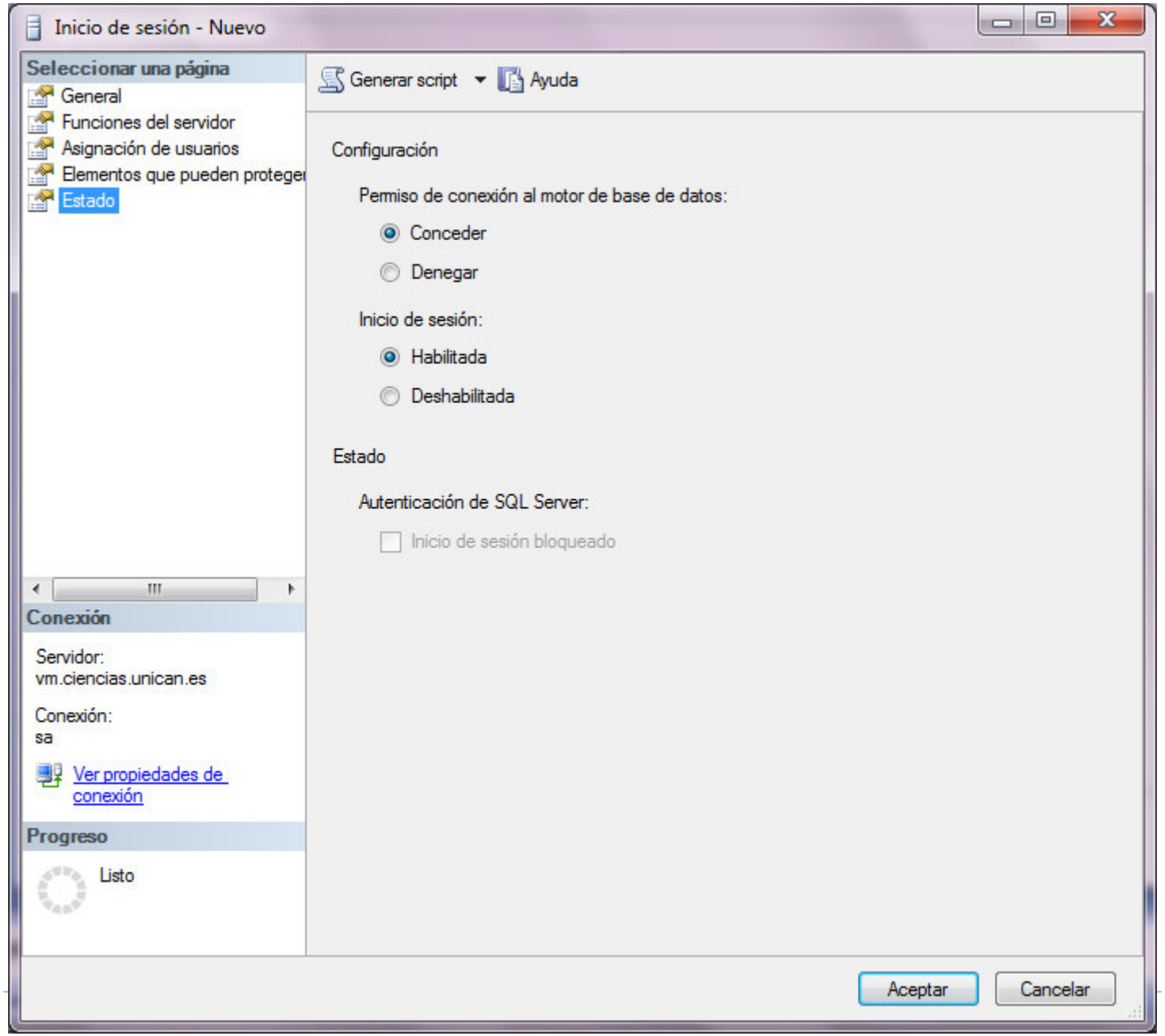
Asignación de usuarios



Elementos que pueden proteger



Estado del inicio de sesión



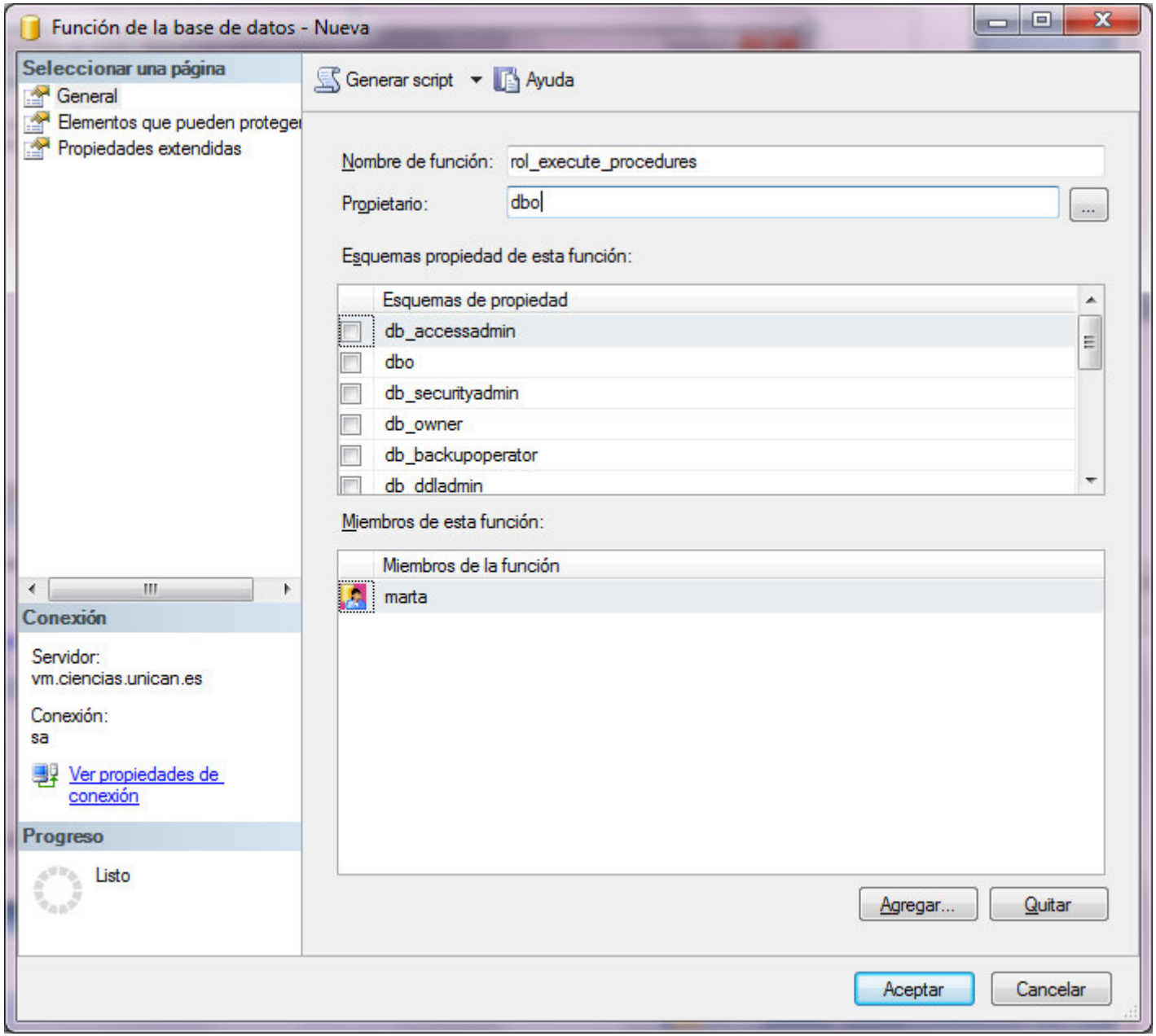
Usuarios de Base de Datos

- Crear un Usuario
  - *CREATE USER <usuario> FOR LOGIN <login> WITH DEFAULT_SCHEMA = <schema>*
  - *Podemos crear un usuario sin asociar: WITHOUT LOGIN*
- Modificar
  - *ALTER USER*
- Eliminar
  - *DROP USER*
  - *No si es propietario de objetos*
- Invitado
  - *GRANT CONNECT TO GUEST*
- Información sobre usuarios en: sys.database_principals

