Indice

Referencias a tablas

En Maple, toda referencia a tabla deberá ser procedida por el símbolo $. Esto indica a Maple que lo que le sigue a este símbolo es el nombre de una tabla.
Por ejemplo, si tenemos una tabla Persona, cuando queramos hacer referencia a ella desde Maple, la llamaremos $Persona.

Visualización de datos (SELECT)

Símbolo: >

Al saber referenciar tablas, ya podemos utilizar esas referencias para mostrar datos, haciendo uso de la operación SELECT. Utilizaremos el símbolo > para notificarle a Maple que ésta es la operación que deseamos hacer.

Tipos de SELECT

Implícito

Este SELECT sólo puede utilizarse si deseamos hacer un SELECT simple, y sobre una sola tabla. En este SELECT, no es necesario escribir el símbolo de SELECT >, ya que al darle a Maple una referencia sobre una tabla, Maple entiende que lo que deseamos hacer es un SELECT sobre ella.
Ejemplo:

Input	Output
$Personas	SELECT * FROM `Personas`;

Explícito

Este tipo de SELECT lo utilizaremos si deseamos filtrar un SELECT por columnas y/o agregar INNER JOINs.
Para especificar columnas, pondremos los nombres de cada una de ellas después del símbolo >.
En este caso, es obligatorio utilizar el símbolo de SELECT >

Input	Output
$Personas >	SELECT * FROM `Personas`;
$Personas > nombre, apellido	SELECT `nombre`, `apellido` FROM `Personas`;
$Personas > nombre, apellido, SUM(edad)	SELECT `nombre`, `apellido`, SUM(edad) FROM `Personas`;

Importante: no agregar `` para especificar los nombres de las columnas. Esto puede derivar en comportamiento no deseado. Maple agregará estas comillas de forma automática al momento de convertir la sentencia a SQL, salvo que la columna tenga una función o forme parte de un condicional.

Condicionales

Símbolo: ?

En una sentencia Maple SELECT, podemos escribir condicionales con el fin de filtrar resultados. Es posible utilizar condicionales con ambos tipos de SELECT, el implícito y explícito.
La sintaxis de los condicionales es igual a la de cualquier condicional MySQL. Maple acepta conectores lógicos &&/AND, ||/OR, e incluso LIKE.

Input	Output

$Personas ? nombre = ‘John’	SELECT * FROM `Personas` WHERE nombre = 'John';
$Personas > ? nombre = ‘John’	SELECT * FROM `Personas` WHERE nombre = 'John';
$Personas > nombre, apellido ? edad > 18	SELECT `nombre`, `apellido` FROM `Personas` WHERE edad > 18;
$Personas > nombre, apellido ? nombre LIKE ‘%on’	SELECT `nombre`, `apellido` FROM `Personas` WHERE nombre LIKE '%on';

Nota: por ahora, en esta versión de Maple, las columnas que se encuentren en los condicionales no serán envueltas con ``.

GROUP/ORDER BY

Al utilizar el SELECT explícito, podemos hacer uso de GROUP BY y ORDER BY. Esto se logra especificando columnas, y utilizando los símbolos * y ^ respectivamente.

Input	Output
$Personas > ^nombre	SELECT `nombre` FROM `Personas` ORDER BY 1;
$Personas > ^nombre, *edad	SELECT `nombre`, `edad` FROM `Personas` GROUP BY 2 ORDER BY 1;
$Personas > ^*nombre, edad	SELECT `nombre`, `edad` FROM `Personas` GROUP BY 1 ORDER BY 1;

Importante: Maple no soporta agrupar u ordenar por varias columnas a la vez. Esto quiere decir, que no es posible marcar dos columnas como ORDER BY. Es sólo una columna por sentencia.
Por ejemplo

Input	Output
$Personas > ^nombre, ^apellido	SELECT `nombre`, ^`apellido` FROM `Personas` ORDER BY 1;

Como podemos ver, Maple nos devuelve una sentencia SQL que no es válida, ya que sólo procesó la primera columna en el ORDER BY.

