



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

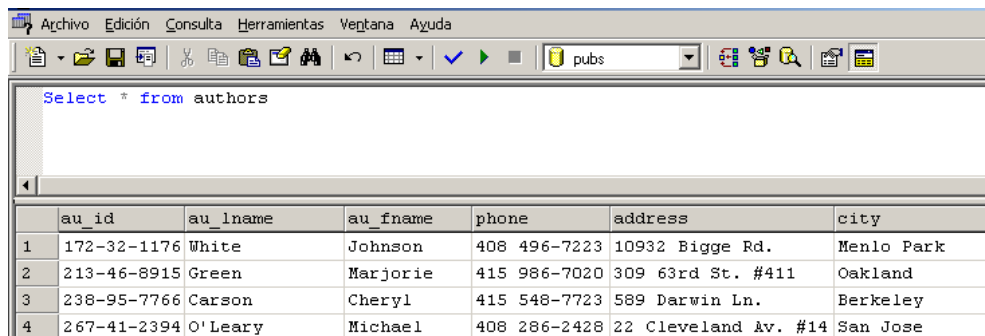
Bases de Datos
Guía de Trabajo Nro. 1
SQL: Consultas

Transact-SQL

- Transact-SQL es la implementación SQL del producto SQL Server de Microsoft.
- Las diferentes implementaciones o dialectos de SQL adoptan parte o todo del estándar y generalmente agregan además capacidades no especificadas en el mismo.

El analizador de consultas SQL

- El analizador de consultas SQL es la aplicación cliente que nos permite disparar sentencias SQL contra el RDBMS. Iniciamos el analizador de consultas desde el **Administrador Empresarial** seleccionando *Herramientas/Analizador de consultas SQL*.
- El analizador de consultas SQL es la aplicación cliente que nos permite disparar sentencias SQL contra el RDBMS. Iniciamos el analizador de consultas desde el **Administrador Empresarial** seleccionando *Herramientas/Analizador de consultas SQL*.



The screenshot shows the SQL Enterprise Manager application window. The menu bar includes Archivo, Edición, Consulta, Herramientas, Ventana, and Ayuda. The toolbar contains various icons for file operations, editing, and execution. The query window displays the SQL statement: `Select * from authors`. Below the query window, a results grid is shown with the following data:

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

- El área superior es un editor donde escribimos la consulta SQL. Ejecutamos la consulta haciendo click sobre la flecha verde (▶). Para ejecutar solo una fracción de la consulta tipeada, simplemente pintamos la fracción con el mouse y disparamos la consulta.

- Un **conjunto resultado**¹ es el conjunto de datos retornado por el RDBMS en respuesta a una consulta `SELECT`, y posee una disposición de columnas y filas. Los conjuntos resultado aparecen como una grilla en la parte inferior del *Analizador de Consultas SQL*.
- Un **conjunto resultado** puede resultar vacío (no contener filas), pero debe poseer al menos una columna.

Tipos de datos

- Los tipos de datos entre diferentes RDBMS no son estándar. SQL Server posee 17 tipos de datos. Los siguientes son los más comunes y compatibles entre diferentes RDBMS:
 - El tipo `char` es apto para datos de tipo carácter de longitud fija. Por ejemplo: `char(6)`.
 - El tipo `varchar` se aplica a datos de tipo carácter de longitud variable. Por ejemplo: `varchar(12)`.
 - El tipo `int` permite definir datos de tipo numérico entero. El tipo `smallint` sirve para el mismo fin pero para enteros pequeños desde `-32768` a `32768`.
 - El tipo `float` se aplica a datos de tipo numérico real..

Información para la realización de los ejercicios de práctica

La mayoría de los ejercicios de práctica se aplican al modelo físico de la base de datos `Pubs` provista por la instalación de SQL Server. `Pubs` modeliza un sistema de gestión de publicaciones.

¹ Result set.

La sentencia `SELECT`
Lista de salida del `SELECT`
La cláusula `FROM`

La forma más simple de sentencia `SELECT` sólo especifica las columnas a recuperar y la tabla donde se encuentran las mismas:

```
SELECT nomColumna1, nomColumna2
FROM tabla
```

A la lista de columnas le llamaremos *lista de salida del `SELECT`*.

