

Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática

Bases de Datos

Aspectos internos
Catálogo del sistema

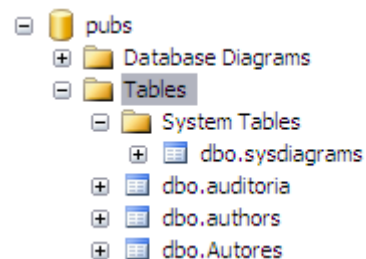
Metadata de SQL Server

System base tables

SQL Server posee un conjunto de tablas que almacenan información acerca de todos los objetos, tipos de datos, constraints, opciones de configuración y recursos del servidor de base de datos. En las últimas versiones de SQL Server estas tablas se denominan **system base tables**.

Algunas de éstas system base tables residen en la base de datos `master` y contienen **información system-wide**. Otro grupo de system base tables reside en cada base de datos existente (incluida la misma base de datos `master`), y contiene información acerca de recursos y objetos de cada base de datos particular.

A partir de SQL Server 2005, las system base tables **no son visibles**, ni en `master` ni en ninguna base de datos. No las vemos cuando expandimos el nodo `tables` en el *Object Explorer* en el *SQL Server Management Studio*:



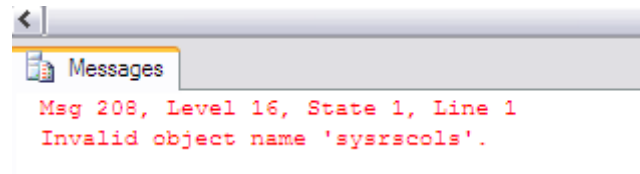
El system administrator puede usar una catalog view especial llamada `sys.objects` para obtener los **nombres** de las *system base tables*. En `pubs` por ejemplo, obtenemos cuarenta y cinco tablas:

```
Select name
FROM sys.objects
WHERE type_desc = 'SYSTEM_TABLE'
```

	name
1	sysrscols
2	sysrowsets
3	sysallocunits
4	sysfiles1
5	syspriorities
6	sysfgfrag
7	sysphfg
8	sysprufiles
9	sysftinds
10	sysowners

Pero aún siendo un administrador, no podemos disparar un `SELECT` sobre ninguna de estas tablas:

```
SELECT * FROM sysrscols
```



Esto se debe a que las system base tables son usadas para propósitos internos en el ámbito de la Database Engine y no son destinadas a uso general. Además, continuamente están sujetas a cambios, y Microsoft no garantiza que se mantenga la compatibilidad sobre ellas.

En SQL Server 2012, tres tipos de system metadata objects están destinados a uso general: las **Compatibility Views**, las **Catalog Views** y los **Dynamic Management Objects**.

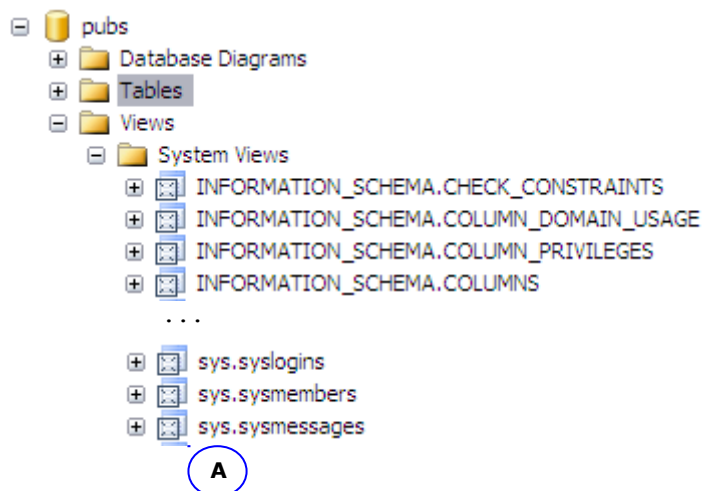
Compatibility Views

En versiones previas a SQL Server 2005 se podía acceder directamente a las entonces llamadas *system tables*. A partir de SQL Server 2005 esto no es posible.

Como muchos desarrolladores poseen código antiguo basado en esos nombres de tabla y de columnas, Microsoft proporciona actualmente una serie de *compatibility views*, que son accesibles desde cualquier base de datos y justamente poseen los mismos nombres y estructura que las *system tables* de SQL Server 2000.

Las compatibility views son proporcionadas solo para compatibilidad hacia atrás. Los desarrollos actuales deberían utilizar otros mecanismos de acceso a metadata, tales como las catalog views. Además, las compatibility views serán eliminadas en futuras versiones de SQL Server.

Las Compatibility Views residen en el schema `sys` (A):



Por ejemplo, en master, `sys.sysdatabases` es una **compatibility view**:

```
Select * from sys.sysdatabases
```

	name	dbid	sid	mode	status	status2	crdate	reserved
1	master	1	0x01	0	65544	1090520064	2003-04-08 09:13:36.390	1900-01-01
2	tempdb	2	0x01	0	65544	1090520064	2015-10-03 19:12:18.437	1900-01-01
3	model	3	0x01	0	65536	1090519040	2003-04-08 09:13:36.390	1900-01-01
4	msdb	4	0x01	0	65544	1627390976	2008-07-09 16:46:27.767	1900-01-01
5	pubs	5	0x01	0	65544	1627389952	2014-07-07 08:29:42.857	1900-01-01

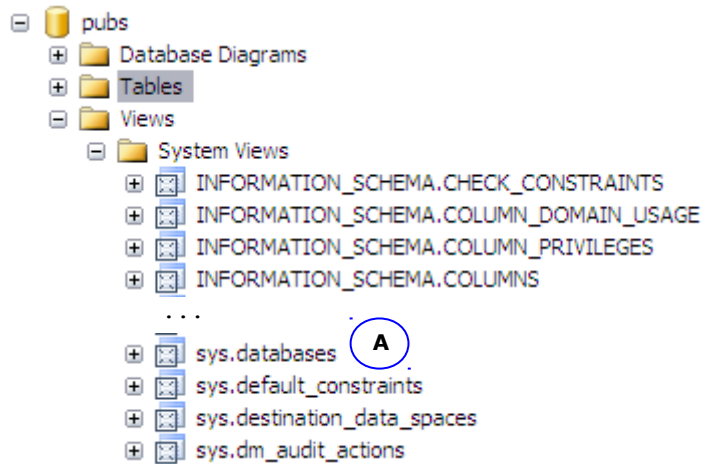
Catalog Views

SQL Server 2005 introdujo una serie de catalog views como una interface general a la metadata del sistema.

