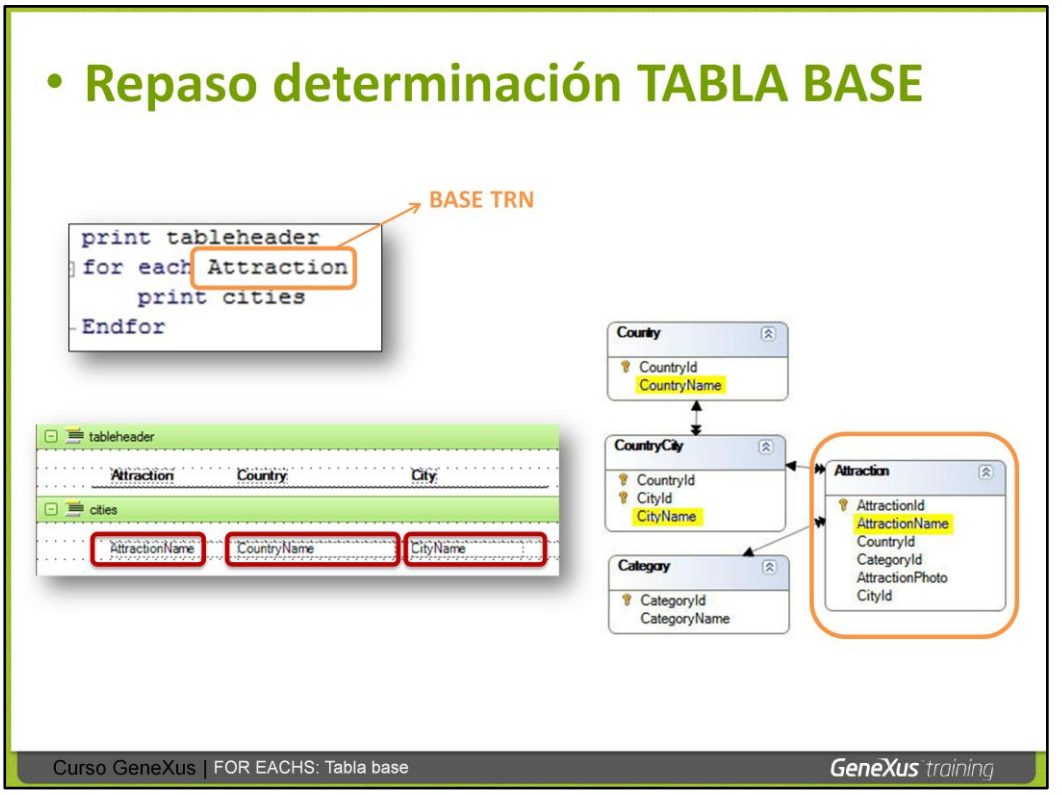
FOR EACHS

Más conocimientos
sobre For each simple,
índices, cláusulas, etc.