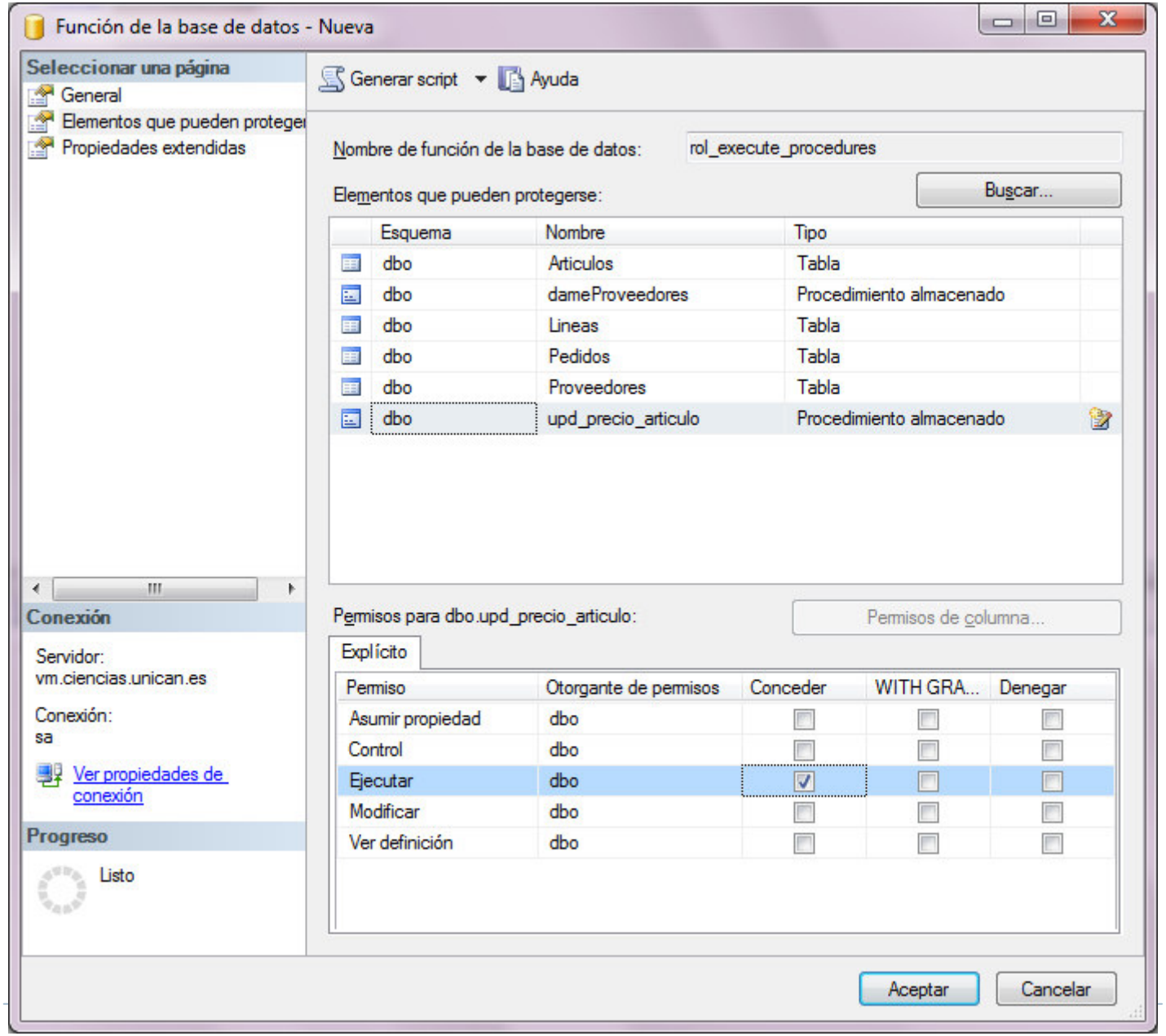
Roles o funciones de Base de Datos

- CREATE ROLE <nombre>
- Asignar a un usuario
  - sp_addrolemember <role>,<usuario>
- Ver información en: sys.database_role_members
- Rol Public