Muchos nombres son similares a los de las system tables de SQL Server 2000.

Al igual que en SQL Server 2000, la parte “**system-wide**” de la metadata (por ejemplo, bases de datos, logins, etc.) está disponible sólo en la base de datos `master`, mientras que la metadata particular de cada base de datos (por ejemplo, objetos y permisos) está disponible en cada base de datos existente (incluida la misma base de datos `master`).

Las Catalog Views también residen en el schema `sys` (A):



Por ejemplo, en `master`, `sys.databases` es una **catalog view**:

```
Select * from sys.databases
```

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc
1	sysrscols	3	NULL	4	0	S	SYSTEM_TABLE
2	sysrowsets	5	NULL	4	0	S	SYSTEM_TABLE
3	sysallocunits	7	NULL	4	0	S	SYSTEM_TABLE
4	sysfiles1	8	NULL	4	0	S	SYSTEM_TABLE
5	syspriorities	17	NULL	4	0	S	SYSTEM_TABLE
6	sysdbfrag	18	NULL	4	0	S	SYSTEM_TABLE
7	sysfgfrag	19	NULL	4	0	S	SYSTEM_TABLE

Las catalog views son la interface recomendada para acceder al información del catálogo en SQL Server 2012 y a futuro.

Columnas

Como vemos en nuestro caso de ejemplo, las columnas de la catalog view `sys.databases` difieren de las columnas de la compatibility view `sys.sysdatabases`.

Las catalog views han sido diseñadas cuidadosamente usando un enfoque de "herencia". Por ejemplo, `sys.objects` posee los atributos comunes a todos los tipos de objetos. `sys.tables` y `sys.views` "heredan" los mismos atributos que `sys.objects`, y agregan aquellos que son relevantes solo a su tipo de objeto particular.

Por ejemplo, si ejecutamos la siguiente sentencia `SELECT`:

```
USE pubs
SELECT * FROM sys.objects
```

obtenemos doce columnas: `name`, `object_id`, `principal_id`, `schema_id`, `parent_object_id`, `type`, `type_desc`, `create_date`, `modify_date`, `is_ms_shipped`, `is_published` y `is_schema_published`.

Y, si ejecutamos la siguiente:

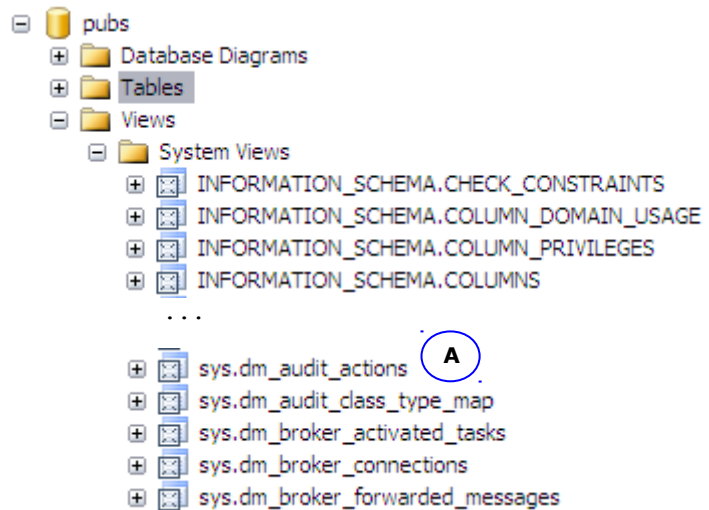
```
SELECT * FROM sys.tables
```

obtenemos las mismas doce columnas más quince columnas más que aplican sólo al tipo de objeto tabla.

Dynamic Management Objects

Los Dynamic Management Objects (también denominados **Dynamic Management Views - DMV**) son views y functions que permiten a los desarrolladores y administradores de base de datos examinar el **comportamiento** interno de SQL Server.

Las DMVs también residen en el schema `sys` y se identifican porque su nombre comienza con el prefijo `dm_` (**A**):



Por ejemplo, en `pubs`, `sys.dm_tran_locks` es una **DMV**:

```
USE pubs
SELECT * FROM sys.dm_tran_locks
```

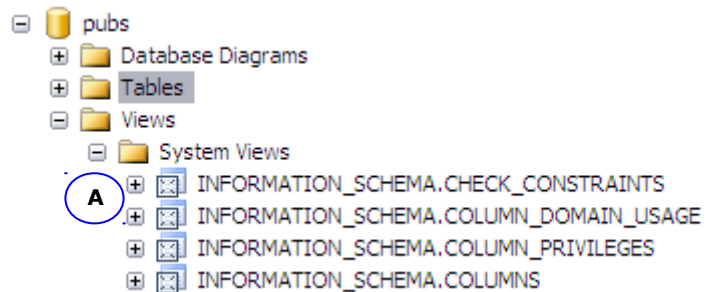
	resource_type	resource_subtype	resource_database_id	resource_description	resource_associated_entity_id
1	DATABASE		5		0

Las DMVs se agrupan en varias categorías en base al área funcional de la información que exponen. Las categorías se especifican a través de códigos que se indican a continuación del prefijo `dm_`. En el ejemplo, el código `tran` indica que estamos viendo detalles acerca de las transacciones. Otras códigos son `exec` (conexiones y ejecución de código de usuario), `os` (información del sistema de bajo nivel, como manejo de memoria), `io` (actividad de entrada salida en la red y en disco) y `db` (detalle acerca de las bases de datos e índices).

Information Schema Views

Las Information Schema Views son la forma de acceder al catálogo del sistema definida por el estándar ANSI SQL-92.