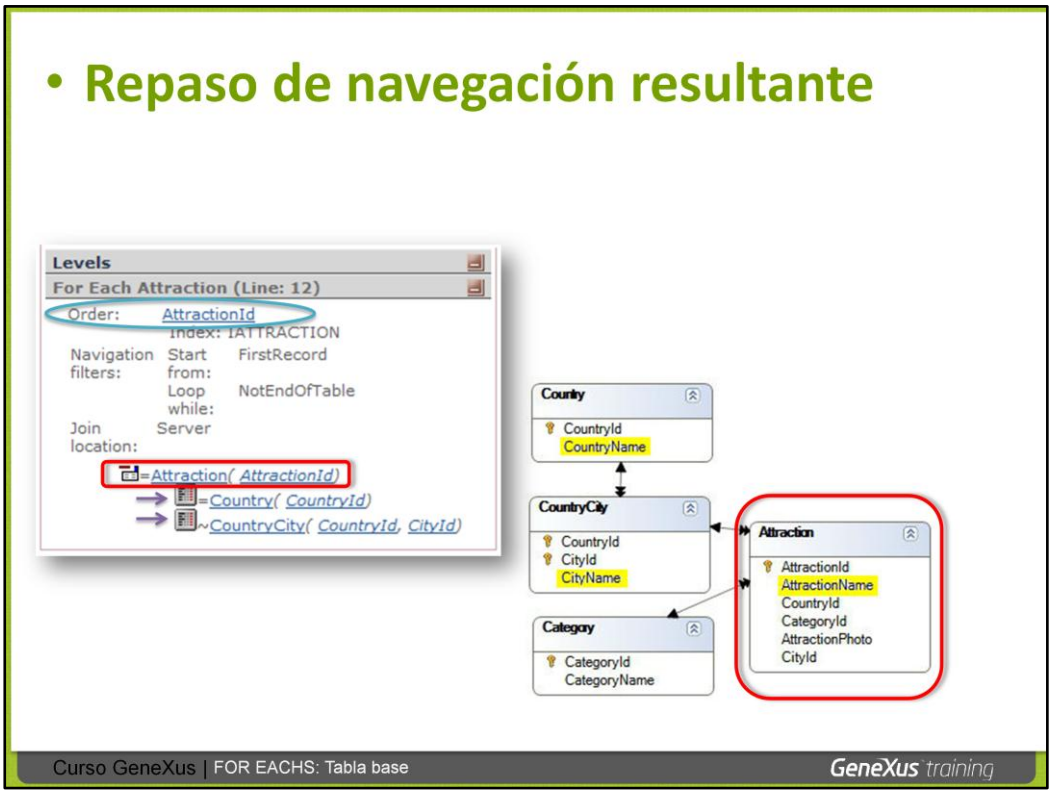
TABLA BASE



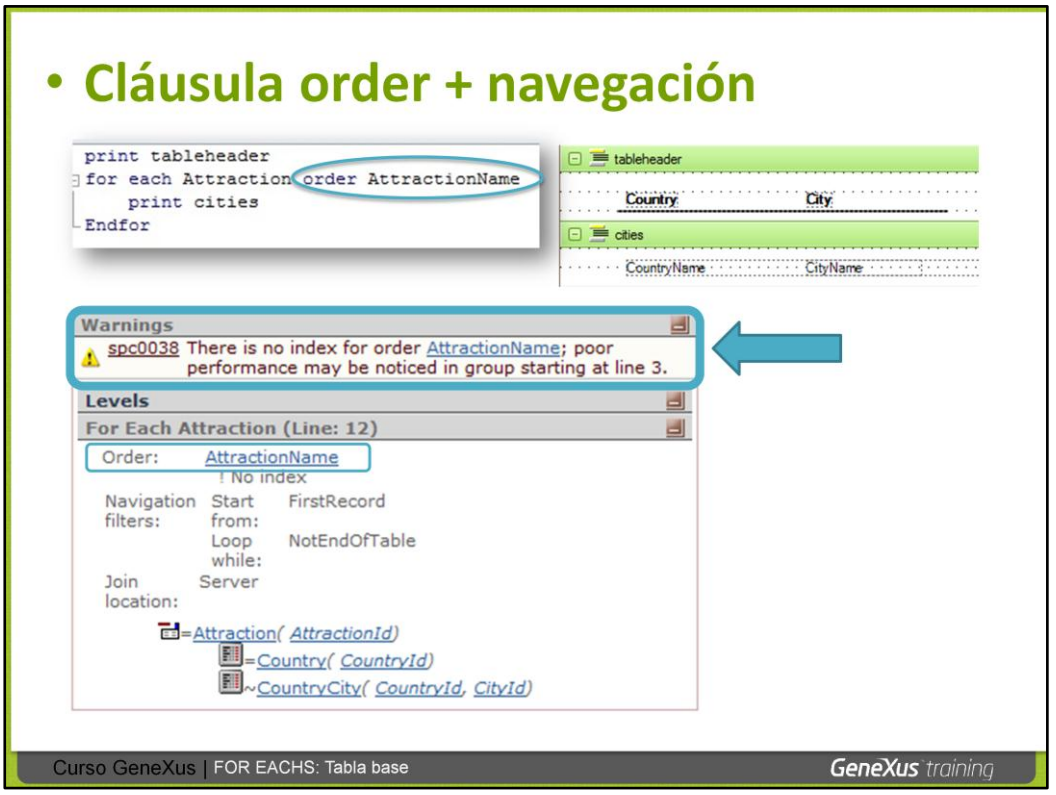
Recordemos que GeneXus determina la tabla base del for each teniendo en cuenta el nombre de la transacción que declaramos al lado del for each (que debe ser la transacción cuya tabla física asociada queremos recorrer).

Además, los atributos declarados en el cuerpo del for each (printblocks, where, order, etc.), deben pertenecer a la tabla base o extendida de la tabla base navegada.

En el ejemplo presentado en la diapositiva, la tabla base del for each será ATTRACTION, o sea la tabla que se recorrerá y se accederá a su tabla extendida para acceder a los datos requeridos.



El listado de navegación nos informa claramente que la tabla base es ATTRACTION, que la recorrida será ordenada por la clave primaria de dicha tabla: AttractionId, y se que recorrerá toda la tabla, accediendo tanto a la tabla COUNTRY (para recuperar CountryName, el país de la atracción), como a COUNTRYCITY para recuperar a CityName.



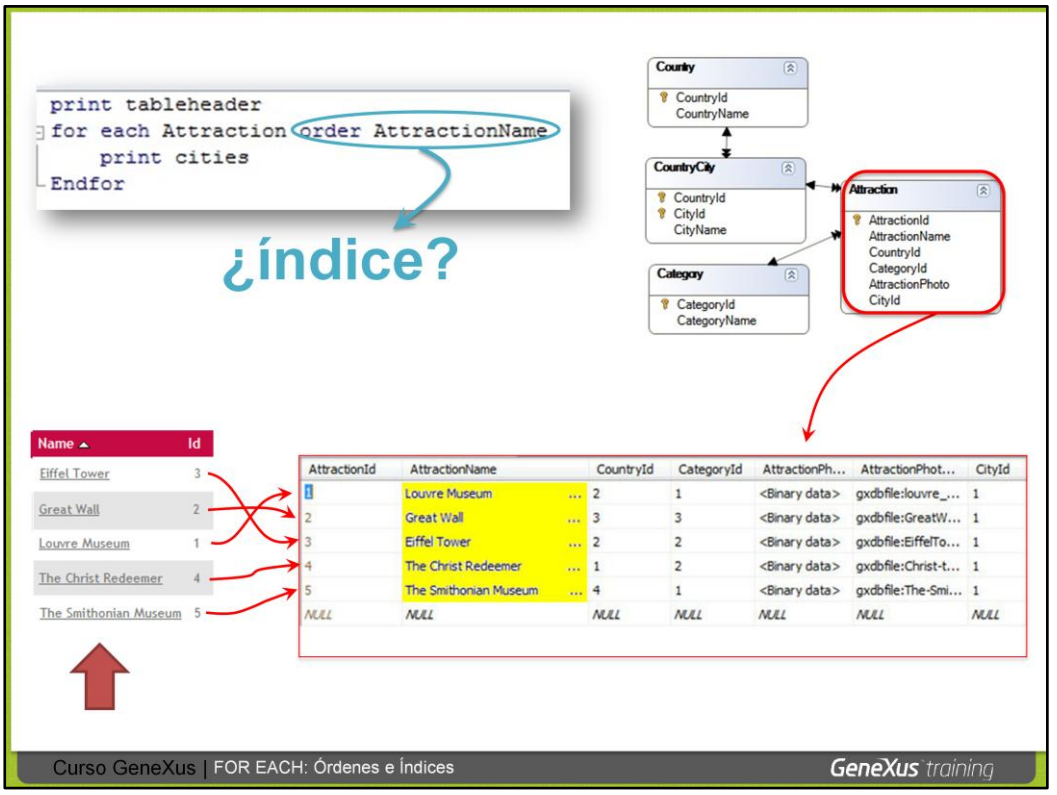
¿Qué sucedería si en vez de agregar el atributo AttractionName al printblock, lo agregáramos en la cláusula order?