Ordenamiento: La cláusula `ORDER BY`

Recordemos que la cláusula `ORDER BY` nos permite especificar el *orden* en el que aparecerán las filas recuperadas por una sentencia `SELECT`, y -de estar definida- debe ser la última cláusula de la sentencia `SELECT`².

El orden por omisión es el *ascendente*. Podemos anexar el modificador `DESC` para especificar un orden descendente.

1. Obtenga el código de título, título, tipo y precio incrementado en un 8% de todas las publicaciones, ordenadas por tipo y título.

Observe que la tercer columna del *conjunto resultado* no posee encabezado. Este encabezado se puede definir proporcionando un *alias* para esa columna.

Alias de columna

Podemos agregar un alias a una columna calculada, como en este caso. Sin embargo, un alias se puede agregar a cualquier columna cuyo nombre deseemos abreviar o hacer más significativo.

Especificamos un alias directamente después del nombre de la columna. Recordemos que en el caso de que el alias incluya espacios, debemos encerrarlo entre comillas:

```
SELECT au_lname 'Apellido del autor', city ciudad
FROM authors
```

² Cuando un RDBMS debe resolver una cláusula `ORDER BY`, realiza una búsqueda por algún índice existente definido sobre la columna especificada. Si existe, lo utiliza para resolver la consulta más rápidamente. Si no existe índice, el RDBMS debe construir una tabla temporal para resolver el orden. Esto siempre es más costoso en lo que respecta a performance.

- Reescriba la consulta del ejercicio 1 pero proporcionando el alias *precio actualizado* para la columna calculada.
- Modifique la consulta del ejercicio 2 a fin de obtener los datos por orden descendente de precio actualizado. Que observa respecto a los valores `NULL` de la columna ordenada?
- Muchos RDBMS permiten expresar el *número de orden en la lista de salida* del `SELECT` para identificar la columna sobre la cual se desea ordenar. Por ejemplo: `ORDER BY` 5. Reescriba la consulta de esta forma.

Constantes en la lista de salida del `SELECT`

Podemos especificar constantes como parte de la lista de salida del `SELECT`. Por ejemplo, la siguiente sentencia SQL utiliza el operador de concatenación `+` para generar un *conjunto resultado* con una única columna de salida:

```
SELECT 'El apellido del empleado es ' + lname 'Datos del empleado'
FROM employee
```

- Obtenga en una única columna el apellido y nombres de los autores separados por coma con una cabecera de columna *Listado de Autores*. Ordene el conjunto resultado.

- Obtenga un conjunto resultado para la tabla de publicaciones que proporcione, para cada fila, una salida como la siguiente.

	(Sin nombre de columna)
1	BU1032 posee un valor de \$19.99
2	BU1111 posee un valor de \$11.95
3	BU2075 posee un valor de \$2.99
4	BU7832 posee un valor de \$19.99
5	MC2222 posee un valor de \$19.99
6	MC3021 posee un valor de \$2.99

Conversión de datos numéricos a caracter

Podemos convertir datos de tipo numérico a caracter a través de la función de T-SQL `convert()`. La sintaxis de uso es:

```
convert (tipo-de-dato-destino, columna-a-convertir)
```

- Reescriba el ejercicio 6 utilizando la función T-SQL `convert()`.

Procedimientos almacenados del sistema

SQL Server incluye, con su instalación, una serie de procedimientos almacenados de utilidad. A estos procedimientos almacenados se les denomina *procedimientos almacenados del sistema*, y se caracterizan porque comienzan con el prefijo `sp_`.

Podemos ver un listado y descripción de los mismos en los **libros en pantalla**: *Referencia de Transact-SQL/Procedimientos almacenados del sistema*.

Los procedimientos almacenados del sistema se ejecutan directamente desde el editor como cualquier sentencia SQL. Sin embargo, si tenemos más de una sentencia SQL en el editor y al la invocación del procedimiento almacenado no es la primera, debemos anteponer la cláusula `EXEC` al nombre del mismo.