INNER JOIN

Símbolo: <>

En las sentencias SELECT explícitas, es posible especificar tablas adicionales, las cuales se unirán a la consulta original mediante un INNER JOIN. Maple tiene dos tipos de INNER JOIN.

Implícito

En el INNER JOIN implícito, Maple “asume” el nombre de las columnas a relacionar y las incluye automáticamente en la consulta resultante. Para que este tipo de INNER JOIN funcione, es necesario que se respete la nomenclatura: tabla1.id_tabla2 = tabla2.id
Por ejemplo:

Input	Output
$Personas > <> $Roles	SELECT * FROM `Personas` INNER JOIN `Roles` ON Personas.id_Roles = Roles.id;
$Personas > <> $Ciudades <> $Paises	SELECT * FROM `Personas` INNER JOIN `Ciudades` ON Personas.id_Ciudades =

	Ciudades.id INNER JOIN `Paises` ON Ciudades.id_Paises = Paises.id;

Esto quiere decir que, si utilizamos un INNER JOIN implícito y nuestras tablas NO respetan esta nomenclatura, la consulta SQL resultante no funcionará al ejecutarse contra la base de datos.

Explícito

En este tipo de INNER JOIN, le especificamos a Maple sobre qué columnas vamos a llevar a cabo la unión. Es necesario utilizar este tipo de INNER JOIN si no deseamos seguir la nomenclatura descripta anteriormente.

Input	Output
$Personas > <role_id = id_roles> $Roles	SELECT * FROM `Personas` INNER JOIN `Roles` ON Personas.role_id = Roles.id_roles;

Uso de condicionales

Al ser el INNER JOIN una extensión del SELECT explícito, es posible especificar condicionales.

Input	Output
$Personas > <> $Roles ? role_desc = 'admin'	SELECT * FROM `Personas` INNER JOIN `Roles` ON Personas.id_Roles = Roles.id WHERE role_desc = 'admin';

Funciones nativas de SQL

Maple es compatible con funciones nativas de SQL. Si la función SQL dada actúa sobre el nombre de una columna, el SELECT deberá ser explícito. Si utilizamos alguna función SQL que actúe sobre alguno de los valores en los condicionales, el SELECT podrá ser implícito o explícito.
Por ejemplo:

Input	Output
$Producto > AVG(precio), SUM(stock)	SELECT AVG(precio), SUM(stock) FROM `Producto`;
$Producto > AVG(precio), SUM(stock) <> $Proveedor	SELECT AVG(precio), SUM(stock) FROM `Producto` INNER JOIN `Proveedor` ON Producto.id_Proveedor = Proveedor.id;
$Producto > AVG(precio), SUM(stock) <> $Proveedor ? nombre_proveedor != 'Microsoft'	SELECT AVG(precio), SUM(stock) FROM `Producto` INNER JOIN `Proveedor` ON Producto.id_Proveedor = Proveedor.id WHERE nombre_proveedor <> 'Microsoft';

Nota: Maple soporta funciones anidadas.

LIMIT

Símbolos: []

Maple permite limitar la cantidad de resultados que serán devueltos por la consulta generada haciendo uso de la palabra clave LIMIT de MySQL. El equivalente de LIMIT en Maple, son corchetes con el número de resultados que deseamos obtener. Es posible utilizar LIMIT con cualquier tipo de SELECT, incluyendo SELECT con INNER JOIN, con condicionales, explícitos e implícitos.

Input	Output
$Personas [1000]	SELECT * FROM `Personas` LIMIT 1000;

$Personas > id, nombre ? nombre = 'John' [1000]	SELECT `id`, `nombre` FROM `Personas` WHERE nombre = 'John' LIMIT 1000;

Maple soporta LIMIT por rangos. Esto se logra especificando el rango entre los corchetes.