En este caso, GeneXus ordenará la recorrida por **AttractionName**, y para cada registro navegado, accederá a la tabla COUNTRY a través del valor de la clave foránea: CountryId, para recuperar el valor de CountryName. Lo mismo hará con la tabla física: COUNTRYCITY, para recuperar el valor de CityName.

En el listado pdf resultante, ahora se imprimirán sólo el país y ciudad de cada atracción (pues no colocamos AttractionName en el printblock).

Además, el listado de navegación nos informa que en la base de datos no existe un **índice** por el atributo por el que necesitamos ordenar la información, por lo que podríamos tener baja performance para esta consulta. ¿Por qué?

ÓRDENES
E
ÍNDICES



Supongamos que la tabla ATTRACTION tiene los datos que se muestran. Si necesitamos obtener sus registros ordenados por el atributo AttractionName, entonces tendrán que reordenarse los registros de algún modo por ese atributo.

Recordemos que los **índices** son vías de acceso eficientes a los datos. Podemos pensar por ejemplo, en un libro de cocina con muchas páginas que contienen recetas, el cual tiene varios índices (índice alfabético, índice por tipos de comidas, etc.). De igual forma, las tablas que almacenan registros, tienen índices también.

GeneXus al crear tablas físicas, crea para las mismas 1 índice por el atributo primario de la tabla (es decir por su clave primaria sea simple o compuesta) y 1 índice por cada clave foránea.

Cuando se define una consulta, si hay un índice físico creado en la tabla, por el atributo a ordenar, GeneXus lo usará. Pero en este caso la consulta se necesita ordenada por un atributo secundario: AttractionName. Y GeneXus nos advierte en el listado de navegación asociado al objeto, que que no hay un índice físico definido.

La existencia del índice físico optimizaría la consulta. Pero la desventaja de crear un índice, es que, a partir de allí, debe ser mantenido. Es decir, si los usuarios van agregando, modificando o eliminando atracciones en la tabla ATTRACTION, debe reacomodarse el índice (o sea, los punteros del índice deben reacomodarse de forma tal de tener incluidas las nuevas atracciones, donde correspond,a para mantener el orden).

Según el DBMS, la cantidad de registros de la tabla en cuestión y la frecuencia con la cual se suele solicitar la consulta, el analista deberá decidir si crear cada índice por el cual necesite ordenar consultas, o dejar que en algunos casos el DBMS cree índices temporales en tiempo de ejecución y los use. Puede ser una buena opción en algunos casos usar índices temporales para evitar el mantenimiento de demasiados índices. Eso lo debe analizar y decidir el analista GeneXus, según el caso.

Definir un índice para una tabla de la base de datos es sencillo y puede hacerse en cualquier momento.

¿Cómo? Buscamos la tabla, la abrimos y vamos a la sección relacionada a los índices definidos.

Los tres primeros, que aparecen antecedidos por el prefijo "I", son los creados automáticamente por GeneXus a partir de las claves primaria y foráneas.

