Authorization Database-level

Un concepto fundamental que debemos comprender es que, dentro de una base de datos son los database users –y no los logins-, los que son propietarios de objetos, y es a los database users –y no a los logins- a los que le son GRANTeados –o denegados- permisos.

Permisos sobre relaciones

- Los database permissions permiten asignar permisos de una manera *más granular* y *específica* que la que obtenemos asignando un database role.

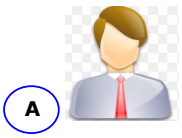
- Para que este usuario pueda leer datos, un administrador debe otorgarle al usuario el permission `SELECT`. De manera similar, si el usuario necesita agregar datos a una tabla, necesitará el permission `INSERT`. Y si necesita eliminar datos, necesitará el permission `DELETE`. Si necesita modificar información en la tabla, necesitará el permission `UPDATE`.

- Para que este usuario pueda referenciar otra relación a través de una constraint foreign key, un administrador deberá otorgarle al usuario el permission `REFERENCES`.

- Para que este usuario pueda ejecutar persistent stored modules, un administrador debe otorgarle al usuario el permission `EXECUTE`.

GRANTear (conceder) un database permission a un Database User

- El grant de un database permission a un Database User se implementa a través de la sentencia SQL `GRANT`. En el siguiente ejemplo asignamos el database permision `SELECT` sobre la tabla `Authors` al Database user `curso`:




```
GRANT SELECT
   ON Authors
   TO curso
```

B

Al usuario que otorga el privilegio (**A**) se le denomina **grantor**.

Como estamos hablando de database permissions, a continuación de `ON` (**B**) siempre tendremos un elemento (objeto) de base de datos

- Podemos hacer un `GRANT` de todos los privilegios usando la keyword `ALL`. En el siguiente ejemplo otorgamos todos los database permisions sobre la tabla `Authors` al Database user `curso`:

```
GRANT ALL PRIVILEGES
   ON Authors
   TO curso
```

- Un mismo permiso puede ser granteado a varios usuarios. En el siguiente ejemplo otorgamos el database permisions `SELECT` sobre la tabla `Authors` a los Database users `curso1 y curso2`:

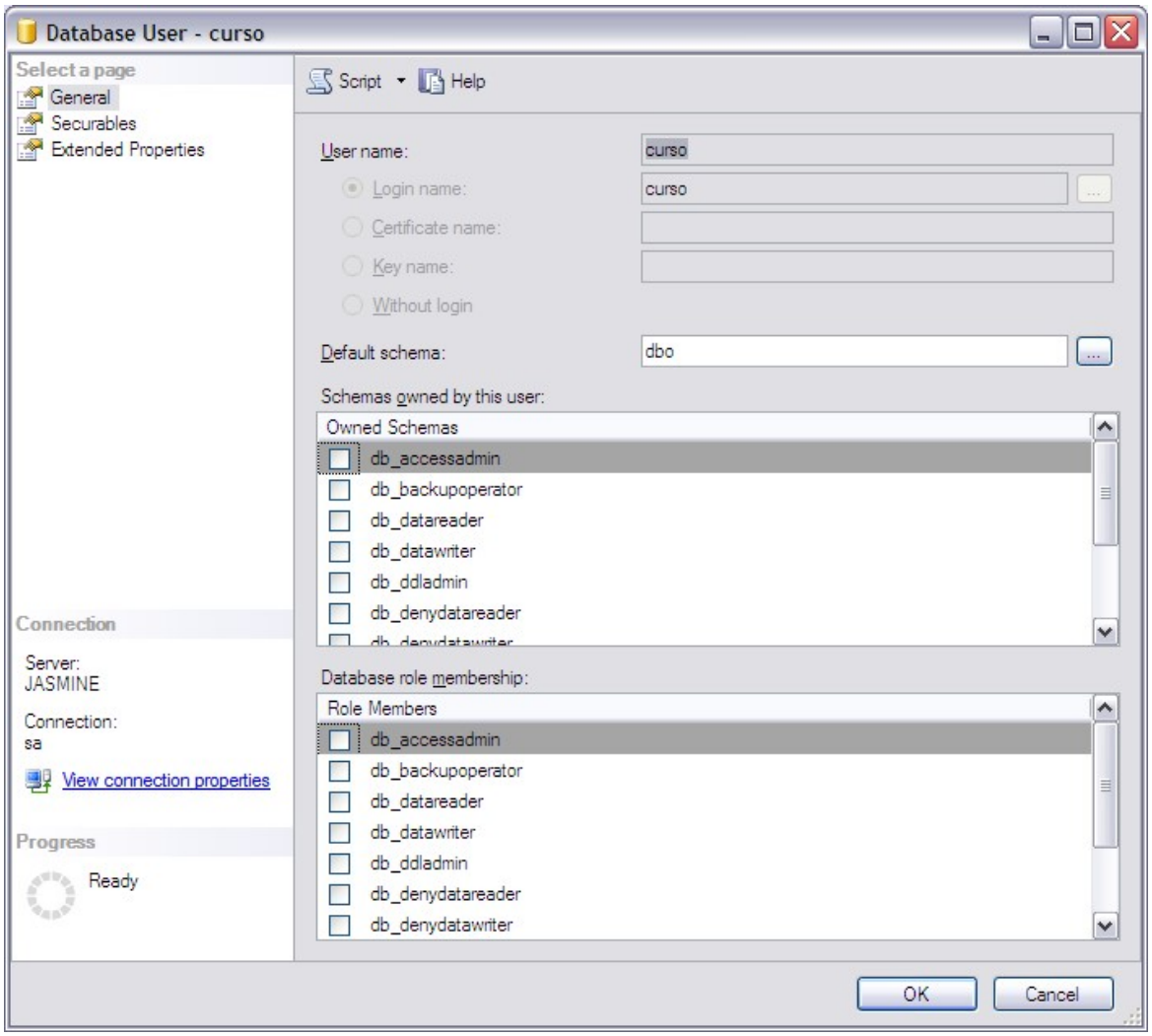
```
GRANT SELECT
   ON Authors
   TO curso1, curso2
```