Input	Output
$Personas [1000,2000];	SELECT * FROM `Personas` LIMIT 1000,2000;

Ingreso de datos (INSERT)

Símbolo: <

Maple ofrece dos maneras de ingresar datos a una tabla. La manera implícita y la explícita. Al igual que el SELECT, la tabla que vamos a referenciar debe ser precedida por un signo $.

Tipos de INSERT

Implícito

En este tipo de INSERT, no se especifican las columnas en las cuales se ingresarán datos, sino que Maple asume que los datos ingresados son compatibles y en orden con las columnas en la tabla.
Por ejemplo:

Input	Output
$Personas < 1, 'Harry', 'Potter';	INSERT INTO `Personas` VALUES(1, 'Harry', 'Potter');

Explícito

En la mayoría de los casos, por temas de seguridad e integridad de los datos, es recomendable hacer uso de este INSERT. A diferencia del implícito, aquí le damos a Maple las columnas en las cuales se insertarán los valores, en orden respectivo.
Por ejemplo:

Input	Output
$Personas < Nombre, Apellido < 'Ron', 'Weasley';	INSERT INTO `Personas`(`Nombre`, `Apellido`) VALUES('Ron', 'Weasley');

Edición de datos (UPDATE)

Símbolo: <<

Para llevar a cabo la edición de datos, tomaremos una referencia de tabla, elegiremos la columna en la cual se editará el dato, y opcionalmente, le daremos un condicional.

Input	Ouput
$Facturas << Estado << 'Pagada';	UPDATE `Facturas` SET `Estado` = 'Pagada';
$Personas << Grupo << 'Adulto' ? edad > 18	UPDATE `Personas` SET `Grupo` = 'Adulto' WHERE edad > 18;

Si precisamos editar datos en más de una columna a la vez, usaremos la siguiente sintaxis. Los datos serán ingresados a sus respectivas columnas.

Input	Output
$Personas << Grupo, Ingreso_Permitido << 'Adulto', 'Sí' ? edad > 18	UPDATE `Personas` SET `Grupo` = 'Adulto', `Ingreso_Permitido` = 'Sí' WHERE edad > 18;

Modificación del esquema de base de datos

Maple ofrece funcionalidad y facilidades para la alteración del esquema de base de datos. Esto incluye la creación de tablas, edición de definición, inserción, y eliminación de columnas en tablas. Para que Maple sepa que haremos uso de operaciones que modifiquen el esquema, deberemos envolver toda referencia a tabla a modificar en una serie de paréntesis. Por ejemplo, si queremos alterar cualquier factor en la estructura de una tabla Personas, empezaremos nuestra sentencia con ($Personas).

Creación de tablas (CREATE TABLE)

Símbolo: +

Maple nos permite crear tablas haciendo uso de una sintaxis simple y fácil de recordar. La creación de tablas en Maple soporta la creación de primary keys por default, y uniones entre tablas, entre otras funciones que ya exploraremos.

Al crear una tabla, Maple automáticamente asume que esa tabla llevará una primary key. Esto significa que no será necesario especificarle a Maple que existirá una columna de nombre id, ya que esta misma ya vendrá incluida en la sintaxis generada de forma predeterminada. Incluso si no especificamos ninguna columna extra, Maple incluirá una columna ID automáticamente, siguiendo las buenas prácticas de SQL.

Input	Output
($Personas) +	CREATE TABLE `Personas` (`id` int NOT NULL AUTO_INCREMENT, PRIMARY KEY (`id`)) ENGINE=InnoDB ROW_FORMAT=COMPACT;

Como Podemos ver, la sentencia SQL resultante ya incluye una columna ID, automáticamente de tipo int, auto incrementable y convertida en primary key.

Importante: Maple no soporta la creación de tablas con más de una primary key, la utilización de otro ENGINE que no sea InnoDB, y otro ROW_FORMAT que no sea COMPACT.