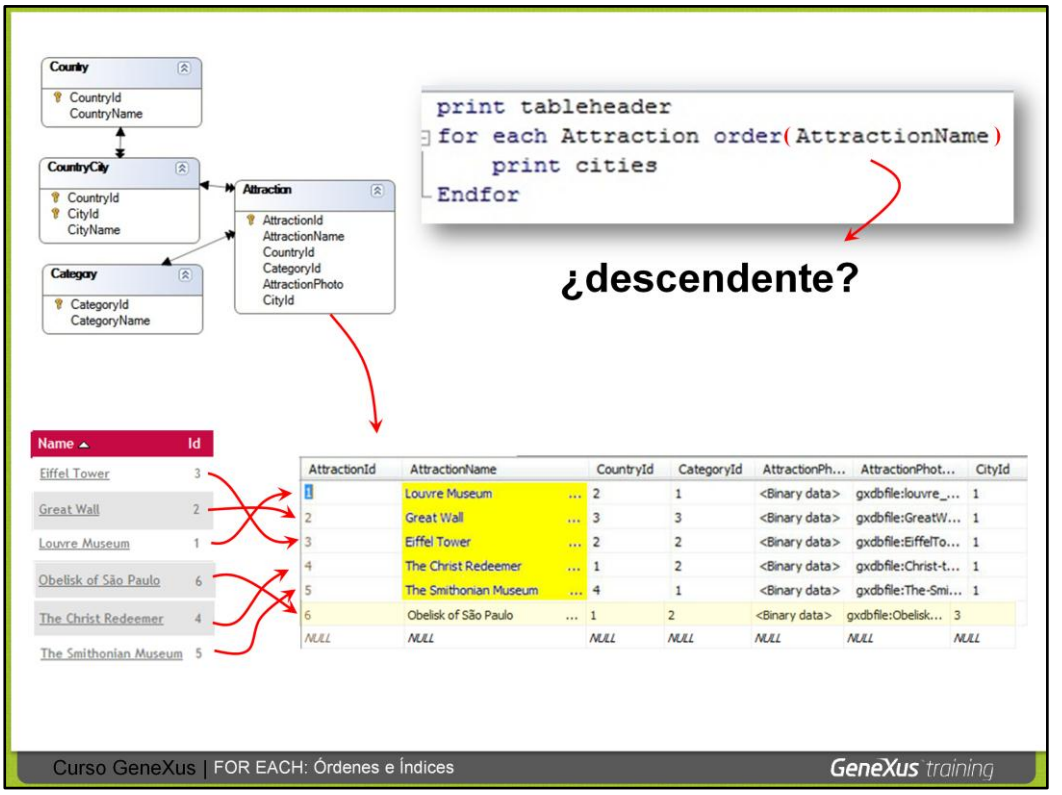
Necesitamos crear uno de usuario. Para ello presionamos enter, tras lo que aparecerá el nombre por defecto UAttraction. Lo modificamos a nuestro gusto (agregándole Name al final, por ejemplo). El prefijo "U" es por **U**ser.

Deseamos que este índice esté compuesto por el atributo AttractionName, ordenado en sentido ascendente.

Si fuera un requisito que los nombres de atracciones no pudieran repetirse, podemos controlarlo indicando que el índice sea **Unique**, y no **Duplicate**. Si definimos para un índice que sea **Unique**, se controlará **automáticamente** cuando se ingrese una atracción (o modifique su nombre), que no exista otra con el mismo nombre –utilizando este índice–. En nuestro ejemplo los nombres pueden repetirse (por ejemplo pensemos que cada país suele tener un Obelisco), así que para este índice por AttractionName, dejamos el valor: Duplicate.

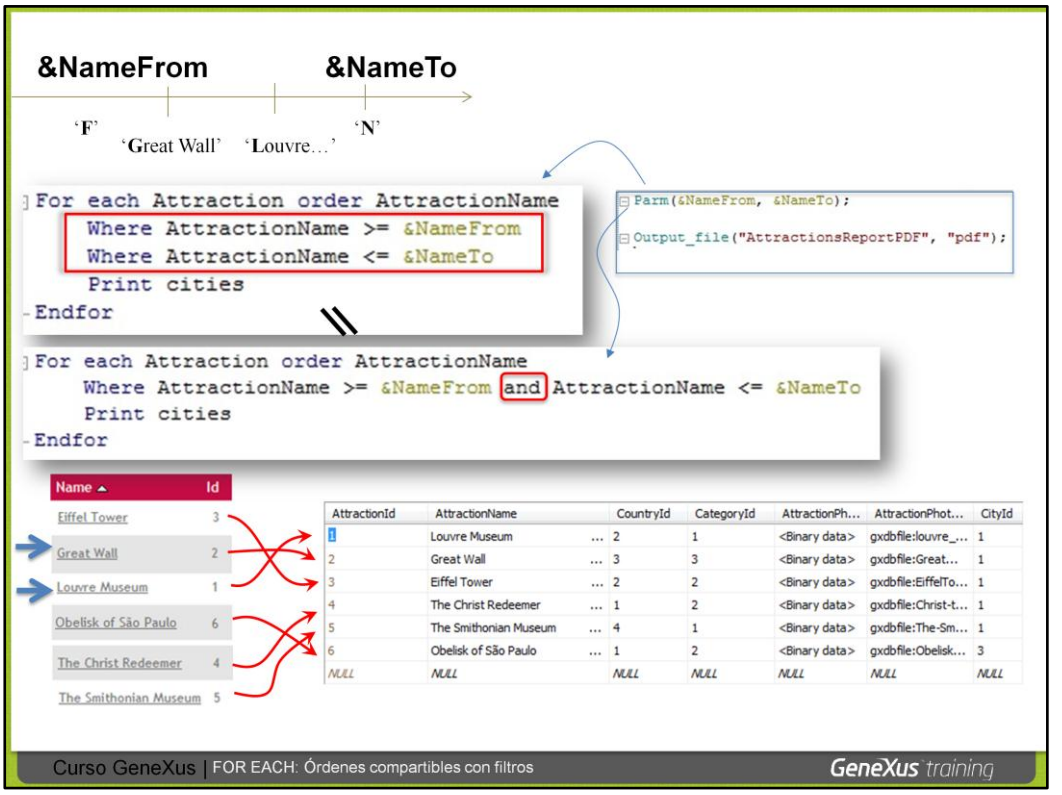
Una vez hecho esto, al dar F5 deberá reorganizarse la base de datos, para crear el nuevo índice. Y el listado de navegación asociado a nuestro procedimiento que efectúa la consulta ordenada por AttractionName, nos informa que utilizará el índice que se acaba de crear.

Así como lo creamos, en cualquier momento podemos eliminarlo, y al hacer F5 y reorganizar, volveremos a la situación de la que habíamos partido antes de crearlo.



¿Cómo hacemos para solicitar un orden descendente? Simplemente rodeando de paréntesis curvos al atributo.