En SQL Server, estas views residen en el schema `INFORMATION_SCHEMA` (A):



Por ejemplo, en `pubs`, `INFORMATION_SCHEMA.tables` es una **Information Schema View**:

```
USE pubs
SELECT * FROM INFORMATION_SCHEMA.tables
```

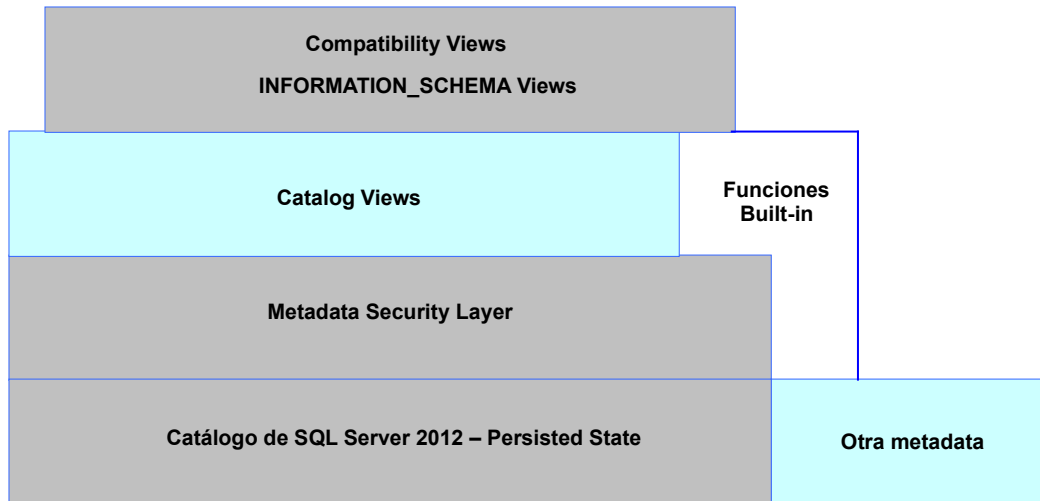
	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE
1	pubs	dbo	publishers	BASE TABLE
2	pubs	dbo	VueloAsiento	BASE TABLE
3	pubs	dbo	titles	BASE TABLE
4	pubs	dbo	titleauthor	BASE TABLE
5	pubs	dbo	tblAnios	BASE TABLE
6	pubs	dbo	Cantidad	BASE TABLE
7	pubs	dbo	VentasPorAnio	BASE TABLE
8	pubs	dbo	stores	BASE TABLE
9	pubs	dbo	sales	BASE TABLE

Las Information Schema Views son la interface recomendada si necesitamos escribir código que acceda a metadata y a la vez sea **portable**.

Como contrapartida, sólo muestran información sobre objetos compatibles con el estándar SQL-92. Por ejemplo, no podremos obtener información sobre índices u otros objetos específicos a SQL Server.

Layers de metadata

El siguiente gráfico muestra las diferentes layers de metadata en SQL Server 2012:



1. Metadata a nivel de instancia de SQL Server (Server level)

1.1. Bases de datos

Obtenemos información acerca de las bases de datos creadas en la instancia a través de la catalog view `databases` del schema `sys` de la base de datos `master`:

```
SELECT name, database_id, create_date, snapshot_isolation_state_desc,  
       is_read_committed_snapshot_on  
FROM master.sys.databases
```

	name	database_id	create_date	snapshot_isolation_state_desc	is_read_committed_snapshot_on
1	master	1	2003-04-08 09:13:36.390	ON	0
2	tempdb	2	2015-10-22 11:03:24.857	OFF	0
3	model	3	2003-04-08 09:13:36.390	OFF	0
4	msdb	4	2008-07-09 16:46:27.767	ON	0
5	pubs	5	2015-10-21 12:34:03.077	OFF	0

1.2. Logins

Obtenemos información acerca de los logins creados en la instancia a través de la catalog view `server_principals` del schema `sys` de la base de datos `master`:

```
SELECT name, principal_id, type, type_desc, default_database_name
FROM master.sys.server_principals
```

name	principal_id	type	type_desc	default_database_name
sa	1	S	SQL_LOGIN	master
public	2	R	SERVER_ROLE	NULL
sysadmin	3	R	SERVER_ROLE	NULL
securityadmin	4	R	SERVER_ROLE	NULL
serveradmin	5	R	SERVER_ROLE	NULL
setupadmin	6	R	SERVER_ROLE	NULL
processadmin	7	R	SERVER_ROLE	NULL
diskadmin	8	R	SERVER_ROLE	NULL
dbcreator	9	R	SERVER_ROLE	NULL
bulkadmin	10	R	SERVER_ROLE	NULL
##MS_SQLResourceSigningCertificate##	101	C	CERTIFICATE_M...	master
##MS_SQLReplicationSigningCertificate##	102	C	CERTIFICATE_M...	master
##MS_SQLAuthenticatorCertificate##	103	C	CERTIFICATE_M...	master
##MS_PolicySigningCertificate##	105	C	CERTIFICATE_M...	master
##MS_PolicyEventProcessingLogin##	256	S	SQL_LOGIN	master
##MS_PolicyTsqlExecutionLogin##	257	S	SQL_LOGIN	master
##MS_AgentSigningCertificate##	258	C	CERTIFICATE_M...	master
NT AUTHORITY\SYSTEM	259	U	WINDOWS_LOGIN	master
JASMINE\LPD	260	U	WINDOWS_LOGIN	master
JASMINE\bdatos2015	261	U	WINDOWS_LOGIN	pubs
curso	264	S	SQL_LOGIN	pubs
sololocal	266	S	SQL_LOGIN	pubs

`name` es el nombre del principal, único dentro de la instancia de SQL Server.

`principal_id` es el ID del principal también único dentro de la instancia de SQL Server.

`type` es el tipo de principal.

SQL Server posee dos modos de autenticación: **SQL Server Authentication** y **Windows Authentication**. Una **S** en la columna `type` indica que el principal es un **SQL login** (que usa **SQL Server Authentication**), mientras que una **U** en la columna `type` indica que se trata de un **Windows login** (que usa **Windows Authentication**)

Otros types son Server Role (**R**) y Login mapeado a certificate (**C**).