8. Investigue que *procedimiento almacenado del sistema* le proporciona información acerca de las tablas existentes en la base de datos `pubs`. Ejecútelo. Identifique las tablas de usuario.

9. Investigue que *procedimiento almacenado del sistema* le proporciona información acerca de las columnas de -por ejemplo- la tabla `authors` en `pubs`. Ejecútelo.

La cláusula `WHERE`

Hasta ahora no hemos especificado cláusulas `WHERE`. Las cláusulas `WHERE` nos permiten especificar una *condición* que las filas deben cumplir a fin de formar parte de la *lista de salida* del `SELECT`.

Operadores

Para especificar las **condiciones** de las cláusulas `WHERE` necesitamos de operadores *relacionales* y *lógicos*.

Los *operadores relacionales* en T-SQL son `>`, `>=`, `<`, `<=`, `=` y `<>`.

Los *operadores lógicos* son `AND`, `OR` y `NOT`.

10. Obtenga título y precio de todas las publicaciones que no valen más de \$13. Pruebe definir la condición de `WHERE` con el operador `NOT`.

Fechas en cláusulas **WHERE**

Las fechas en SQL Server son almacenadas como un valor *fecha+hora (datetime)*. Si no se especifica hora, se asume un valor default de medianoche (12:00:00.000AM).

Las consultas que involucran fechas deben tener en cuenta esta característica. Si no se conoce la hora exacta de la fecha buscada, no se deben utilizar operadores de igualdad. Por ejemplo, para saber que pedidos corresponden al 21 de junio de 2006, una condición de **WHERE** podría ser (fechaped >= '06/21/2006 ') and (fechaped < '06/22/2006')

Una consulta como esta recuperaría todos los pedidos realizados desde la cero hora del 21/06 inclusive hasta las 11:59:59 PM del mismo día.

El predicado **BETWEEN**

Recordemos que el predicado **BETWEEN** especifica la comparación dentro de un intervalo entre valores cuyo tipos de datos son comparables:

```
WHERE precio BETWEEN 5 AND 10
```

La cláusula **NOT** se puede utilizar con **BETWEEN** para indicar que la condición debe evaluar contra lo que existe fuera del intervalo:

```
WHERE precio NOT BETWEEN 5 AND 10
```

11. Obtenga los apellidos y fecha de contratación de todos los empleados que fueron contratados entre el 01/01/91 y el 01/01/92. Use el predicado **BETWEEN** para elaborar la condición.

El predicado [NOT] **IN**

Recordemos que el predicado **IN** especifica la comparación cuantificada: hace una lista de un conjunto de valores y evalúa si un valor está en la lista. La lista debe expresarse entre paréntesis.

```
WHERE precio IN (25, 30)
```

12. Obtenga los códigos, dirección y ciudad de los autores con código 172-32-1176 y 238-95-7766. Utilice el predicado **IN** para definir la condición de búsqueda. Modifique la consulta para obtener todos los autores que no poseen esos códigos.

El predicado `LIKE`. Caracteres comodín.

Recordemos que el predicado `LIKE` permite especificar una comparación de caracteres utilizando *caracteres comodín* (*wild cards*) para recuperar filas cuando solo se conoce un *patrón* del dato buscado. `LIKE` generalmente se utiliza sobre columnas de tipo carácter, pero se puede utilizar también con columna de tipo fecha.

Los caracteres comodín soportados por MS SQL Server son³:

- `%` 0 a n caracteres (de cualquier carácter).
- `_` exactamente un carácter (de cualquier carácter).
- `[]` exactamente un carácter del conjunto o rango especificado, por ej.: `[aAc]` o `[a-c]`
- `[^]` Que no coincida con el conjunto o rango especificado.

13. Obtenga código y título de todos los títulos que incluyen la palabra `Computer` en su título.

El predicado `NULL`

Un valor `NULL` para una columna indica que el valor para esa columna es desconocido. No es lo mismo que blanco, cero o cualquier otro valor discreto. A veces puede significar que el valor para una columna particular no es significativo.