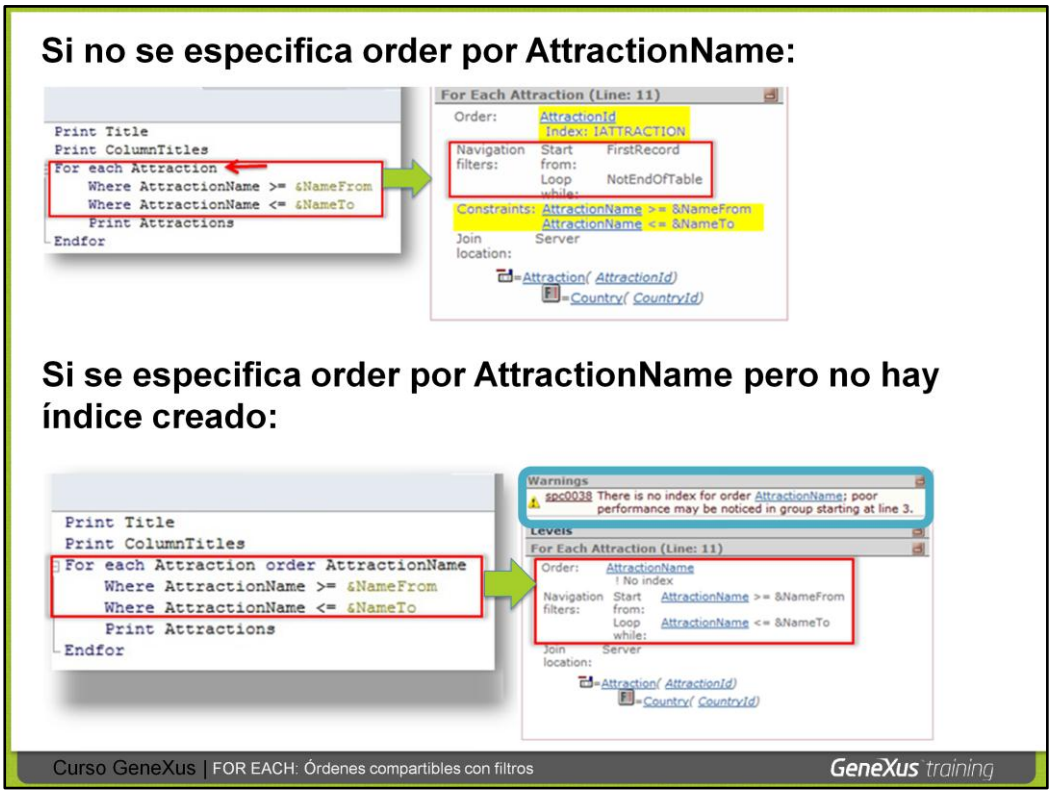
VENTAJA DE ÓRDENES COMPATIBLES CON FILTROS



Supongamos que lo que nos interesa es obtener un listado de las atracciones cuyos nombres estén alfabéticamente entre un par de valores recibidos por parámetro. Por ejemplo, entre la "F" y la "N".

Para eso especificamos las cláusulas where que se ven arriba.

Tener varias cláusulas where es equivalente a tener una sola, donde las condiciones se conjugan con el operador lógico "and". Es decir, se considerarán sólo los registros que cumplan con todas las condiciones **a la vez**.