Los Login IDs poseen una base de datos por omisión. La misma es la que aparece en la columna `default_database_name`.

1.3. Mensajes de error

Obtenemos información acerca de los mensajes de error almacenados en la instancia a través de la catalog view `messages` del schema `sys` de la base de datos `master`. La columna `language_id` especifica el idioma del mensaje. El valor para inglés es `'1033'` y para español `'3082'`:

```
SELECT message_id, language_id, severity, text
FROM master.sys.messages
WHERE language_id = '3082'
ORDER BY severity, message_id
```

message_id	language_id	severity	text
13008	3082	0	recibir
13010	3082	0	leer
13012	3082	0	una instrucción de base de datos USE
13013	3082	0	un procedimiento, una función o un desencadenador
13014	3082	0	vistas indizadas y/o índices en columnas calculad...
13015	3082	0	operaciones de índice espacial
13016	3082	0	una cláusula INTO
13018	3082	0	una cláusula COMPUTE
13019	3082	0	una instrucción SELECT INTO
13020	3082	0	opción
13021	3082	0	opción de desplazamiento
13022	3082	0	opción de estadísticas
13024	3082	0	nombre de función

El valor de la columna `message_id` corresponde al **Number** del Error (Ver *Guía de Trabajo Nro. 6 – Tratamiento de errores, Sección 1.2.2. Componentes de un error*). El valor de la columna `severity` corresponde a la **Severity** del error y el valor de la columna `text` corresponde al **Message** del error.

2. Metadata a nivel de base de datos

2.1. Users

Obtenemos información acerca de los usuarios de una base de datos a través de la catalog view `database_principals` del schema `sys` de cualquier base de datos:

```
USE pubs
SELECT name, principal_id, type, type_desc, default_schema_name
FROM sys.database_principals
```

	name	principal_id	type	type_desc	default_schema_name
1	public	0	R	DATABASE_ROLE	NULL
2	dbo	1	S	SQL_USER	dbo
3	guest	2	S	SQL_USER	guest
4	INFORMATION_SCHEMA	3	S	SQL_USER	NULL
5	sys	4	S	SQL_USER	NULL
6	db_owner	16384	R	DATABASE_ROLE	NULL
7	db_accessadmin	16385	R	DATABASE_ROLE	NULL
8	db_securityadmin	16386	R	DATABASE_ROLE	NULL
9	db_ddladmin	16387	R	DATABASE_ROLE	NULL
10	db_backupoperator	16389	R	DATABASE_ROLE	NULL
11	db_datareader	16390	R	DATABASE_ROLE	NULL
12	db_datawriter	16391	R	DATABASE_ROLE	NULL
13	db_denydatareader	16392	R	DATABASE_ROLE	NULL
14	db_denydatawriter	16393	R	DATABASE_ROLE	NULL

`type` es el tipo de principal. Son similares a los que aparecen en la catalog view `sys.server_principals`. Una **S** en la columna `type` indica que el principal es un **SQL user** (que usa **SQL Server Authentication**), mientras que una **R** indica que se trata de un Database Role.

La siguiente consulta utiliza la columna `sid` (globally unique **security identifier**) para vincular los usuarios de base de datos con su Login server-level correspondiente:

```
SELECT dp.name 'User',
       dp.principal_id,
       dp.type, dp.type_desc 'Tipo de usuario',
       dp.default_schema_name,
       sp.name 'Login',
       sp.type_desc 'Tipo de Login'
FROM sys.database_principals dp LEFT JOIN sys.server_principals sp
     ON dp.sid = sp.sid
```

User	principal_id	type	Tipo de usuario	default_schema_name	Login	Tipo de Login
public	0	R	DATABASE_ROLE	NULL	NULL	NULL
dbo	1	S	SQL_USER	dbo	sa	SQL_LOGIN
guest	2	S	SQL_USER	guest	NULL	NULL
INFORMATION_SCHEMA	3	S	SQL_USER	NULL	NULL	NULL
sys	4	S	SQL_USER	NULL	NULL	NULL
db_owner	16384	R	DATABASE_ROLE	NULL	NULL	NULL
db_accessadmin	16385	R	DATABASE_ROLE	NULL	NULL	NULL
db_securityadmin	16386	R	DATABASE_ROLE	NULL	NULL	NULL
db_ddladmin	16387	R	DATABASE_ROLE	NULL	NULL	NULL
db_backupoperator	16389	R	DATABASE_ROLE	NULL	NULL	NULL
db_datareader	16390	R	DATABASE_ROLE	NULL	NULL	NULL
db_datawriter	16391	R	DATABASE_ROLE	NULL	NULL	NULL
db_denydatareader	16392	R	DATABASE_ROLE	NULL	NULL	NULL
db_denydatawriter	16393	R	DATABASE_ROLE	NULL	NULL	NULL

Como vemos, un *database role* cualquiera, por ejemplo `db_owner` (**A**) no está asociado a ningún login (**B**), ya que su propósito es solamente agrupar usuarios a los fines de simplificar la asignación de permisos a nivel de base de datos.

2.2. Objetos

Obtenemos información acerca de los objetos de una base de datos a través de la catalog view `objects` del schema `sys` de cualquier base de datos:

```
USE pubs
SELECT type,
       type_desc,
       name,
       object_id,
       schema_id,
       parent_object_id,
       OBJECT_NAME(parent_object_id) 'Objeto Padre'
FROM sys.objects
ORDER BY type, name
```

type	type_desc	name	object_id	schema_id	parent_object_id	Objeto Padre
P	SQL_STORED_PROCE...	usp_GetErrorInfo	869578136	1	0	NULL
PK	PRIMARY_KEY_CONS...	PK_jobs__6E32B6A51A14E395	453576654	1	405576483	jobs
PK	PRIMARY_KEY_CONS...	PK_emp_id	581577110	1	565577053	employee
PK	PRIMARY_KEY_CONS...	UPKCL_storeid	261575970	1	245575913	stores
PK	PRIMARY_KEY_CONS...	UPKCL_auidind	2121058592	1	2105058535	authors
PK	PRIMARY_KEY_CONS...	UPKCL_pubind	53575229	1	37575172	publishers
PK	PRIMARY_KEY_CONS...	UPKCL_pubinfo	533576939	1	517576882	pub_info
PK	PRIMARY_KEY_CONS...	UPKCL_sales	293576084	1	277576027	sales
PK	PRIMARY_KEY_CONS...	UPKCL_taind	197575742	1	181575685	titleauthor
PK	PRIMARY_KEY_CONS...	UPKCL_titleidind	117575457	1	101575400	titles
S	SYSTEM_TABLE	sysallocunits	7	4	0	NULL
S	SYSTEM_TABLE	sysasymkeys	95	4	0	NULL