Crear un rol de BD



Establecer protección a asegurables



Usuarios de BD y esquemas

- Separación de principales y esquemas
  - Principal: Entidad contra la que se securiza un objeto
  - Esquema: Contenedor de objetos:

- ANSI SQL-92:
  - Colección de objetos de la BBDD cuyo propietario es un único principal y forma un único espacio de nombres (conjunto de objetos que no pueden tener nombres duplicados)

servidor.basededatos.esquema.objeto

Esquemas (y 2)

- Los objetos ahora pertenecen al esquema de forma independiente al usuario
- Beneficios
  - El borrado de un usuario no requiere que tengamos que renombrar los objetos
  - Resolución de nombres uniforme
  - Gestión de permisos a nivel de esquema
- Nuevas sentencias DDL:
  - CREATE/ALTER/DROP para USER, ROLE y SCHEMA

Esquemas (y 3)

- Una BBDD puede contener múltiples esquemas
- Cada esquema tiene un propietario (principal): usuario o rol
- Cada usuario tiene un **default schema** para resolución de nombres
- La mayoría de los objetos de la BBDD residen en esquemas
- Creación de objetos dentro de un esquema requiere permisos CREATE y ALTER o CONTROL sobre el esquema

Conceder permisos

Concede permisos en un asegurable a una entidad de seguridad.

```
GRANT { ALL [ PRIVILEGES ] }
    | permission [ ( column [ ,...n ] ) ] [ ,...n ]
    [ ON [ class :: ] securable ] TO principal [ ,...n ]
    [ WITH GRANT OPTION ] [ AS principal ]
```

- ALL : Esta opción no concede todos los permisos posibles.

-Si el asegurable es una base de datos, "ALL" significa BACKUP DATABASE, BACKUP LOG, CREATE DATABASE, CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE RULE, CREATE TABLE y CREATE VIEW.

- Si es una función escalar, "ALL" significa EXECUTE y REFERENCES.

- Si es una función con valores de tabla, "ALL" se refiere a DELETE, INSERT, REFERENCES, SELECT y UPDATE.

- Si es un proc. almacenado, "ALL" significa DELETE, EXECUTE, INSERT, SELECT y UPDATE.

- Si es una tabla o vista, "ALL" significa DELETE, INSERT, REFERENCES, SELECT y UPDATE.

- WITH GRANT OPTION: el usuario al que se le otorga permiso, puede a su vez, otorgárselo a otro.

Establecer permisos: ejemplos

• Permitir a los usuarios Maria, Juan y Marta crear bases de datos y tablas

```
GRANT CREATE DATABASE, CREATE TABLE
TO Maria, Juan, [Servidor\Marta]
```

• Permitir a María y a Juan, insertar, modificar y borrar en la tabla autores.

```
GRANT INSERT, UPDATE, DELETE ON autores
TO Maria, Juan
```

• Permitir a María actualizar el importe del préstamo.

```
GRANT UPDATE( importe ) ON prestamo
TO Maria
```

Revocar permisos

Quita un permiso concedido o denegado previamente. Su sintaxis es:

```
REVOKE [ GRANT OPTION FOR ]
   {   [ ALL [ PRIVILEGES ] ]
         | permission [ ( column [ ,...n ] ) ] [ ,...n ] }
   [ ON [ class :: ] securable ]
   [TO | FROM } principal [ ,...n ]
   [ CASCADE] [ AS principal ]
```

- **GRANT OPTION FOR** : se quita al usuario la capacidad de dar o quitar permisos que le fueron concedidos por la cláusula WITH GRANT OPTION
- ***permiso***: SELECT, INSERT, DELETE, UPDATE, REFERENCES, EXECUTE, CREATE, etc.
- **CASCADE** : se quita el permiso al usuario/role y a los usuarios/roles a los que dio permiso, si se le concedió GRANT OPTION.
- **AS** : usuario o role que quita el permiso

