Links de Interés para la resolución de prácticas de SQL

NOTA IMPORTANTE: Todos los links aquí presentados, son a sólo modo de ejemplo de la innumerable cantidad de información existente en Internet, que hace referencia tanto a la plataforma Microsoft propuesta, su manipulación y gestión, así como información general sobre los lenguajes SQL que se utilizan y explican en clase en el transcurso de la cursada. La cátedra no determina la utilización exclusiva de ningún material, sólo da indicio del tipo de material con el que el alumno puede contar.

Se propone que nuestras prácticas sean resueltas sobre la plataforma libre de Microsoft SQL Server Express o cualquier otra como ser: SQLite (https://www.sqlite.org/cli.html) o MySQL (http://www.mysql.com/downloads/).

Para la instalación de la plataforma de base de datos, se deben bajar 2 archivos los que contienen:

1. Motor de base de datos (SQL Express Engine): https://www.microsoft.com/en-us/download/confirmation.aspx?id=55994
2. Herramienta de administración (Sql Mannagement Sutio Express): https://aka.ms/ssmsfullsetup

Como recurso a mano para tener una definición de todos los comandos y sentencias usualmente utilizados en el diseño, desarrollo y administración de base de datos pueden contar con:

1. En inglés pero realmente muy completo: http://www.tutorialspoint.com/sql/index.htm
2. En español, del propio Microsoft: http://msdn.microsoft.com/es-es/library/bb510741.aspx
3. Otra opción: http://es.tldp.org/Postgresql-es/web/navegable/tutorial/sql-language.html

Conocimientos básicos sobre T-SQL (Sql Server)

Recopilación de información relevante sobre TRANSACT SQL para Microsoft SQL Server.

Tipos de datos

Numéricos exactos

bigint	numeric
bit	smallint
decimal	smallmoney
int	tinyint
money	

Numéricos aproximados

float	real

Fecha y hora

date	datetimeoffset
datetime2	smalldatetime
datetime	time

Cadenas de caracteres

char	varchar
text	

Cadenas de caracteres Unicode

nchar	nvarchar
ntext	

Cadenas binarias

binary	varbinary
image	

Otros tipos de datos

cursor	timestamp
hierarchyid	uniqueidentifier

sql_variant	xml
tabla	Tipos espaciales

Sentencias de creación y eliminación

Crear base de datos

La creación de bases de datos directamente desde la línea de comando, puede tener un sinnúmero de opciones, pero lo usual, a niveles básicos alcanza con escribir:

```
CREATE DATABASE [NombreBaseDatos];
```

Esta sentencia, dejará los seteos posibles en sus valores por defecto, lo que en principio no nos causará inconvenientes, y cuyos valores y parámetros podremos configurar y modificar a futura desde código, como desde el entorno gráfico SSMS.

Crear tabla

La sentencia básica para la creación de tablas se puede definir como:

```
CREATE TABLE NombreTabla
    (Campo1 int PRIMARY KEY NOT NULL,
     Campo2 varchar(25) NOT NULL,
     Campo3 money NULL,
          .
          .
          .
          .
          .
     CampoN text NULL)
GO
```

Adicionar columna

Una tabla se puede modificar desde código con la sentencia ALTER TABLE de la siguiente manera:

```
ALTER TABLE NombreTabla ADD column_b VARCHAR(20) NULL, column_c INT NULL;
```

Quitar columna

Para eliminar una columna podemos proceder:

```
ALTER TABLE dbo.doc_exb DROP COLUMN column_b;
```

Elminiar tabla

Y finalmente para eliminar la tabla completa:

```
DROP TABLE NombreTabla;
```

Sentencias sobre los datos

Agregar datos

```
INSERT INTO table
(column1, column2, ... )
VALUES
(expression1, expression2, ... ),
(expression1, expression2, ... ),
...;
```

```
INSERT INTO table
(column1, column2, ... )
SELECT expression1, expression2, ...
FROM source_table
[WHERE conditions];
```

Modificar datos

```
UPDATE [ database_name . [ schema_name ] . | schema_name . ] table_name
SET { column_name = { expression | NULL } } [ ,...n ]
[ FROM from_clause ]
[ WHERE <search_condition> ]
[ OPTION ( LABEL = label_name ) ]
[;]
```

Eliminar datos

```
DELETE
    [ FROM [database_name . [ schema ] . | schema. ] table_name ]
    [ WHERE <search_condition> ]
    [ OPTION ( <query_options> [ ,...n ]  ) ]
[; ]
```

Sentencias y cláusulas de consulta

Seleccionar Filas

Se realiza con la sentencia SELECT, y se especifica de la siguiente manera:

```
SELECT Campo1, Campo2, Campo3, …… , CampoN FROM NombreTabla
```

Permite la utilización del operador * que representa {TODO}. Puede utilizarse solo o acompañado del nombre de la tabla (TablaNombre.*).