La columna `type` y su descripción asociada, la columna `type_desc` determinan los diferentes tipos de objetos. Por ejemplo, tenemos **constraints default** (type `'D'`), **constraints CHECK** (type `'C'`), **constraints PRIMARY KEY**, **FOREIGN KEY** y **UNIQUE** (type `'PK'`, `'FK'` y `'UK'`), **tablas de usuario** (type `'U'`), **tablas del sistema** (type `'S'`), **views** (type `'V'`), **procedimientos almacenados** (type `'P'`), **triggers** (type `'TR'`) y **sequence Objects** (type `'SO'`) por solo mencionar algunos.

Como la catalog view `sys.objects` nos permite obtener información de varios tipos de objetos, volveremos sobre ella cada vez que analicemos un tipo de objeto en particular.

2.3. Schemas

Podemos obtener los schemas definidos en una base de datos consultando la catalog view `schemas` del schema `sys` de la base de datos:

```
USE pubs
SELECT schema_id, principal_id, name
FROM sys.schemas
```

schema_id	principal_id	name
1	1	dbo
2	2	guest
3	3	INFORMATION_SCHEMA
4	4	sys
16384	16384	db_owner
16385	16385	db_accessadmin
16386	16386	db_securityadmin
16387	16387	db_ddladmin
16389	16389	db_backupoperator
16390	16390	db_datareader
16391	16391	db_datawriter
16392	16392	db_denydatareader
16393	16393	db_denydatawriter

2.4. Tablas de usuario

Podemos obtener las tablas de usuario definidas en una base de datos consultando la catalog view `tables` del schema `sys` de la base de datos:

```
SELECT t.schema_id,  
       s.name Schema1,  
       type,  
       type_desc,  
       t.name Tabla,  
       object_id  
FROM sys.tables t INNER JOIN sys.schemas s  
      ON t.schema_id = s.schema_id
```

schema_id	Schema1	type	type_desc	Tabla	object_id
1	dbo	U	USER_TABLE	publishers	37575172
1	dbo	U	USER_TABLE	titles	101575400
1	dbo	U	USER_TABLE	titleauthor	181575685
1	dbo	U	USER_TABLE	stores	245575913
1	dbo	U	USER_TABLE	sales	277576027
1	dbo	U	USER_TABLE	roysched	341576255
1	dbo	U	USER_TABLE	discounts	373576369
1	dbo	U	USER_TABLE	jobs	405576483
1	dbo	U	USER_TABLE	pub_info	517576882
1	dbo	U	USER_TABLE	employee	565577053
1	dbo	U	USER_TABLE	Setup	885578193
1	dbo	U	USER_TABLE	AutoresBadSeller	901578250
1	dbo	U	USER_TABLE	authors	2105058535

Obtenemos un resultado similar filtrando la catalog view `sys.objects` por un `type = 'U'`:

```
SELECT t.schema_id,
       s.name Schema1,
       type,
       type_desc,
       t.name Tabla,
       object_id
FROM sys.objects t INNER JOIN sys.schemas s
                  ON t.schema_id = s.schema_id
WHERE t.type = 'U'
```

schema_id	Schema1	type	type_desc	Tabla	object_id
1	dbo	U	USER_TABLE	publishers	37575172
1	dbo	U	USER_TABLE	titles	101575400
1	dbo	U	USER_TABLE	titleauthor	181575685
1	dbo	U	USER_TABLE	stores	245575913
1	dbo	U	USER_TABLE	sales	277576027
1	dbo	U	USER_TABLE	roysched	341576255
1	dbo	U	USER_TABLE	discounts	373576369
1	dbo	U	USER_TABLE	jobs	405576483
1	dbo	U	USER_TABLE	pub_info	517576882
1	dbo	U	USER_TABLE	employee	565577053
1	dbo	U	USER_TABLE	Setup	885578193
1	dbo	U	USER_TABLE	Autores...	901578250
1	dbo	U	USER_TABLE	authors	2105058...



Tablas de usuario

También podemos acceder a información similar a través de la Information Schema View `INFORMATION_SCHEMA.TABLES`:

```
SELECT *  
FROM INFORMATION_SCHEMA.TABLES  
WHERE TABLE_TYPE = 'BASE TABLE'  
ORDER BY TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
```

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE
pubs	dbo	authors	BASE TABLE
pubs	dbo	AutoresBadSeller	BASE TABLE
pubs	dbo	discounts	BASE TABLE
pubs	dbo	employee	BASE TABLE
pubs	dbo	jobs	BASE TABLE
pubs	dbo	pub_info	BASE TABLE
pubs	dbo	publishers	BASE TABLE
pubs	dbo	roysched	BASE TABLE
pubs	dbo	sales	BASE TABLE
pubs	dbo	Setup	BASE TABLE
pubs	dbo	stores	BASE TABLE
pubs	dbo	titleauthor	BASE TABLE
pubs	dbo	titles	BASE TABLE