Revocar permisos: ejemplos

- Impedir a los usuarios Maria y Marta crear vistas en la BD activa.

```
REVOKE CREATE VIEW
TO Maria, [Servidor\Marta]
```

- Impedir que María ejecute la función “dameprecio”.

```
REVOKE SELECT ON dbo.dameprecio
TO Maria
```

Denegar permisos

Deniega un permiso a una entidad de seguridad. Evita que la entidad de seguridad herede permisos por su pertenencia a grupos o funciones. Su sintaxis es:

```
DENY  {   [ ALL [ PRIVILEGES ] ]
          | permission [ ( column [ ,...n ] ) ] [ ,...n ] }
   [ ON [ class :: ] securable ]  TO principal [ ,...n ]
   [ CASCADE] [ AS principal ]
```

- ***permiso***: SELECT, INSERT, DELETE, UPDATE, REFERENCES, EXECUTE, CREATE, etc.
- **CASCADE** : Indica que el permiso se deniega para la entidad de seguridad especificada y para el resto de entidades de seguridad a las que ésta concedió el permiso. Es obligatorio cuando la entidad de seguridad tiene el permiso con GRANT OPTION.
- **AS** : usuario o role que quita el permiso

Permisos efectivos

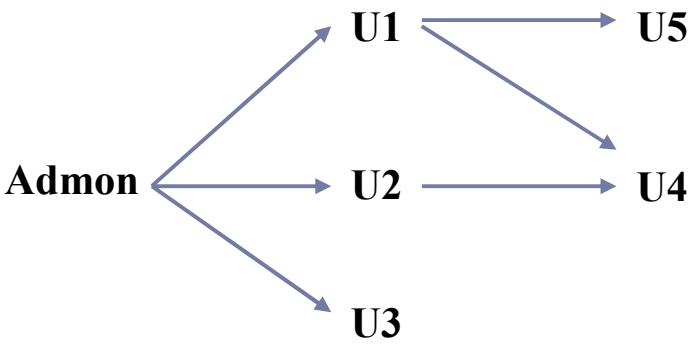
- $PE = C - D$
- $C = O_c + S_c + R_c$
- $D = O_d + S_d + R_d$

- PE: permisos efectivos
- C: permisos concedidos
- D: permisos denegados
- O_c: permisos de objeto concedidos
- S_c: permisos de sentencias concedidos
- R_c: permisos concedidos de los roles a los que pertenezca
- O_d: permisos de objeto denegados
- S_d: permisos de sentencias denegados
- R_d: permisos denegados de los roles a los que pertenezca

A tener en cuenta

SQL Server establece el rol PUBLIC a todos los usuarios de la BD. Para aquellos usuarios que no tienen cuenta en la BD, pero sí acceso al gestor, pueden conectarse a la BD como GUEST, si este usuario está habilitado en ella (GRANT CONNECT TO GUEST).

Se ha de tener cuidado respecto a la manera en que se establecen las autorizaciones, si se quiere garantizar que luego se puedan quitar.



Limitaciones de seguridad

- No se puede establecer privilegios a nivel de fila (p. ej. cada alumno sólo vea sus notas)
- Hay extensiones para proporcionar control de acceso en el nivel de las filas y para trabajar con gran número de usuarios pero aún no están normalizadas.
- Utilizar vistas para restringir la información.
- Establecer la seguridad en aplicaciones de BD:

  - Usuarios
    - Usuarios de domino / de gestor con sus privilegios
    - Usuarios de dominio / de gestor con rol de aplicación
    - Usuario único con privilegios
  - Crear BD de seguridad donde se establece con detalle las acciones que cada usuario de aplicación puede hacer
    - Código de aplicación se entremezcla con el de autorización
    - Más difícil de garantizar la existencia de “agujeros” de seguridad