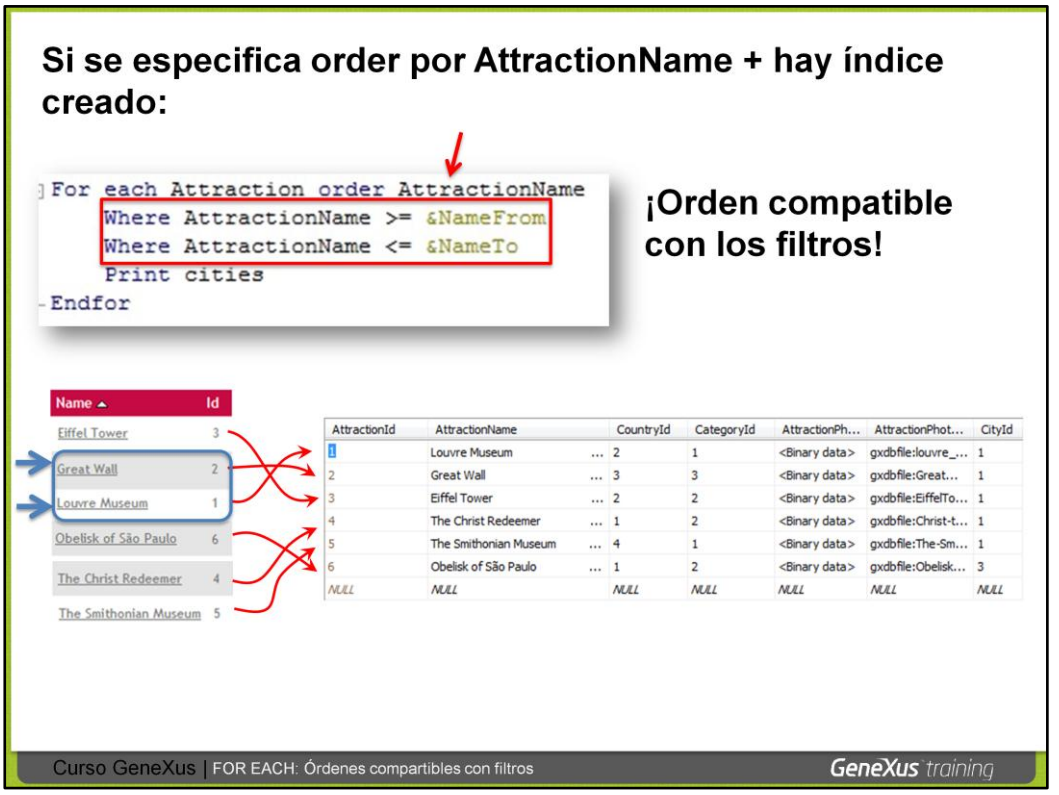


Observemos lo que sucede si no se especifica **order**. La información mostrada será la misma, pero… al no especificar criterio de ordenamiento, GeneXus elige como order el índice por la clave primaria de la tabla, y haciendo esto, tendrá que recorrer toda la tabla para quedarse con los registros que cumplen con las restricciones. El primer registro podría caer dentro del rango, algunos en el medio sí, otros no y el último también caer dentro del rango. ¡Este listado no está optimizado!

Si en cambio especificamos **order AttractionName**, hay 2 casos posibles:

1) Que el índice físico no exista
2) Que el índice físico se defina como índice de usuario

El caso 1) ya lo hemos explicado en páginas anteriores. En la siguiente página veremos el caso 2).



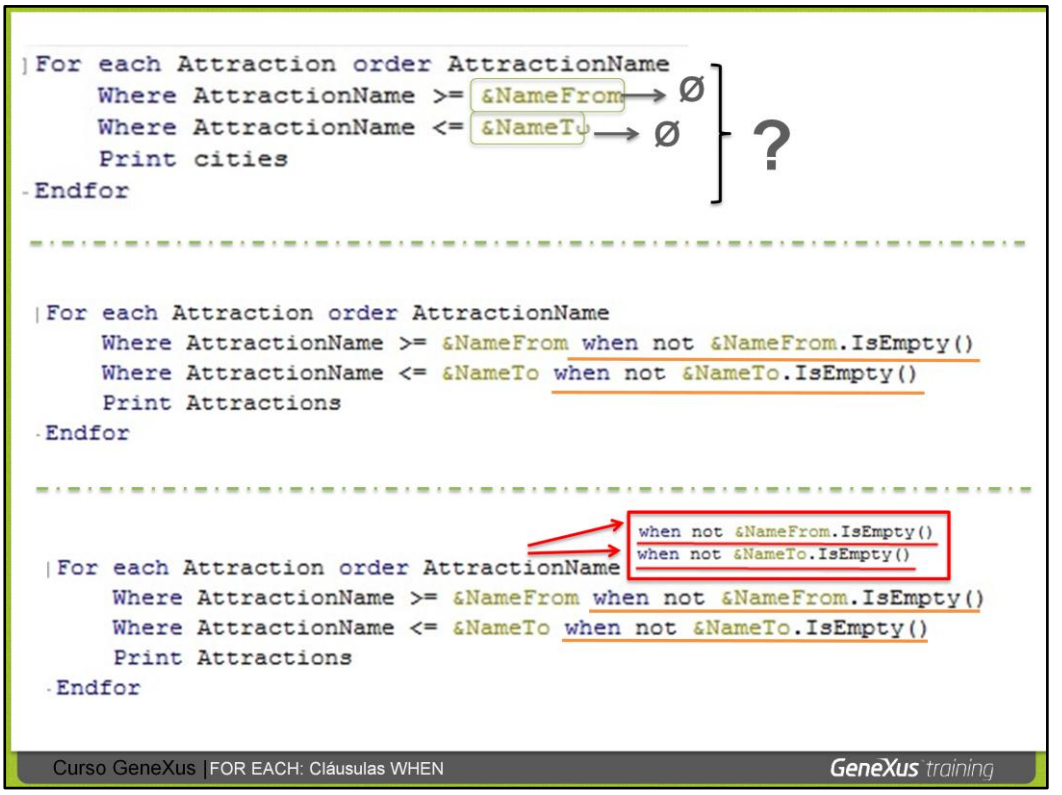
Si se ordena la consulta por AttractionName y el índice por dicho atributo se crea para la tabla física ATTRACTION, GeneXus determinará usar dicho índice **y el listado de navegación nos informará que al ordenar por AttractionName, no será necesario recorrer toda la tabla base.**

Imaginemos que estamos buscando una palabra que empieza con "G" en un diccionario en papel. No recorreremos todo el diccionario de principio a fin, ¡pues está ordenado! Aquí es igual.

A los efectos de determinar la tabla base, da igual que esté o no la cláusula **order AttractionName**, dado que el atributo AttractionName aparece de todos modos en las cláusulas where y está especificada la trn base al lado del For each.

De ser posible, se sugiere buscar ordenar por criterios compatibles con los filtros para optimizar las recorridas (si el índice está creado, más aún, ¡sin duda alguna! y si no es el caso, igual los DBMSs hoy resuelven con eficiencia la creación de índices temporales).

CLÁUSULAS WHEN PARA ÓRDENES Y FILTROS



¿Qué resultado se obtendrá para el for each de arriba si las variables &NameFrom y &NameTo están vacías? Si existiera una atracción con nombre vacío, será la única devuelta, pues será la única que cumplirá ambas condiciones. En caso contrario, ninguna atracción será listada.