2.5. Tablas del sistema

Podemos obtener las tablas del sistema filtrando la catalog view `sys.objects` por un `type = 'S'`:

```
USE pubs
SELECT t.schema_id,
       s.name Schema1,
       type,
       type_desc,
       t.name Tabla,
       object_id
FROM sys.objects t INNER JOIN sys.schemas s
                  ON t.schema_id = s.schema_id
WHERE t.type = 'S'
```

schema_id	Schema1	type	type_desc	Tabla	object_id
4	sys	S	SYSTEM_TABLE	sysrscols	3
4	sys	S	SYSTEM_TABLE	sysrowsets	5
4	sys	S	SYSTEM_TABLE	sysallocunits	7
4	sys	S	SYSTEM_TABLE	sysfiles1	8
4	sys	S	SYSTEM_TABLE	syspriorities	17
4	sys	S	SYSTEM_TABLE	sysfgfrag	19
4	sys	S	SYSTEM_TABLE	sysphfg	23
4	sys	S	SYSTEM_TABLE	sysprfiles	24
4	sys	S	SYSTEM_TABLE	sysftinds	25
4	sys	S	SYSTEM_TABLE	sysowners	27
4	sys	S	SYSTEM_TABLE	sysprivs	29
4	sys	S	SYSTEM_TABLE	syschobjs	34
4	sys	S	SYSTEM_TABLE	syscolpars	41
4	sys	S	SYSTEM_TABLE	sysnsobjs	44
4	sys	S	SYSTEM_TABLE	syscerts	46

2.6. Constraints

Como vimos, podemos filtrar la catalog view `sys.objects` a fin de obtener diferentes tipos de constraints:

- constraints **default** (type '**D**')
- constraints **CHECK** (type '**C**')
- constraints **PRIMARY KEY** (type '**PK**')
- constraints **FOREIGN KEY** (type '**FK**')
- constraints **UNIQUE** (type '**FK**')



Constraints

En cuanto a las constraints, podemos obtenerlas través de la Information Schema View `INFORMATION_SCHEMA.TABLE_CONSTRAINTS`:

```
USE pubs
SELECT * FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
```

	CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE
1	pubs	dbo	CK_authors__zip_014935CB	pubs	dbo	authors	CHECK
2	pubs	dbo	UPKCL_pubind	pubs	dbo	publishers	PRIMARY KEY
3	pubs	dbo	CK_publisher__pub_j_0425A276	pubs	dbo	publishers	CHECK
4	pubs	dbo	UPKCL_titledind	pubs	dbo	titles	PRIMARY KEY
5	pubs	dbo	FK_titles__pub_id_08EA5793	pubs	dbo	titles	FOREIGN KEY
6	pubs	dbo	UPKCL_taind	pubs	dbo	titleauthor	PRIMARY KEY
7	pubs	dbo	FK_titleauth__au_id_0CBAE877	pubs	dbo	titleauthor	FOREIGN KEY
8	pubs	dbo	FK_titleauth__title_0DAF0CB0	pubs	dbo	titleauthor	FOREIGN KEY
9	pubs	dbo	UPKCL_storeid	pubs	dbo	stores	PRIMARY KEY
10	pubs	dbo	UPKCL_sales	pubs	dbo	sales	PRIMARY KEY
11	pubs	dbo	FK_sales__stor_id_1273C1CD	pubs	dbo	sales	FOREIGN KEY
12	pubs	dbo	FK_sales__title_id_1367E606	pubs	dbo	sales	FOREIGN KEY
13	pubs	dbo	FK_roysched__title_15502E78	pubs	dbo	roysched	FOREIGN KEY

Podemos obtener solo las constraints FOREIGN KEY a través de la Information Schema View

`INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS`:

```
SELECT * FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS
```

...y solo las constraints CHECK a través de la Information Schema View

`INFORMATION_SCHEMA.CHECK_CONSTRAINTS`:

```
SELECT * FROM INFORMATION_SCHEMA.CHECK_CONSTRAINTS
```

2.7. Views

Como vimos, podemos filtrar la catalog view `sys.objects` a fin de obtener las views definidas (type `'V'`).

Dado una view, la catalog view `sys.sql_modules` nos proporciona más información, incluido su definición:

```
SELECT sys.sql_modules.object_id, definition
FROM sys.sql_modules INNER JOIN sys.objects
    ON sys.sql_modules.object_id = sys.objects.object_id
WHERE sys.objects.type = 'V'
```

object_id	definition
725577623	CREATE VIEW titleview AS select title, au_ord, au_lname, price...



Views

También podemos obtener información sobre Views usando la Information Schema View `INFORMATION_SCHEMA.VIEWS`:

```
USE pubs
SELECT * FROM INFORMATION_SCHEMA.VIEWS
```

La columna `VIEW_DEFINITION` proporciona el código de creación de la View.

2.8. Stored procedures

Podemos obtener los procedimientos almacenados definidos en una base de datos consultando la catalog view `procedures` del schema `sys` de la base de datos:

```
USE pubs
SELECT p.schema_id,
       s.name Schema1,
       type,
       type_desc,
       p.name Tabla,
       object_id
FROM sys.procedures p INNER JOIN sys.schemas s
ON s.schema_id = p.schema_id
```

schema_id	Schema1	type	type_desc	SP	object_id
1	dbo	P	SQL_STORED_PROCEDURE	byroyalty	741577680
1	dbo	P	SQL_STORED_PROCEDURE	reptq1	757577737
1	dbo	P	SQL_STORED_PROCEDURE	reptq2	773577794
1	dbo	P	SQL_STORED_PROCEDURE	reptq3	789577851
1	dbo	P	SQL_STORED_PROCEDURE	ObtenerPublicacionesDeUnAutor	805577908
1	dbo	P	SQL_STORED_PROCEDURE	ObtenerCantVendidaDeCadaP...	821577965
1	dbo	P	SQL_STORED_PROCEDURE	ObtenerVentasDeUnaPublicaci...	837578022
1	dbo	P	SQL_STORED_PROCEDURE	ObtenerAutoresDeUnaPublicac...	853578079
1	dbo	P	SQL_STORED_PROCEDURE	usp_GetErrorInfo	869578136
1	dbo	P	SQL_STORED_PROCEDURE	ObtenerID	917578307
1	dbo	P	SQL_STORED_PROCEDURE	EliminarPublicacion	949578421
1	dbo	P	SQL_STORED_PROCEDURE	EliminarAutor	965578478