Los valores `NULL` son casos especiales y deben tratarse en forma especial cuando se realizan comparaciones.

A los propósitos del ordenamiento, los valores `NULL` son considerados los más bajos.

14. Obtenga el nombre, ciudad y estado de las editoriales cuyo estado (columna `state`) no está definido. Recordemos que para ello debemos utilizar la cláusula `IS` (o su negación `IS NOT`) Que sucede si explícitamente compara la columna contra el valor `NULL`?

15. La admisión de valores `NULL` para una columna se define en el momento de creación de la estructura de la tabla. Como averiguaría a posteriori si la columna `state` de la tabla de editoriales admite o no valores nulos?

³ Cuando se utiliza `LIKE` y se solicita la búsqueda de un patrón que comienza con un carácter comodín, SQL Server no puede utilizar índices -si es que están definidos- para mejorar la performance de la consulta. Puede utilizarlos sólo si el patrón comienza con una constante.

Funciones de agregación.

La cláusula **DISTINCT**

Recordemos que las *funciones de agregación* ANSI SQL son **COUNT**, **COUNT (*)**, **SUM**, **MAX**, **MIN** y **AVG**.

La cláusula **DISTINCT** se puede utilizar asociada a las *funciones de agregación* **COUNT**, **SUM**, y **AVG** cuando necesitamos que la función se aplique **sólo** a valores diferentes. O sea, no deseamos que cuenten, sumen o promedien valores repetidos en diferentes filas:

```
SELECT COUNT (DISTINCT type)
FROM titles
```

Recordemos también que las funciones **COUNT**, **SUM**, y **AVG** no consideran a los valores **NULL** de las columnas a las que se aplican.

Debemos tener en cuenta que si tenemos una *función de agregación* en la lista de salida de un **SELECT**, solo podrán existir otras funciones de agregación en tal lista de salida. (La excepción ocurre en combinación con la cláusula **GROUP BY**).

Las funciones de agregación se pueden aplicar también a expresiones.

16. Obtenga la cantidad de publicaciones con precio en contraposición a la cantidad total de publicaciones.

17. Obtenga la cantidad de precios diferentes en la tabla de publicaciones.

18. Algunas funciones de agregación pueden aplicarse a fechas. Obtenga el apellido y nombre del empleado contratado más recientemente.

19. La columna `ytd-sales` de la tabla `titles` posee la cantidad vendida de cada título. Obtenga la cantidad de dinero recaudada en ventas.

Obtener los componentes de una fecha

Hay tres funciones ANSI SQL estándar para obtener los componentes de día, mes y año de una columna de tipo fecha (**DATETIME** en SQL Server):

YEAR(columna) retorna el año de la fecha como un entero de cuatro dígitos,
MONTH(columna) proporciona el mes de la fecha como un entero de 1 a 12, mientras que
DAY (columna) retorna el día del mes de la fecha como un entero.

20. La información de publicaciones vendidas reside en la tabla `sales` (ventas). Calcule la cantidad de publicaciones vendidas (columna `qty`) para el mes de Junio.

Funciones no estándar

- Transact-SQL, como todos los RDBMS comerciales, posee una serie de funciones predefinidas que no son ANSI SQL estándar.
- Debemos tener en cuenta que, cuando se aplican funciones a las columnas de salida de un `SELECT`, los RDBMS no pueden utilizar los índices –de estar definidos– para resolver la consulta de manera más eficiente. El utilizar funciones como parte de cláusulas `WHERE` puede degradar también la performance.

Funciones de manejo de texto

- Ya vimos que podíamos convertir el dato de una columna a un tipo destino a través de la función `convert()`. La misma función tiene una versión extendida para convertir datos de columnas de tipo `datetime` a diferentes formatos. La sintaxis es:

```
convert (varchar, columna-datetime, codigo-de-formato)
```

`codigo-de-formato` es un código que establece como se va a mostrar la fecha en formato varchar. Por ejemplo, el formato 3 muestra la fecha con formato `dd/mm/yyyy`.