¿Es posible condicionar los ordenamientos y los filtros, para que sólo se apliquen ante determinadas circunstancias? Por ejemplo, que sólo se aplique el primer where **cuando** la variable &NameFrom no esté vacía. Y que sólo se aplique el segundo where **cuando** la variable &NemeTo no esté vacía. La respuesta es sí. Lo conseguimos condicionando las cláusulas where con **when**, como vemos en el segundo for each. Sólo se aplicarán los where, cuando la condición se satisfaga. Así, en ejecución, cuando dejemos ambas variables vacías, no se aplicará ninguno de los where, por lo que saldrán listadas todas las atracciones de la tabla.

De la misma manera puede condicionarse la aplicación o no de un order, como mostramos en el tercer for each. De hecho puede especificarse una sucesión de órdenes condicionados, de manera que el primero cuya condición se satisfaga sea el elegido.

Vea más de órdenes y filtros en el wiki de GeneXus

(ie: http://wiki.gxtechnical.com/commwiki/servlet/hwikibypageid?6075).

WHEN NONE

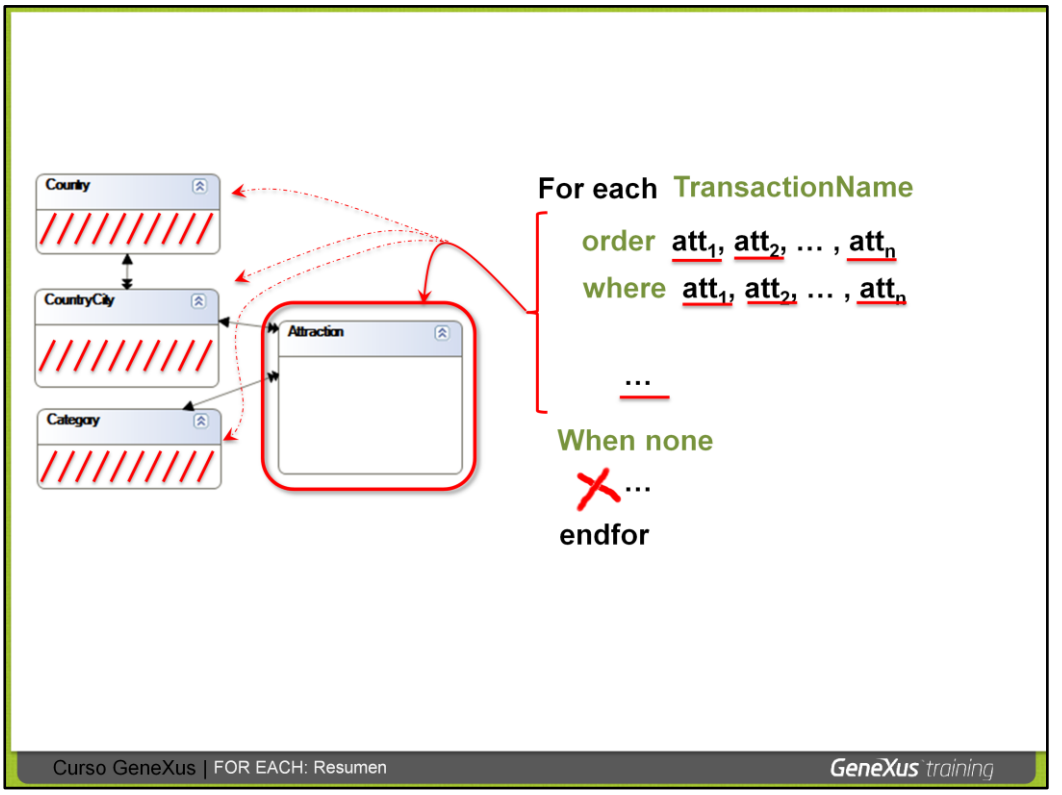
¿Qué pasa cuando ninguno de los registros de la tabla base cumple con las condiciones?

Supongamos que queremos en ese caso imprimir en la salida un mensaje que lo advierta… para eso programamos la cláusula **when none**.

Todos los comandos que se escriban entre el when none y el endfor, se ejecutarán secuencialmente y **en el único caso en que no se hayan encontrado registros de la tabla base del for each que cumplieran las condiciones.**