Además, como vimos, podemos filtrar la catalog view `sys.objects` a fin de obtener los stored procedures definidos (type 'P').

```
USE pubs
SELECT t.schema_id,
       s.name Schema1,
       type,
       type_desc,
       t.name SP,
       object_id
FROM sys.objects t INNER JOIN sys.schemas s
                  ON t.schema_id = s.schema_id
WHERE t.type = 'P'
```

schema_id	Schema1	type	type_desc	SP	object_id
1	dbo	P	SQL_STORED_PROCEDURE	byroyalty	741577680
1	dbo	P	SQL_STORED_PROCEDURE	reptq1	757577737
1	dbo	P	SQL_STORED_PROCEDURE	reptq2	773577794
1	dbo	P	SQL_STORED_PROCEDURE	reptq3	789577851
1	dbo	P	SQL_STORED_PROCEDURE	ObtenerPublicacionesDeUnAutor	805577908
1	dbo	P	SQL_STORED_PROCEDURE	ObtenerCantVendidaDeCadaPublicacion	821577965
1	dbo	P	SQL_STORED_PROCEDURE	ObtenerVentasDeUnaPublicacion	837578022
1	dbo	P	SQL_STORED_PROCEDURE	ObtenerAutoresDeUnaPublicacion	853578079
1	dbo	P	SQL_STORED_PROCEDURE	usp_GetErrorInfo	869578136
1	dbo	P	SQL_STORED_PROCEDURE	ObtenerID	917578307
1	dbo	P	SQL_STORED_PROCEDURE	EliminarPublicacion	949578421
1	dbo	P	SQL_STORED_PROCEDURE	EliminarAutor	965578478

Dado un SP, la catalog view `sys.sql_modules` nos proporciona más información, incluido su código fuente:

```
SELECT sys.sql_modules.object_id, definition
FROM sys.sql_modules INNER JOIN sys.objects
ON sys.sql_modules.object_id = sys.objects.object_id
WHERE sys.objects.type = 'P'
```

object_id	definition
741577680	CREATE PROCEDURE byroyalty @percentage int AS ...
757577737	CREATE PROCEDURE reptq1 AS select case when ...
773577794	CREATE PROCEDURE reptq2 AS select case when ...
789577851	CREATE PROCEDURE reptq3 @lolimit money, @hilimit ...
805577908	CREATE PROC ObtenerPublicacionesDeUnAutor @a...
821577965	CREATE PROC ObtenerCantVendidaDeCadaPublicacion...
837578022	CREATE PROCEDURE ObtenerVentasDeUnaPublicacio...
853578079	CREATE PROCEDURE ObtenerAutoresDeUnaPublicaci...
869578136	CREATE PROCEDURE usp_GetErrorInfo AS SEL...
917578307	CREATE PROC ObtenerID @NomTabla varchar(20) ...
949578421	CREATE PROC EliminarPublicacion @title_id varchar(...
965578478	CREATE PROC EliminarAutor @au_id varchar(12) A...

También podemos obtener el código fuente de un stored procedure usando la función `OBJECT_DEFINITION`. Por ejemplo, para el procedimiento almacenado `EliminarAutor`:

```
SELECT OBJECT_DEFINITION(OBJECT_ID('pubs..EliminarAutor'))
```

Results

```
-----
CREATE PROC EliminarAutor
    @au_id varchar(12)
AS
    BEGIN TRY
        DELETE Authors where au_id = @au_id
        RETURN 0
    END TRY

    BEGIN CATCH
        EXECUTE usp_GetErrorInfo
        RETURN @@Error
    END CATCH

(1 row(s) affected)
```

También podemos acceder a información de los parámetros de procedimientos almacenados a través de la catalog view `sys.parameters`:

```
USE pubs
SELECT object_id, name, parameter_id, is_output, has_default_value
FROM sys.parameters
```

object_id	name	parameter_id	is_output	has_default_value
741577680	@percentage	1	0	0
789577851	@lolimit	1	0	0
789577851	@hilimit	2	0	0
789577851	@type	3	0	0
805577908	@au_id	1	0	0
837578022	@title_id	1	0	0
853578079	@title_id	1	0	0
917578307	@NomTabla	1	0	0
949578421	@title_id	1	0	0
965578478	@au_id	1	0	0

Por ejemplo, la siguiente consulta retorna todos los parámetros definidos para el procedimiento almacenado `'EliminarAutor'`:

```
SELECT sys.parameters.*
FROM sys.procedures INNER JOIN sys.parameters
    ON sys.procedures.object_id = sys.parameters.object_id
    INNER JOIN sys.schemas
    ON sys.procedures.schema_id = sys.schemas.schema_id
WHERE sys.schemas.name = 'dbo' AND
    sys.procedures.name = 'EliminarAutor'
```

object_id	name	parameter_id	system_type_id	user_type_id	max_length	precision	scale	is_output
965578478	@au_id	1	167	167	12	0	0	0



Stored procedures

También podemos obtener información sobre Stored Procedures usando la Information Schema View `INFORMATION_SCHEMA.ROUTINES`:

```
USE pubs
SELECT * FROM INFORMATION_SCHEMA.ROUTINES
```

La columna `ROUTINE_DEFINITION` proporciona el código de creación del SP.

Otra Information Schema View, `INFORMATION_SCHEMA.PARAMETERS`, proporciona información acerca de los parámetros a SPs:

```
USE pubs
SELECT * FROM INFORMATION_SCHEMA.PARAMETERS
```

2.9. Triggers

Podemos obtener los triggers definidos en una base de datos consultando la catalog view `triggers` del schema `sys` de la base de datos:

```
SELECT object_id, name, parent_id,  
       OBJECT_NAME(parent_id) tabla, type, type_desc  
FROM sys.triggers
```

object_id	name	parent_id	tabla	type	type_desc
709577566	employee_insupd	565577053	employee	TR	SQL_TRIGGER
933578364	InsertarBadSeller	2105058535	authors	TR	SQL_TRIGGER