- También existe la opción de otorgar un permiso a todos los database users. Lo hacemos usando el usuario genérico `PUBLIC`. En el siguiente ejemplo otorgamos el database permisions `SELECT` sobre la tabla `Authors` a todos los database users:

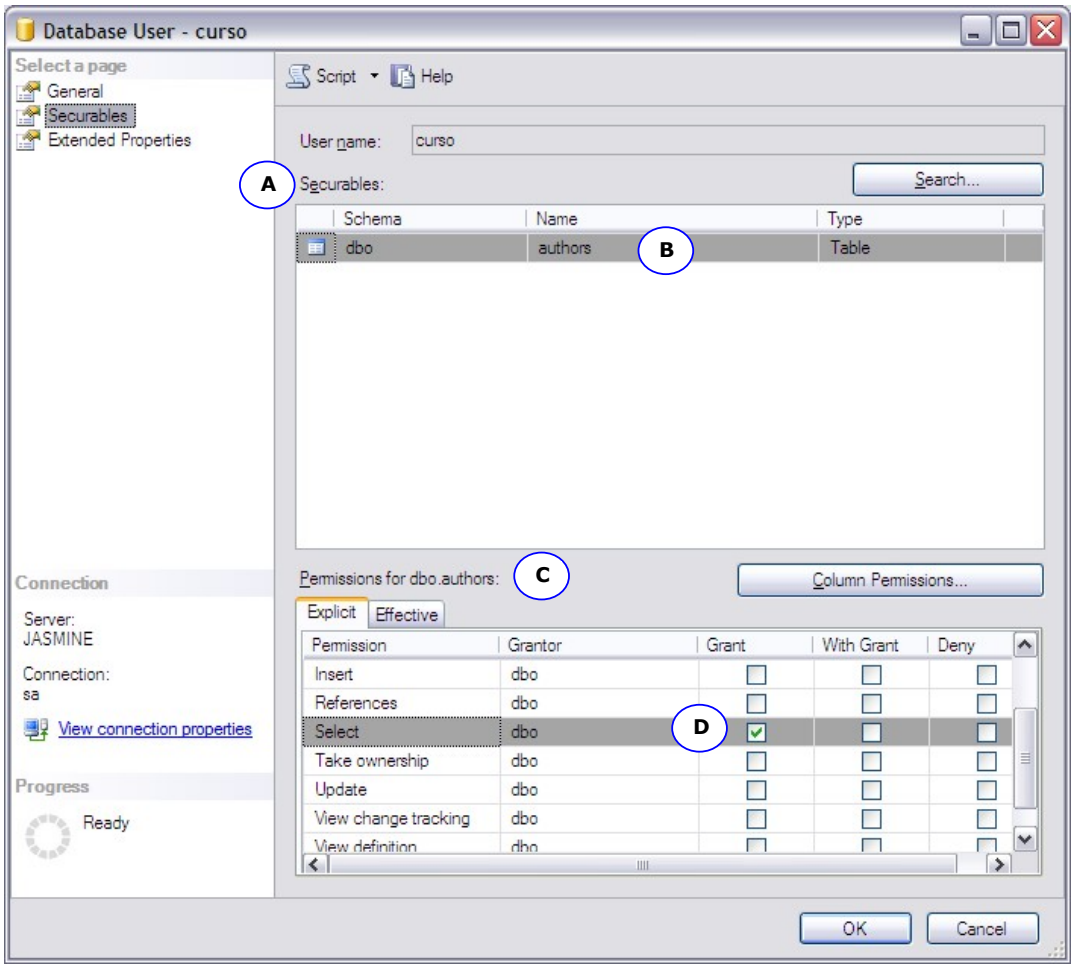
```
GRANT SELECT
   ON Authors
   TO PUBLIC
```