Estrategias de seguridad

- Uso de vistas y funciones
  - Dar permisos a vistas y funciones en lugar de a las propias tablas
  - Ocultan la complejidad de la BD
  - Permiten gestionar el acceso a nivel de columna
- Uso de procedimientos almacenados
  - Impiden operaciones incorrectas asegurando las reglas de negocio
  - Los usuarios no necesitan tener permiso para acceder a las tablas, solo permiso de ejecución de los procedimientos
  - Permiten establecer el nivel de seguridad más fino (contexto)

Encriptación de datos

- ¿para qué?
  - Evitar acceso a datos sensibles
  - Evitar robo de copias de seguridad con datos sensibles
- ¿qué técnicas?
  - Encriptación a nivel de columna
  - Encriptación transparente (TDE), afecta a toda la BD
- ¿coste?
  - Mayor sobrecarga y puede afectar al rendimiento
  - Requiere una estrategia para la definición y mantenimiento de claves, passwords y certificados
  - Por ello no debe considerarse para todos los datos y conexiones

Encriptación de datos (y 2)

- Encriptación a nivel de columna
  - Mediante certificados, keys o frases
  - Requiere el uso de funciones específicas
    - EncrypByCert() – DecryptByCert()
    - EncrypyAsymkey() – DecryptByAsymKey()
    - Encrypykey() – DecryptBKey()
    - EncrypyPassphrase() – DecryptByPassphrase()
  - Encriptación transparente (TDE), afecta a toda la BD
    - No protege las comunicaciones entre aplicación cliente y servidor
    - No encripta FILESCREAM
    - No impide al DBA ver los datos
    - Puede caer el rendimiento si conviven BD TDE y sin encriptar

Encriptación de datos : ejemplo

```
USE AdventureWorks2008R2;

--If there is no master key, create one now.
IF NOT EXISTS
   (SELECT * FROM sys.symmetric_keys
         WHERE symmetric_key_id = 101)
   CREATE MASTER KEY ENCRYPTION
     BY PASSWORD = 'Th15i$aS7riN&ofR@nD0m!T3%t'

select top 5 * from Sales.CreditCard
/*
CreditCardID CardType      CardNumber     ExpMonth ExpYear ModifiedDate
------------ ------------- -------------- -------- ------- -----------
1            SuperiorCard   33332664695310 11       2006    2007-08-30
2            Distinguish    55552127249722 8        2005    2008-01-06
3            ColonialVoice  77778344838353 7        2005    2008-02-15
4            ColonialVoice  77774915718248 7        2006    2007-06-21
5            Vista          11114404600042 4        2005    2007-03-05
*/
```

Encriptación de datos : ejemplo

```
/*create creditCard_encrypt table changing CardNumber by CardNumber_encrypt*/
select CreditCardID, CardType,
      CardNumber_encrypt = CONVERT(varbinary(256), CardNumber),
      ExpMonth,  ExpYear,  ModifiedDate
into Sales.CreditCard_encrypt
from Sales.CreditCard
where 1=2

declare @passphrase varchar(128)
set @passphrase = 'unencrypted credit card numbers are bad, um-kay'
insert Sales.CreditCard_encrypt ( CardType, CardNumber_encrypt, ExpMonth, ExpYear, ModifiedDate )
select top 5 CardType,
      CardNumber_encrypt = EncryptByPassPhrase(@passphrase, CardNumber),
      ExpMonth,  ExpYear,  ModifiedDate from Sales.CreditCard

select * from Sales.CreditCard_encrypt
/*
CreditCardID CardType      CardNumber_encrypt    ExpMonth ExpYear ModifiedDate
------------ ------------- --------------------- -------- ------- -----------
1            SuperiorCard  0x010000007C65089E... 11       2006    2007-08-30
2            Distinguish   0x010000000C624987... 8        2005    2008-01-06
3            ColonialVoice 0x01000000AA8761A0... 7        2005    2008-02-15
4            ColonialVoice 0x010000002C2857CC... 7        2006    2007-06-21
```