Además, como vimos, podemos filtrar la catalog view `sys.objects` a fin de obtener los triggers definidos (type `'TR'`).

```
SELECT t.schema_id,  
       s.name Schema1,  
       type,  
       type_desc,  
       t.name trigger1,  
       object_id,  
       OBJECT_NAME(parent_object_id) 'Objeto Padre'  
FROM sys.objects t INNER JOIN sys.schemas s  
                  ON t.schema_id = s.schema_id  
WHERE t.type = 'TR'
```

schema_id	Schema1	type	type_desc	trigger1	object_id	Objeto Padre
1	dbo	TR	SQL_TRIGGER	employee_insupd	709577566	employee
1	dbo	TR	SQL_TRIGGER	InsertarBadSeller	933578364	authors

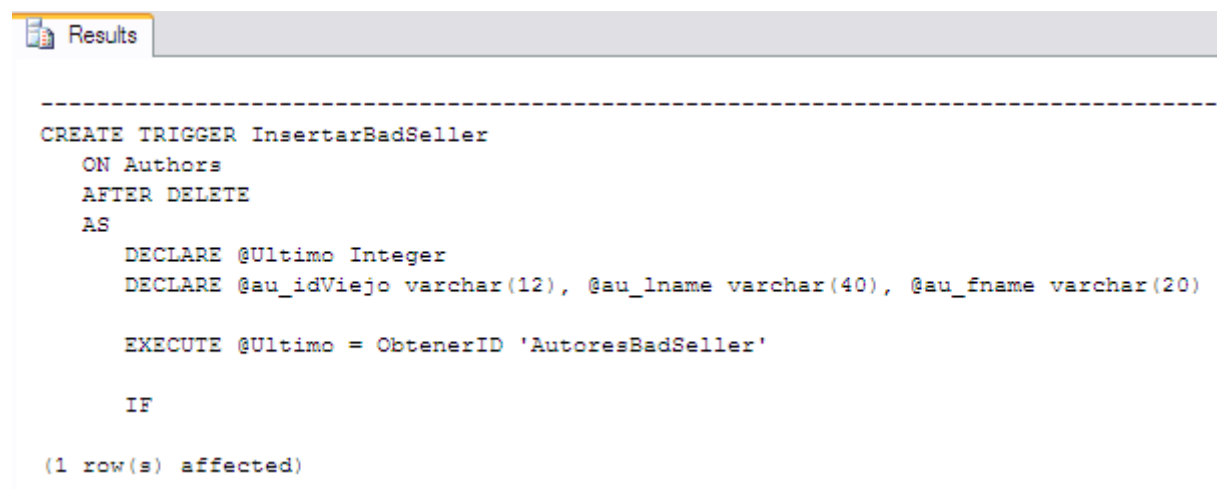
Dado un trigger, la catalog view `sys.sql_modules` nos proporciona más información, incluido el código fuente:

```
SELECT sys.sql_modules.object_id, definition
FROM sys.sql_modules INNER JOIN sys.objects
    ON sys.sql_modules.object_id = sys.objects.object_id
WHERE sys.objects.type = 'TR'
```

object_id	definition
709577566	CREATE TRIGGER employee_insupd ON employee FO...
933578364	CREATE TRIGGER InsertarBadSeller ON Authors AF...

También podemos obtener el código fuente de un trigger usando la función `OBJECT_DEFINITION`. Por ejemplo, para el trigger `InsertarBadSeller`:

```
SELECT OBJECT_DEFINITION(OBJECT_ID('pubs..InsertarBadSeller'))
```



The screenshot shows the 'Results' pane in SQL Server Enterprise Manager. It displays the source code of the 'InsertarBadSeller' trigger. The code is as follows:

```
-----
CREATE TRIGGER InsertarBadSeller
    ON Authors
    AFTER DELETE
    AS
        DECLARE @Ultimo Integer
        DECLARE @au_idViejo varchar(12), @au_lname varchar(40), @au_fname varchar(20)

        EXECUTE @Ultimo = ObtenerID 'AutoresBadSeller'

        IF
```

At the bottom of the results pane, it indicates '(1 row(s) affected)'.

2.10. Columnas

Podemos obtener las columnas definidas consultando la catalog view `columns` del schema `sys` de la base de datos:

```
SELECT t.name, c.name, c.system_type_id
FROM sys.tables t INNER JOIN sys.columns c
ON t.object_id = c.object_id
WHERE t.name = 'AUTHORS'
```

name	name	system_type_id
authors	au_id	167
authors	au_lname	167
authors	au_fname	167
authors	phone	175
authors	address	167
authors	city	167
authors	state	175
authors	zip	175
authors	contract	104



Columnas

También podemos obtener información acerca de las columnas usando la Information Schema View `INFORMATION_SCHEMA.COLUMNS`:

```
USE pubs
```

```
SELECT TABLE_NAME, COLUMN_NAME, ORDINAL_POSITION,
       DATA_TYPE,
       CHARACTER_MAXIMUM_LENGTH, IS_NULLABLE
FROM INFORMATION_SCHEMA.COLUMNS
```

2.11. Indices

Podemos obtener los índices definidos consultando la catalog view `indexes` del schema `sys` de la base de datos.

En el siguiente ejemplo filtramos por los índices definidos únicamente sobre tablas de usuario:

```
SELECT i.object_id 'Object_id de tabla', OBJECT_NAME(i.object_id) 'Tabla',  
       i.name,  
       index_id, i.type, i.type_desc  
FROM sys.indexes i INNER JOIN sys.tables t  
      ON i.object_id = t.object_id  
WHERE t.type = 'U'  
ORDER BY i.object_id, index_id
```

Object_id de tabla	Tabla	name	index_id	type	type_desc
37575172	publishers	UPKCL_pubind	1	1	CLUSTERED
101575400	titles	UPKCL_titleidind	1	1	CLUSTERED
101575400	titles	titleind	2	2	NONCLUSTERED
181575685	titleauthor	UPKCL_taind	1	1	CLUSTERED
181575685	titleauthor	auind	2	2	NONCLUSTERED
181575685	titleauthor	titleidind	3	2	NONCLUSTERED
245575913	stores	UPK_storeid	1	1	CLUSTERED
277576027	sales	UPKCL_sales	1	1	CLUSTERED
277576027	sales	titleidind	2	2	NONCLUSTERED

...