La cláusula WHERE

Aplica criterios de selección para la devolución de selecciones de datos.

Va especificado luego de la cláusula FROM de un SELECT

```
SELECT Campo1, Campo2, Campo3, …… , CampoN FROM NombreTabla
WHERE [criterios de seleccion]
```

Para los criterios de selección, se pueden utilizar las operaciones lógicas tradicionales, o funciones especiales concatenadas por operadores AND, OR, XOR, etc. Según corresponda.

=
: Es el operador que se utiliza para probar la igualdad entre dos expresiones.

<>
: Es el operador que se utiliza para probar si dos expresiones no son iguales entre sí.

!=
: Es el operador que se utiliza para probar si dos expresiones no son iguales entre sí.

>
: Es el operador que se utiliza para probar si una expresión es mayor que la otra.

>=
: Es el operador que se utiliza para probar si una expresión es mayor o igual que la otra expresión.

!>
: Es el operador que se utiliza para probar si una expresión no es mayor que la otra expresión.

<
: Es el operador que se utiliza para probar si una expresión es menor que la otra.

<=
: Es el operador que se utiliza para probar si una expresión es menor o igual que la otra expresión.

!<
: Es el operador que se utiliza para probar si una expresión no es menor que la otra expresión.

Intervalos de Valores

Para indicar que deseamos recuperar los registros según el intervalo de valores de un campo emplearemos el operador Between cuya sintaxis es:

```
campo [Not] Between valor1 And valor2 (la condición Not es opcional)
```

En este caso la consulta devolvería los registros que contengan en "campo" un valor incluido en el intervalo valor1, valor2 (ambos inclusive). Si anteponemos la condición Not devolverá aquellos valores no incluidos en el intervalo.

```
SELECT *
FROM
    Pedidos
WHERE
    CodPostal Between 28000 And 28999
```

El Operador Like

Se utiliza para comparar una expresión de cadena con un modelo en una expresión SQL. Su sintaxis es:

```
expresión Like modelo
```

En donde expresión es una cadena modelo o campo contra el que se compara expresión. Se puede utilizar el operador Like para encontrar valores en los campos que coincidan con el modelo especificado. Por modelo puede especificar un valor completo (Ana María), o se puede utilizar una cadena de caracteres comodín como los reconocidos por el sistema operativo para encontrar un rango de valores (Like An*).

El operador Like se puede utilizar en una expresión para comparar un valor de un campo con una expresión de cadena. Por ejemplo, si introduce Like C* en una consulta SQL, la consulta devuelve todos los valores de campo que comiencen por la letra C. En una consulta con parámetros, puede hacer que el usuario escriba el modelo que se va a utilizar.

El ejemplo siguiente devuelve los datos que comienzan con la letra P seguido de cualquier letra entre A y F y de tres dígitos:

```
Like 'P[A-F]###'
```

Este ejemplo devuelve los campos cuyo contenido empiece con una letra de la A a la D seguidas de cualquier cadena.

```
Like '[A-D]*'
```

En la tabla siguiente se muestra cómo utilizar el operador Like para comprobar expresiones con diferentes modelos.

Ejemplo	Descripción
LIKE 'A%'	Todo lo que comience por A
LIKE '_NG'	Todo lo que comience por cualquier carácter y luego siga NG
LIKE '[AF]%'	Todo lo que comience por A ó F
LIKE '[A-F]%'	Todo lo que comience por cualquier letra comprendida entre la A y la F
LIKE '[A^B]%'	Todo lo que comience por A y la segunda letra no sea una B

En determinado motores de bases de datos, esta cláusula, no reconoce el asterisco como carácter comodín y hay que sustituirlo por el carácter tanto por ciento (%).