Encriptación de datos : ejemplo

```
declare @passphrase varchar(128)
set @passphrase = 'unencrypted credit card numbers are bad, um-kay'
select CreditCardID,
      CardType,
      CardNumber = convert(nvarchar(25), DecryptByPassPhrase(@passphrase,
 CardNumber_encrypt)),
      ExpMonth,
      ExpYear,
      ModifiedDate
from Sales.CreditCard_encrypt
/*
CreditCardID CardType      CardNumber     ExpMonth ExpYear ModifiedDate
------------ ------------- -------------- -------- ------- -----------
1            SuperiorCard  33332664695310 11       2006    2007-08-30
2            Distinguish   55552127249722 8        2005    2008-01-06
3            ColonialVoice 77778344838353 7        2005    2008-02-15
4            ColonialVoice 77774915718248 7        2006    2007-06-21
5            Vista         11114404600042 4        2005    2007-03-05
*/
```

SQL Server Auditing

- Característica incorporada en versión 2008
- Permite auditar los accesos y acciones sobre una base de datos u objetos que contenga
  - Útil para cumplir con la ley de protección de datos
  - No actúa igual que la traza, esta supone menos coste de rendimiento.

SQL Server Auditing: ejemplo

```
/* Create the SQL Server Audit object, and send the results to the  Windows Application event log. */
USE master;
CREATE SERVER AUDIT NEW_SQL_Server_Audit
   TO APPLICATION_LOG
   WITH ( QUEUE_DELAY = 1000,  ON_FAILURE = CONTINUE);
/* Create the Database Audit Specification object using an Audit event */
USE AdventureWorks;
CREATE DATABASE AUDIT SPECIFICATION NEW_Database_Audit_Specification
FOR SERVER AUDIT NEW_SQL_Server_Audit
   ADD (SELECT ON HumanResources.Employee BY dbo)
   WITH (STATE = ON);
/* Enable the audit. */
USE master;
ALTER SERVER AUDIT NEW_SQL_Server_Audit WITH (STATE = ON);
/* Test the audit is working */
SELECT * from HumanResources.Employee;
/* Disable the audit. */
ALTER SERVER AUDIT NEW_SQL_Server_Audit WITH (STATE = OFF);
```

Inyección SQL

- Sentencias o cláusulas SQL inyectadas sobre un comando SQL existente
- Cadena inyectada se añade a la entrada de la aplicación:
  - Text boxes
  - Cadenas de consultas
  - Cadenas HTML manipuladas
- ¿Por qué funciona la inyección SQL?
  - Mala validación de aplicaciones
  - Conexión realizada en el contexto de una cuenta con privilegios elevados

Inyección SQL (Ejemplo)

CODIGO DE LA APLICACIÓN

```
var shipcity;
ShipCity = Request.form ("Shipcity")
var sql = "SELECT * FROM OrdersTable
      WHERE ShipCity = '" + Shipcity + "'";
```

USUARIO 'NORMAL'

Mete "Valencia" en el formulario

La consulta enviada al gestor es:

```
SELECT * FROM OrdersTable WHERE ShipCity =
'Valencia'
```

USUARIO MALICIOSO

Mete la siguiente sentencia en el formulario:

Valencia' ; DROP TABLE OrdersTable - -

La consulta enviada al backend es:

```
SELECT * FROM OrdersTable WHERE ShipCity =
'Valencia';
DROP TABLE OrdersTable--'
```

Buenas prácticas: Mitigación de la Inyección SQL

- No fiarse de la entrada del usuario!
  - Asegurarse del formato de entrada, y rechazar todo lo que no cumpla el formato
  - Empleo de expresiones regulares
- No componer SQL por concatenacion!
  - Usar consultas parametrizadas
- Minimizar la información en los mensajes de error, no mostrar información interesante para el desarrollador
- Principio de Mínimo Privilegio

Buenas prácticas: Autenticación

- Principio del mínimo privilegio
- Usar Autenticación Windows
- Si hay que emplear autenticación SQL:
  - Políticas de contraseña en SQL server
- Para el acceso desde aplicaciones, definir un rol de aplicación

Otras Buenas Prácticas: encriptado

- Sopesar el uso de cifrado de datos
  - Aceso a través de procedimientos almacenados
- Cifrado e Indexado
  - No cifrar columnas con índices si es posible