Averiguar que Database permissions posee un database user

1. Abrimos el menú contextual del database user –por ejemplo, `curso`- y seleccionamos Properties. Se abre la ventana *Database User - curso*:



En el área *Select a page*, seleccionamos el nodo *Securables*:



En el Area *Securables* (**A**), vemos la tabla sobre la que otorgamos el permiso (B) y debajo, en el área Permisions for `dbo.authors` (**C**), vemos el detalle de los permisos GRANTeados (**D**).

REVOKE de un database permission a un Database User

- Un **grantor** quita (o revoca) un database permission a un Database User a través de la sentencia SQL `REVOKE`. En el siguiente ejemplo revocamos el server permision `SELECT` sobre la tabla `Authors` al Database user `curso`:

```
REVOKE SELECT (A)
   ON Authors
   TO curso
```

Obviamente para que `REVOKE` tenga sentido antes debe haberse otorgado el permiso en cuestión vía `GRANT`.

La keyword `TO` puede ser reemplazada por `FROM`.

- También se puede denegar un permiso particular con la sentencia SQL `DENY`.

```
DENY SELECT
   ON Authors
   TO curso
```

- De manera similar a como ocurría con `GRANT`, podemos hacer un `REVOKE` de todos los privilegios usando la keyword `ALL`. En el siguiente ejemplo revocamos todos los database permisions sobre la tabla `Authors` al Database user `curso`:

```
REVOKE ALL PRIVILEGES
   ON Authors
   TO curso
```

Si se han concedido distintos tipos de acceso, entonces se pueden revocar algunos o todos los niveles mediante revocación selectiva.

La sintaxis es:

```
REVOKE privilegios_especificados
   FROM nombre_del_usuario;
```

Por ejemplo:

```
REVOKE ALL PRIVILEGES
   FROM user2;
```

Otorgamiento de privilegios con granularidad de columna

Los privilegios INSERT, UPDATE y REFERENCES pueden ser otorgados con granularidad de columna

Por ejemplo:

```
GRANT UPDATE (Nom_estudiante)
   ON schAlumnado.Estudiante
   TO curso;
```

```
GRANT UPDATE (price)
   ON titles
   TO curso;
```

En el caso de un GRANT de privilegio INSERT, se deben incluir en la sentencia todas las columnas NOT NULL.

¿Quienes pueden ejecutar un GRANT?

¿Quién puede ser grantor?

- El sa

- Alguien que:
  - Posea los privilegios de los que está haciendo GRANT y
  - Le haya sido concedido el poder de otorgar el privilegio del cual está haciendo GRANT

Concesión de poder para otorgar privilegios

- Un **grantor** puede transferir a otro usuario la capacidad de otorgar un permiso determinado.