Creación de tablas con columnas

Para especificar columnas que se crearán con la tabla, basta solo con escribir su definición luego del símbolo +.

Input	Output
($Personas) + nombre varchar(255), apellido varchar(255);	CREATE TABLE `Personas` (`id` int NOT NULL AUTO_INCREMENT, `nombre` varchar(255) NOT NULL, `apellido` varchar(255) NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB ROW_FORMAT=COMPACT;

Por default, las columnas creadas serán NOT NULL. Si deseamos que una de las columnas creadas acepte valores nulos, agregaremos un ? luego del nombre de la columna. Esto es similar al lenguaje de programación C#, el cual identifica a variables que aceptan valores nulos de la misma manera.

Por ejemplo, si deseamos que la columna teléfono acepte valores nulos:

Input	Output
($Personas) +	CREATE TABLE `Personas` (

nombre varchar(255), apellido varchar(255), telefono? varchar(255);	`id` int NOT NULL AUTO_INCREMENT, `nombre` varchar(255) NOT NULL, `apellido` varchar(255) NOT NULL, `telefono` varchar(255), PRIMARY KEY (`id`)) ENGINE=InnoDB ROW_FORMAT=COMPACT;

Asimismo, Maple permite asignarle un valor default a la columna, esto sigue la misma metodología que en MySQL.

Input	Output
($Personas) + nombre varchar(255), apellido varchar(255) DEFAULT 'Smith';	CREATE TABLE `Personas` (`id` int NOT NULL AUTO_INCREMENT, `nombre` varchar(255) NOT NULL, `apellido` varchar(255) DEFAULT 'Smith' NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB ROW_FORMAT=COMPACT;

En otras palabras, cualquier tipo de sintaxis válida en la declaración de columnas en una sentencia CREATE TABLE de MySQL es también legal en Maple.

Uniones con tablas

Símbolo <>

Es posible indicarle a Maple que genere una sentencia de creación de tabla e incluya una clave foránea que haga referencia a otra tabla. La tabla referenciada se escribirá precedida por un símbolo $ y luego de <>.

Una de las ventajas que conlleva esta metodología es que Maple automáticamente incluirá la creación de una columna de referencia en la tabla original. Esta nueva columna llevará el nombre de id[nombre_tabla_referenciada].

Para que la creación de esta unión sea efectiva, es necesario que la tabla referenciada tenga una columna llamada “id” y que ésta sea clave primaria.

Por ejemplo:

Input	Output
($Personas) + <> $Roles	CREATE TABLE `Personas` (`id` int NOT NULL AUTO_INCREMENT, `id_Roles` int(10) NOT NULL , PRIMARY KEY (`id`), CONSTRAINT `fk_Personas_Roles` FOREIGN KEY (`id_Roles`) REFERENCES `Roles` (`id`) ON DELETE CASCADE ON UPDATE CASCADE) ENGINE=InnoDB ROW_FORMAT=COMPACT;

Como podemos ver en la sentencia SQL resultante, Maple se encargó de crear una columna llamada “id_roles”, la cual es de tipo int y no acepta valores nulos.

También es posible incluir varias tablas luego de la primera unión.

Input	Output
($Empleado) + <> $Ciudad <> $Pais	CREATE TABLE `Empleado` (`id` int NOT NULL AUTO_INCREMENT, `id_Ciudad` int(10) NOT NULL , `id_Pais` int(10) NOT NULL , PRIMARY KEY (`id`), CONSTRAINT `fk_Empleado_Ciudad` FOREIGN KEY (`id_Ciudad`) REFERENCES `Ciudad` (`id`) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT `fk_Empleado_Pais` FOREIGN KEY (`id_Pais`) REFERENCES `Pais` (`id`) ON DELETE CASCADE ON UPDATE CASCADE) ENGINE=InnoDB ROW_FORMAT=COMPACT;

Importante: la creación de claves foráneas con Maple sólo soporta el modo CASCADA de eliminación y edición.