El Operador In

Este operador devuelve aquellos registros cuyo campo indicado coincide con alguno de los en una lista. Su sintaxis es:

```
expresión [Not] In(valor1, valor2, . . .)
```

```
SELECT *
FROM
    Pedidos
WHERE
    Provincia In ('Madrid', 'Barcelona', 'Sevilla')
```

La cláusula GROUP BY

Agrupa un conjunto de filas seleccionado en un conjunto de filas de resumen de acuerdo con los valores de una o más columnas o expresiones en SQL Server 2014. Se devuelve una fila para cada

grupo. Las funciones de agregado de la lista <select> de la cláusula SELECT proporcionan información de cada grupo en lugar de filas individuales.

Las expresiones de la cláusula GROUP BY pueden contener columnas de las tablas, de las tablas derivadas o de las vistas de la cláusula FROM. No es necesario que aparezcan las columnas en la lista de <selección> de la cláusula SELECT.
Deben incluirse en la lista GROUP BY todas las columnas de la tabla o la vista de cualquier expresión no agregada de la lista de <selección>:

- **Están permitidas las siguientes instrucciones:**
- `SELECT ColumnA, ColumnB FROM T GROUP BY ColumnA, ColumnB;`
- `SELECT ColumnA + ColumnB FROM T GROUP BY ColumnA, ColumnB;`
- `SELECT ColumnA + ColumnB FROM T GROUP BY ColumnA + ColumnB;`
- `SELECT ColumnA + ColumnB + constant FROM T GROUP BY ColumnA, ColumnB;`
- **No están permitidas las siguientes instrucciones:**
- `SELECT ColumnA, ColumnB FROM T GROUP BY ColumnA + ColumnB;`
- `SELECT ColumnA + constant + ColumnB FROM T GROUP BY ColumnA + ColumnB;`

Si se incluyen funciones de agregado en la <lista de selección> de la cláusula SELECT, GROUP BY calcula un valor de resumen para cada grupo. Se conocen como agregados vectoriales.
Las filas que no cumplen las condiciones especificadas en la cláusula WHERE se quitan antes de realizar ninguna operación de agrupación.
La cláusula HAVING se usa junto con la cláusula GROUP BY para filtrar los grupos en el conjunto de resultados.
La cláusula GROUP BY no ordena el conjunto de resultados. En su lugar, use la cláusula ORDER BY para ordenarlo.
Si una columna de agrupamiento contiene varios valores NULL, todos ellos se consideran equivalentes y se colocan en un grupo individual.
No es posible usar GROUP BY con un alias para reemplazar el nombre de una columna en la cláusula AS, a menos que dicho alias sustituya a un nombre de columna en una tabla derivada de la cláusula FROM.

La cláusula ORDER BY

Ordenar los datos devueltos por una consulta en SQL Server. Use esta cláusula para:

- Ordenar el conjunto de resultados de una consulta por la lista de columnas especificada y, opcionalmente, limitar las filas devueltas a un intervalo especificado. El orden en que se devuelven las filas en un conjunto de resultados no se puede garantizar, a menos que se especifique una cláusula ORDER BY.
- Determinar el orden en que se aplica el conjunto de resultados.

Se indica al final de todas las cláusulas de selección, agrupamiento, y condicionado. Se escribe de la siguiente manera:
SELECT * FROM Tabla WHERE .....
GROUP BY Columna1 {orden}, Columna2 {orden}, ..., ColumnaN {orden}

Donde ORDEN puede tomar los valores:

- ASC: ascendente.
- DESC: descendente.

La cláusula HAVING

Especifica una condición de búsqueda para un grupo o agregado. HAVING solo se puede utilizar con la instrucción SELECT. Normalmente, HAVING se utiliza en una cláusula GROUP BY. Cuando no se utiliza GROUP BY, HAVING se comporta como una cláusula WHERE.

```
HAVING [Condición de selección]
```

En el ejemplo siguiente, donde se utiliza una cláusula HAVING simple, se recupera el total de cada SalesOrderID de la tabla SalesOrderDetailque exceda $100000.00.

```
USE AdventureWorks2012 ;
GO
SELECT SalesOrderID, SUM(LineTotal) AS SubTotal
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID
HAVING SUM(LineTotal) > 100000.00
ORDER BY SalesOrderID ;
```

La cláusula DISTINCT

Devuelve todas los estados posibles para la columna seleccionada del modelo. Los valores devueltos varían dependiendo de si la columna especificada contiene valores discretos, valores numéricos de datos discretos o valores numéricos continuos.

```
SELECT DISTINCT <expression list> FROM <model>
[WHERE <condition list>][ORDER BY <expression>]
```

La instrucción **SELECT DISTINCT FROM** solo funciona con una sola columna o con un conjunto de columnas relacionadas. Esta cláusula no funciona con un conjunto de columnas no relacionadas.
La instrucción **SELECT DISTINCT FROM** le permite hacer referencia directamente a una columna dentro de una tabla anidada.