- La función `patindex()` retorna la posición de la primer ocurrencia de un patrón dentro de una cadena. El patrón puede incluir comodines. Su sintaxis es:

```
patindex ('%patron%', columna-o-expresion)
```

Un valor de retorno 0 indica que no se hallaron ocurrencias.

- La función `substring()` extrae una subcadena de una cadena principal. Su sintaxis es:

```
substring (columna-o-expresion, desde, cantidad)
```

`columna-o-expresion` es la cadena desde la cual se extraeran los caracteres. `desde` es la posición de inicio de la extracción. `substring()` retorna una cadena de tipo `varchar`.

- La función `str()` convierte una expresión numérica a una cadena. Su sintaxis es:

`str (expresion-numerica, longitud-destino, cantidad-decimales)`

- Otras funciones de utilidad para manejo de texto son `right()`, `upper()`, `lower()`, `rtrim()` y `ltrim()`. Una vez finalizada la guía, consulte los *libros en pantalla* para averiguar su aplicación y sintaxis.

Funciones de manejo de fechas

- T-SQL proporciona una serie de funciones llamadas *niladic*, que se caracterizan por no requerir ni parámetros ni paréntesis para su invocación. Una de ellas es `CURRENT_TIMESTAMP`, que retorna la fecha y hora actual como un valor `datetime`.
- Otras funciones de utilidad para manejo de fechas son `dateadd()` y `datediff()`. Una vez finalizada la guía, consulte los libros en pantalla para averiguar su aplicación y sintaxis.

Tablas del sistema

- MS SQL Server almacena su propio diccionario de datos en un modelo físico conocido como *tablas del sistema*. Estas tablas del sistema son en si objetos de cualquier base de datos que se cree. Algunas de las tablas del sistema son:

<code>sysobjects</code>	Posee información sobre objetos tales como tablas y procedimientos.
<code>sysindexes</code>	Posee información sobre <i>índices</i>
<code>syscolumns</code>	Posee información sobre cada <i>columna</i> en una tabla.
<code>sysusers</code>	Posee información de los usuarios de la base de datos actual.

Otras tablas del sistema sólo están definidas en la base de datos `master`:

<code>sysdatabases</code>	Información de todas las <i>bases de datos</i> .
<code>syslogins</code>	Login Ids y passwords.
<code>syslocks</code>	Seguimiento de bloqueo de objetos.
<code>sysmessages</code>	Mensajes de error de MS SQL Server.

Objetos de la base de datos

Podemos obtener información acerca de los objetos definidos en nuestra base de datos consultando la tabla del sistema `sysobjects`. Por ejemplo, la siguiente sentencia SQL nos muestra el nombre de todos los objetos definidos en `Pubs`:

```
SELECT name
FROM sysobjects
WHERE type = 'U'
```

21. Ejecute un `Select` sobre la tabla del sistema `sysobjects` consultando por los tipos `S`, `V`, `P`, `TR`, `K` y `FK`.

Funciones que brindan información del sistema

- La función `USER` retorna el nombre del *database user* actual como un tipo `varchar`. `SYSTEM_USER` retorna el *login ID* actual. Estas funciones son muy útiles en la codificación de *procedimientos almacenados* a fin de testear si el usuario actual posee privilegios de acceso para realizar determinadas operaciones. Por ejemplo, un *procedimiento almacenado* puede estar preparado para ser ejecutado solo por el propietario de la base de datos al cual pertenece (database user *dbo* en SQL Server).
- La función `datalength()` retorna la longitud en bytes de la variable o columna que se indique como parámetro. Se puede aplicar a variables y columnas de cualquier tipo de dato. Si se trata de columnas de longitud fija, la función retorna la longitud definida. Si se trata de columnas con longitud variable, la función retorna la *longitud real del valor* de esa columna en la fila especificada.
- La función `col_length()` es similar a `datalength()` en que retorna el ancho de una columna. Sin embargo, mientras `datalength()` retorna el ancho de una instancia particular del dato, que puede ser de longitud variable, `col_length()` extrae la información de la tabla `syscolumns` que contiene la máxima longitud definida para una columna, sea ésta de longitud fija o variable.