Modificación de definición de tablas (ALTER TABLE)

Maple ofrece la posibilidad de modificar la definición de las tablas. Esto quiere decir que es posible agregar, eliminar, y modificar columnas. Recordar que toda modificación a la definición de una tabla deberá ser declarada con los símbolos () envolviendo el nombre de la tabla.

Agregar columna (ADD COLUMN)

Símbolo <

Input	Output
($Personas) < nombre varchar(255)	ALTER TABLE `Personas` ADD COLUMN `nombre` varchar(255) NOT NULL ;

Por default, la columna que se agregará no aceptará valores nulos. Al igual que la sentencia CREATE TABLE, la sentencia ALTER TABLE es compatible con el símbolo ? para especificar la nulidad de la columna.

Por ejemplo:

Input	Output
($Personas) < nombre? varchar(255)	ALTER TABLE `Personas` ADD COLUMN `nombre` varchar(255);

Como podemos observar, la sentencia resultante creará una columna que aceptará valores nulos.

Editar columna (MODIFY COLUMN)

Símbolo <<

Input	Output
($Personas) << nombre varchar(255)	ALTER TABLE `Personas` MODIFY COLUMN `nombre` varchar(255) NOT NULL ;
($Personas) << nombre? varchar(255)	ALTER TABLE `Personas` MODIFY COLUMN `nombre` varchar(255);

Importante: la versión actual de Maple no soporta el renombramiento de columnas, ya que internamente hace uso de MODIFY COLUMN y no CHANGE COLUMN.

Eliminar columna (DROP COLUMN)

Símbolo [x]

Para eliminar una columna, sólo basta con especificar el nombre de la columna que queremos eliminar. No es necesario especificar tipo de dato ni nulidad.

Input	Output
($Personas) [x] nombre	ALTER TABLE `Personas` DROP COLUMN `nombre`;

Funciones especiales de Maple

Maple cuenta con funciones nativas que son procesadas por el parseador de forma automática. Hasta ahora, estas funciones sirven para facilitar la escritura de sentencias Maple, haciéndolas más acortadas.

Función t

Parámetro: un número.

La función t hace referencia a un nombre de tabla mencionado anteriormente en la sentencia. Esta función toma como parámetro un número. Si nuestra sentencia contiene varias tablas, el número sirve para identificar qué tabla queremos referenciar.

Por ejemplo:

Input	Output
$Personas > <> $Roles ? t(1).id = '1'	SELECT * FROM `Personas` INNER JOIN `Roles` ON Personas.id_Roles = Roles.id WHERE Personas.id = '1';

Como se puede ver, la función t(1) nos devolvió el nombre “Personas”, ya que fue la primera tabla que mencionamos. Si hubiésemos querido hacer referencia a la tabla Roles, hubiésemos usado el número 2 como parámetro. Al ser t(1) una referencia actual al nombre de la tabla, podemos concatenarle la columna que queremos de esa tabla.

Esta función es muy útil para sentencias con INNER JOIN y condicionales, en donde es necesario especificar el nombre de la tabla con las columnas para evitar ambigüedades entre tablas.

La función t es aceptada en cualquier parte de la sentencia Maple, siempre y cuando el número dado como parámetro sea compatible con el número de tablas dado en la sentencia.

Por ejemplo, es posible utilizar la función t para especificar tablas con columnas en un SELECT.

Input	Output
$Personas > t(1).nombre, t(1).apellido, t(2).descripcion <> $Roles	SELECT `Personas`.`nombre`, `Personas`.`apellido`, `Roles`.`descripcion` FROM `Personas` INNER JOIN `Roles` ON Personas.id_Roles = Roles.id;

Función c

Parámetro: un número.

La función c es idéntica a la función t, pero con la diferencia que ésta hace referencia a columnas escritas en la sentencia.