Los resultados de la instrucción **SELECT DISTINCT FROM <model>** varían en función del tipo de columna. En la siguiente tabla se describen los tipos de columna admitidos y la salida de la instrucción.

Tipo de columna	Salida
Discreta	Valores únicos de la columna.
De datos discretos	Punto medio de cada depósito de datos discretos de la columna.
Continua	Punto medio de los valores de la columna.

Procedimientos almacenados

Los procedimientos almacenados son similares a los procedimientos de otros lenguajes de programación en tanto que pueden:

- Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al lote o al procedimiento que realiza la llamada.
- Contener instrucciones de programación que realicen operaciones en la base de datos, incluidas las llamadas a otros procedimientos.
- Devolver un valor de estado a un lote o a un procedimiento que realice una llamada para indicar si la operación se ha realizado correctamente o se han producido errores, y el motivo de estos.

Para la creación de los mismos utilizamos la sentencia CREATE PROCEDURE cuya sintaxis sigue los siguientes lineamientos:

```
CREATE PROCEDURE NombreStoredProcedure
        @param1 {tipodato},
        @param2 {tipodato},
                .
                .
                .
                .
                .
                .
        @paramN {tipodato}
AS
    SET NOCOUNT ON;
    SELECT *
    FROM NombreAlgunaTabla;
GO
```

@parametro

Parámetro declarado en el procedimiento. Especifique un nombre de parámetro con una arroba (**@**) como el primer carácter. El nombre del parámetro se debe ajustar a las reglas de los identificadores. Los parámetros son locales respecto al procedimiento; los mismos nombres de parámetro se pueden usar en otros procedimientos.
Se pueden declarar uno o varios parámetros; el valor máximo es 2.100. El usuario debe proporcionar el valor de cada parámetro declarado cuando se llame al procedimiento, a menos que se haya definido un valor predeterminado para el parámetro o se haya establecido en el mismo valor que el de otro parámetro. Si un procedimiento contiene parámetros con valores de tabla y el parámetro no está en la llamada, se pasa una tabla vacía. Los parámetros solo pueden ocupar el lugar de expresiones constantes; no se pueden usar en lugar de nombres de tablas, nombres de columnas o nombres de otros objetos de base de datos.

Un procedimiento almacenado puede ser llamado de las siguientes maneras:

```
EXECUTE NombreStoredProcedure;
GO
-- o
EXEC NombreStoredProcedure;
GO
-- o directamente:
NombreStoredProcedure;
```

Comentarios importantes

No hay ningún tamaño máximo predefinido para un procedimiento.
Las variables especificadas en el procedimiento las puede definir el usuario o pueden ser variables del sistema, como @@SPID.
Cuando un procedimiento se ejecuta por primera vez, se compila para determinar que dispone de un plan de acceso óptimo para recuperar los datos. En las siguientes ejecuciones del procedimiento se puede volver a usar el plan ya generado si aún permanece en la memoria caché de planes de Motor de base de datos.
Los procedimientos se anidan cuando un procedimiento llama a otro o ejecuta código administrado mediante una referencia a una rutina, tipo o agregado CLR. Los procedimientos y las referencias a código administrado se pueden anidar hasta 32 niveles. El nivel de anidamiento aumenta en uno cuando el procedimiento o la referencia a código administrado a los que se ha llamado empiezan a ejecutarse, y disminuye en uno cuando se completa su ejecución. Los métodos que se invocan desde el código administrado no cuentan para este límite de niveles de anidamiento. Sin embargo, cuando un procedimiento almacenado CLR realiza operaciones de acceso a datos mediante el proveedor administrado de SQL Server, se agrega un nivel de anidamiento adicional en la transición desde código administrado a SQL.
Si se intenta superar el nivel máximo de anidamiento, se producirá un error en toda la cadena de llamada. Puede usar la función @@NESTLEVEL para devolver el nivel de anidamiento de la ejecución del procedimiento almacenado actual.