- `GRANT` posee para ello la cláusula `WITH GRANT OPTION`. Esta cláusula suma al otorgamiento normal de un privilegio **la transferencia del poder de otorgar privilegios**. En el siguiente ejemplo asignamos el database permision `SELECT` sobre la tabla `Authors` al Database user `curso`, **pero además** se le otorga al usuario `curso` el poder de otorgar permisos a otros usuarios bajo su órbita:

```
GRANT SELECT
   ON Authors
   TO curso
   WITH GRANT OPTION
```

Cada privilegio posee una *grant option* asociada.

Un ejemplo

El administrador crea un usuario `adminFICH`, para que administre los sistemas de la Facultad:

sa

```
CREATE LOGIN adminFICH
   WITH PASSWORD = 'adminfich1234',
   DEFAULT_DATABASE = pubs;
```

adminFICH

`adminFICH` todavía no tiene acceso a la base de datos `pubs`. Para ello hay que crearlo como usuario de la base de datos
El administrador lo hace con la siguiente sentencia:

sa

```
USE pubs;
CREATE USER adminFICH
   FOR LOGIN adminFICH;
```

El usuario `adminFICH` ahora es usuario de la base de datos `pubs`.

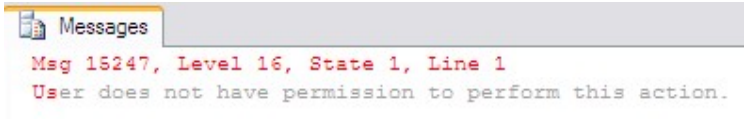
Comienza a trabajar y decide crear un usuario para administrar el sistema de Alumnado de FICH. El usuario se llamará `adminAlumnado`

adminFICH

```
CREATE LOGIN adminAlumnado
   WITH PASSWORD = 'adminalumnado1234',
   DEFAULT_DATABASE = pubs;
```

adminAlumnado

...pero no tiene este permiso, que es server-level..y obtiene:



De manera que solicita al administrador que cree el usuario `adminAlumnado`. El administrador lo crea:

sa

```
CREATE LOGIN adminAlumnado
   WITH PASSWORD = 'adminalumnado1234',
   DEFAULT_DATABASE = pubs;
```

adminAlumnado

sa

```
USE pubs;
GO
CREATE USER adminAlumnado
   FOR LOGIN adminAlumnado;
```

Ahora el administrador comienza a otorgar database permissions a adminFICH. `adminFICH` va a trabajar haciendo consultas sobre la tabla Authors, así que el administrador le otorga este permiso:

sa

```
USE pubs;

GRANT SELECT
   ON Authors
   TO adminFICH
```

adminFICH

Ahora `adminFICH` puede disparar consultas exitosamente:

adminFICH

```
SELECT *
   FROM authors
```

Results | Messages

	au_id	au_lname	au_fname	phone	address
1	172-32-1176	White	Johnson	408 496-7223	10932 Bigge Rd.
2	213-46-8915	Green	Marjorie	415 986-7020	309 63rd St. #411
3	238-95-7766	Carson	Cheryl	415 548-7723	589 Darwin Ln.
4	267-41-2394	O'Leary	Michael	408 286-2428	22 Cleveland Av. #14

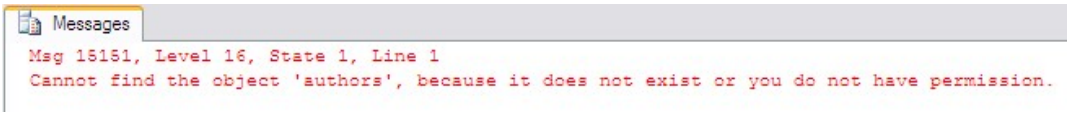
adminFICH intenta otorgar el mismo database permission a adminAlumnado, que trabajará bajo su órbita también haciendo consultas sobre la tabla Authors:

adminFICH

```
USE pubs;

GRANT SELECT
   ON Authors
   TO adminAlumnado
```

adminAlumnado

..pero obtiene un error



Le solicita al Administrador que otorgue este permiso a adminAlumnado.
El administrador, sin embargo, prefiere otorgarle a adminFICH el poder de grantear permisos de SELECT a las tablas de pubs a los usuarios bajo su órbita, y le hace un GRANT similar al anterior pero con la GRANT OPTION:

sa

```
USE pubs;

GRANT SELECT
   ON Authors
   TO adminFICH
   WITH GRANT OPTION
```

adminFICH

Ahora adminFICH sigue teniendo permiso de SELECT:

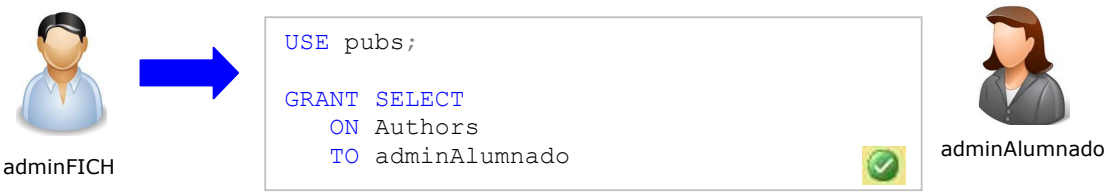
adminFICH

```
SELECT *
   FROM authors
```

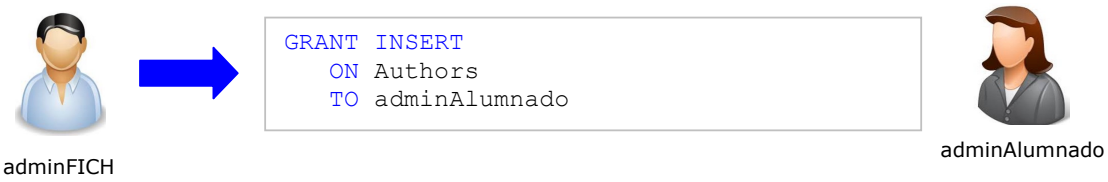
Results | Messages

	au_id	au_lname	au_fname	phone	address
1	172-32-1176	White	Johnson	408 496-7223	10932 Bigge Rd.
2	213-46-8915	Green	Marjorie	415 986-7020	309 63rd St. #411
3	238-95-7766	Carson	Cheryl	415 548-7723	589 Darwin Ln.
4	267-41-2394	O'Leary	Michael	408 286-2428	22 Cleveland Av. #14

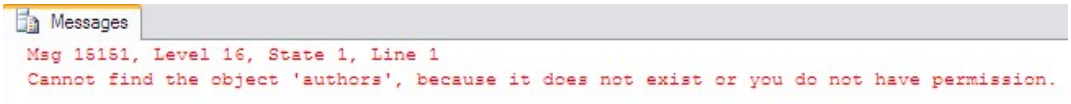
...pero además puede otorgar permission de `SELECT` sobre `Authors` a `adminAlumnado`:



Sin embargo, si `adminFICH` intenta otorgar otro database permission a `adminAlumnado`:



...obtiene un error



...porque no le fue otorgado el poder de otorgar ese database permission.

Concluyendo:

`adminFICH` posee permiso de `SELECT` sobre la tabla `authors` en `pubs`
`adminAlumnado` posee permiso de `SELECT` sobre la tabla `authors` en `pubs`

`adminFICH`, además, tiene el poder de otorgar este database permission a otros usuarios.

Cancelación de privilegios propagados

Antes habíamos dicho que si un SGBD permite la propagación de privilegios, debe proporcionar una manera de poder revocarlos de manera completa.

Veamos un ejemplo:

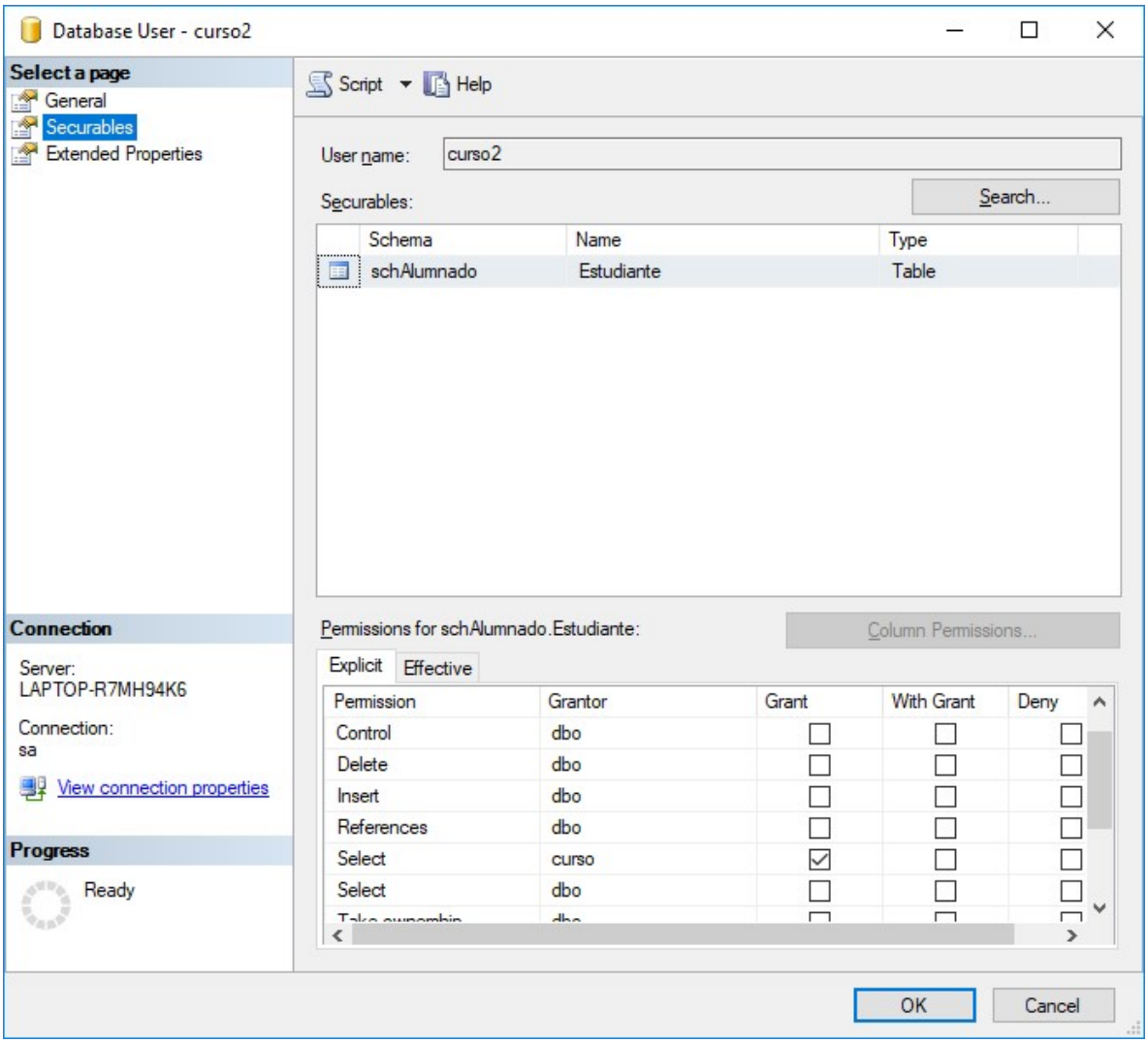
El sa ejecuta:

```
CREATE LOGIN curso2
   WITH PASSWORD = 'cursocurso',
   DEFAULT_DATABASE = Alumnado;


CREATE LOGIN curso3
   WITH PASSWORD = 'cursocurso',
   DEFAULT_DATABASE = Alumnado;


Use alumnado

CREATE USER curso2
   FOR LOGIN curso2;


CREATE USER curso3
   FOR LOGIN curso3;


GRANT SELECT
   ON schAlumnado.Estudiante
   TO curso
   WITH GRANT OPTION
```

El usuario curso ejecuta:

```
GRANT SELECT
   ON schAlumnado.Estudiante
   TO curso2
```