Input	Output
$Personas > nombre ? c(1) = 'John'	SELECT `nombre` FROM `Personas` WHERE nombre = 'John';

En este caso, la columna 'nombre' fue la primera columna, entonces, pasándole a la función c el número 1, obtenemos como resultado 'nombre'.
Importante: la función c no es recomendada para columnas afectadas por funciones SQL nativas. Por ejemplo:

Input	Output
$Personas > AVG(sueldo) ? c(1) > 4500	SELECT AVG(sueldo) FROM `Personas` WHERE AVG(sueldo) > 4500;

Como se ve, el conversor no pudo determinar el nombre exacto de la columna debido a que está envuelta por una función AVG de SQL. Este comportamiento es intencional, debido a que dentro de una función puede haber funciones anidadas u operaciones aritméticas.

Utilización de funciones t y c en una misma sentencia

Ambas funciones pueden utilizarse en una misma sentencia Maple. Por ejemplo, es posible "encadenar" ambas funciones para llegar a un resultado [tabla].[columna]. Las encadenaciones siempre se deben hacer utilizando un punto.

Input	Output
$Personas > nombre, apellido ? t(1).c(1) = 'David'	SELECT `nombre`, `apellido` FROM `Personas` WHERE Personas.nombre = 'David';

O si se desea, se pueden encadenar estas funciones con nombres explícitos de columnas o tablas.

Input	Output
$Personas > nombre, apellido ? Personas.c(1) = 'John'	SELECT `nombre`, `apellido` FROM `Personas` WHERE Personas.nombre = 'John';
$Personas > nombre, apellido ? t(1).nombre = 'Harry'	SELECT `nombre`, `apellido` FROM `Personas` WHERE Personas.nombre = 'Harry';

Para tener en cuenta:
La función c toma el nombre de la columna tal y como la escribimos. Si la misma es escrita con el nombre de la tabla precediéndola, ya sea por uso de la función t o nombre explícito, la referencia devuelta por la función c también tendrá el nombre de la tabla.

Input	Output
$Personas > Personas.nombre ? c(1) = 'John'	SELECT `Personas`.`nombre` FROM `Personas` WHERE Personas.nombre = 'John';
$Personas > t(1).nombre ? c(1) = 'John'	SELECT `Personas`.`nombre` FROM `Personas` WHERE Personas.nombre = 'John';

Embeber SQL en sentencias Maple

Como hemos visto, no es posible llevar a cabo todas las operaciones que se pueden hacer con SQL con Maple.

Maple ofrece una manera de realizar operaciones complejas con SQL sin tener que abandonar el dialecto por completo. Esto quiere decir que podemos escribir sentencias SQL y Maple y enviar ambas al conversor al mismo tiempo. Esto se logra envolviendo cualquier sentencia escrita en SQL con etiquetas <? ?>.
Las etiquetas <? ?> le indican al conversor que ignore todo lo que está envuelto por ellas y que continúe con la sentencia que sigue.

Input	Output
<? SELECT * FROM Personas ?>	SELECT * FROM Personas;
<? SELECT * FROM Personas ?> $Roles;	SELECT * FROM Personas; SELECT * FROM `Roles`;

A continuación, se mostrará un ejemplo más complejo, en donde se crea una nueva tabla con SQL, se agregan datos mediante Maple, y finalmente se mostrarán sus contenidos mediante un SELECT en SQL.

Input	Output
<? CREATE TABLE `Personas` (`id` int NOT NULL AUTO_INCREMENT, `nombre` varchar(255) NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB ROW_FORMAT=COMPACT; ?> $Personas < 1, 'John'; <? SELECT * FROM Personas ?>	CREATE TABLE `Personas` (`id` int NOT NULL AUTO_INCREMENT, `nombre` varchar(255) NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB ROW_FORMAT=COMPACT; INSERT INTO `Personas` VALUES(1, 'John'); SELECT * FROM Personas;

Las operaciones pueden cambiarse, en el caso siguiente, crearemos una nueva tabla con Maple, agregaremos datos mediante SQL, y finalmente mostraremos los contenidos mediante un SELECT de Maple.

Input	Output
($Personas) + nombre varchar(255); <? INSERT INTO Personas VALUES(1, 'John'); ?> $Personas	CREATE TABLE `Personas` (`id` int NOT NULL AUTO_INCREMENT, `nombre` varchar(255) NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB ROW_FORMAT=COMPACT; INSERT INTO Personas VALUES(1, 'John'); SELECT * FROM `Personas`;

Múltiples sentencias SQL dentro de un par de etiquetas

Es posible incluir múltiples sentencias SQL dentro de un par de etiquetas <? ?>.
Por ejemplo, si deseamos crear una tabla exclusivamente con SQL y luego insertar un set de datos con SQL, podemos ahorrarnos un par de etiquetas <? ?> e incluir todas las sentencias necesarias para llevar a cabo esta tarea en un par. El único requerimiento es que todas las sentencias SQL dentro de las etiquetas deberán estar separadas por el símbolo ‘;’ como en cualquier SQL válido.

Input	Output
<? CREATE TABLE `Personas` (	CREATE TABLE `Personas` (`id` int NOT NULL AUTO_INCREMENT,

`id` int NOT NULL AUTO_INCREMENT, PRIMARY KEY (`id`)) ENGINE=InnoDB ROW_FORMAT=COMPACT; INSERT INTO Personas VALUES(1, 'John'); ?>	PRIMARY KEY (`id`)) ENGINE=InnoDB ROW_FORMAT=COMPACT; INSERT INTO Personas VALUES(1, 'John');

Al embeber SQL en Maple, se ahorra el paso de ejecutar una sentencia SQL aparte y luego hacer las operaciones necesarias con Maple. Al hacerlo de esta manera, nos aseguramos de que todas las operaciones se ejecuten una detrás de la otra, sin importar su complejidad.

Es importante recordar que, aunque se envíen los dos tipos de sentencias al motor, ambas no se pueden mezclar en una misma sentencia. Esto se explicará en más detalle en la sección "Limitaciones".

Limitaciones

SQL dentro de una sentencia Maple

El uso de los símbolos <? ?> actúa como una sentencia individual. Esto significa que no es posible incluir estos símbolos dentro de una sentencia Maple individual. Es decir, no estaría permitido por ejemplo llevar a cabo un SELECT utilizando sintaxis de Maple y escribir los condicionales con sintaxis SQL. Esto resultará en comportamiento no definido e indeseado.
Por ejemplo:

Input	Output
$Personas > nombre, apellido <? WHERE nombre = 'John' ?> [1000]	Una sentencia SQL sintácticamente inválida

Maple dentro de una sentencia SQL

Asimismo, incluir Maple envuelto por símbolos <? ?> es inválido y resultará en comportamiento no definido.
Por ejemplo:

Input	Output
<? SELECT * FROM Personas > <> $Roles ?>	Una sentencia SQL sintácticamente inválida

En otras palabras, todo lo que esté envuelto por símbolos <? ?> debe ser SQL válido.

Comentarios

Símbolo: --

Maple acepta comentarios precedidos por "-- ". Estos comentarios serán removidos por el conversor y no aparecerán en la sentencia SQL resultante.

Input	Output
$Personas; -- este es un comentario	SELECT * FROM `Personas`;

Importante: Maple sólo soporta este tipo de comentarios. Comentarios que pueden ser válidos en SQL como /* */ no son permitidos y resultarán en comportamiento no definido.

Es posible incluir comentarios en SQL embebido. Estos comentarios, al igual que en las sentencias Maple, serán removidos por el conversor.

Input	Output
<? SELECT * FROM Personas -- este es un comentario ?>	SELECT * FROM Personas;

Importante: comentarios estilo /* */ son permitidos siempre y cuando estén en una sentencia SQL dentro de los símbolos <? ?>