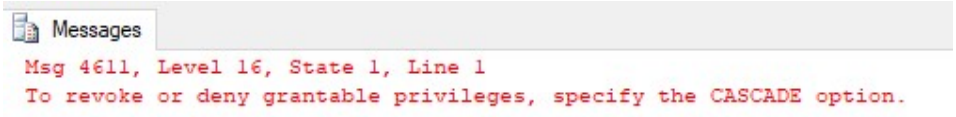
El sa observa y obtiene, para el usuario curso2:



Ahora, el sa decide revocar el permiso de SELECT a curso

```
REVOKE SELECT
   ON schAlumnado.Estudiante
   TO curso
```

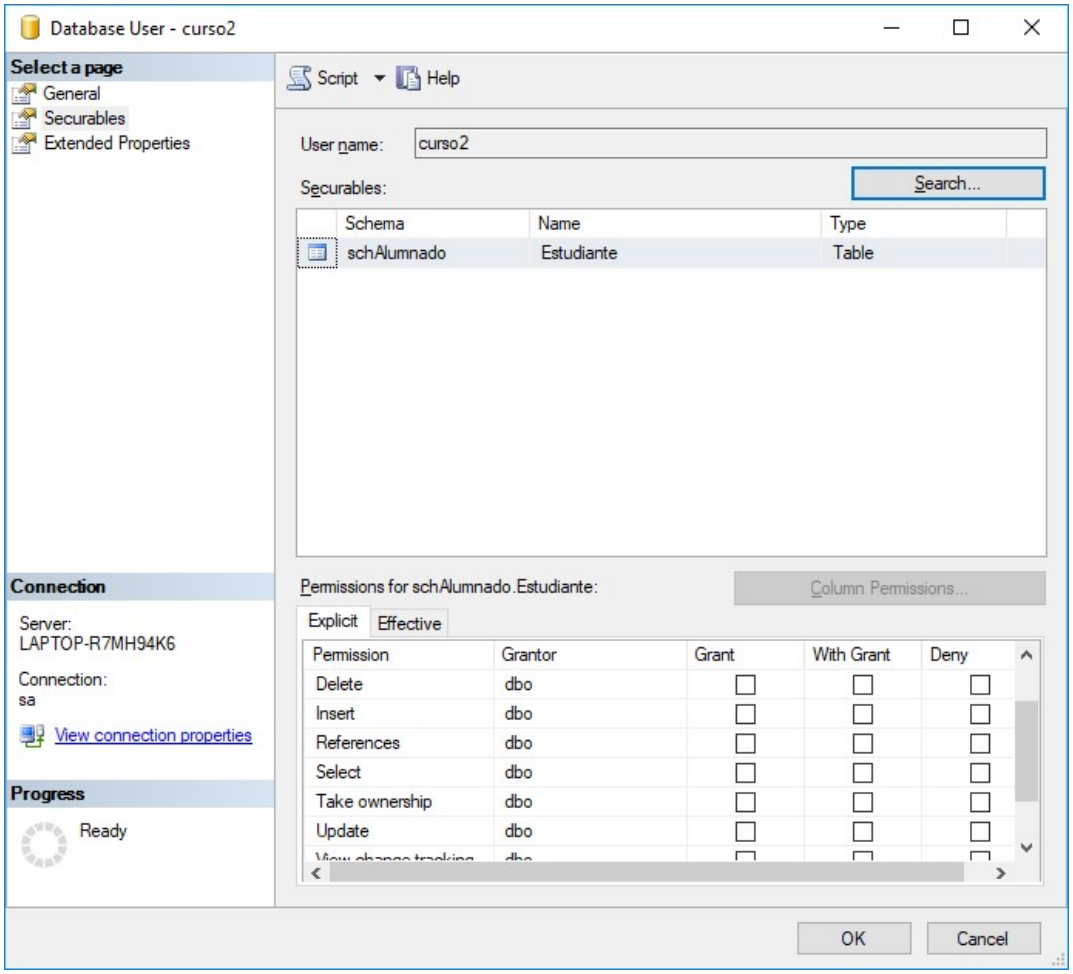
..obtiene el error:



Ahora ejecuta:

```
REVOKE SELECT
   ON schAlumnado.Estudiante
   TO curso
   CASCADE
```

..y controla el estado del usuario curso2:



...y ya ha perdido su privilegio de SELECT.