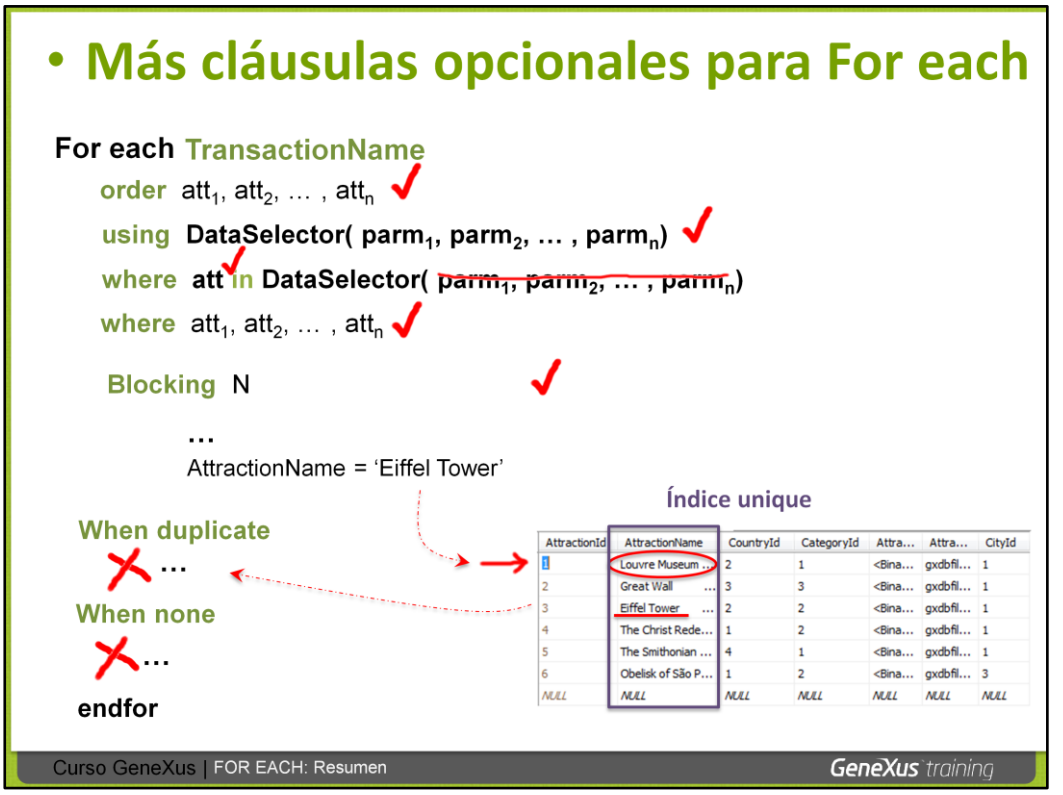
En nuestro caso hemos definido imprimir un mensaje, pero se podría definir un for each, new, o lo que corresponda hacer según el caso.

RESUMEN



Como ya hemos visto, la tabla base de un For each se determinada teniendo en cuenta la transacción mencionada al lado del for each (Base Trn) y el resto de los atributos mencionados, tanto en el cuerpo del For each como en las cláusulas Order, Where deberán pertenecer a su tabla extendida.

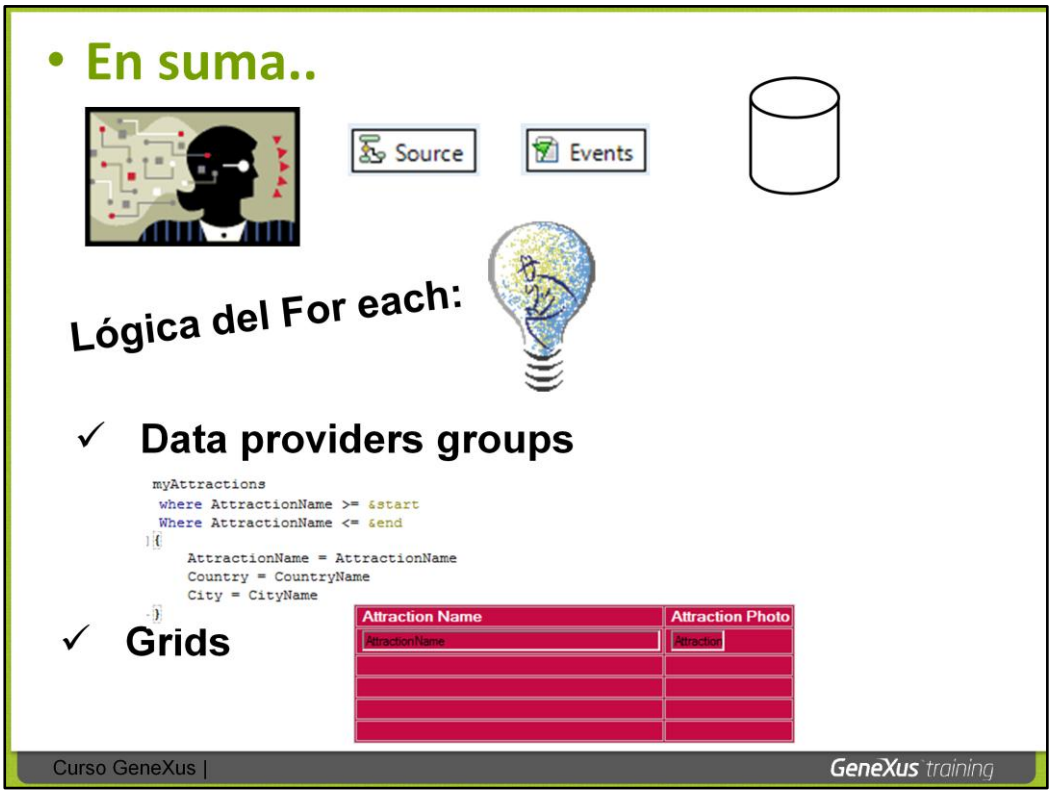
Los atributos que quedan exonerados son los mencionados en el bloque When none.



El for each acepta más cláusulas opcionales. Por ejemplo, existe otra manera de filtrar la información con la que se desea trabajar, y es a través del uso de **Data Selectors**. Este tema se ve en otro conjunto de slides. Y si se desea ver más sobre este objeto en nuestro community wiki: http://wiki.gxtechnical.com/commwiki/servlet/hwikibypageid?5271.

Otra cláusula opcional que se agrega cuando el comando for each se está ejecutando dentro de un procedimiento, para actualizar o eliminar registros de la base de datos, es la cláusula **Blocking**, que permite especificar que la actualización o eliminación se realice en bloques de N registros, reduciendo así los accesos a la base de datos.

Cuando el for each se encuentra dentro de un procedimiento, y se está queriendo actualizar el valor de un atributo que no puede tener valores repetidos, por ejemplo supongamos que AttractionName tiene definido un índice Unique, y se está ejecutando el for each sobre el registro 1… Al intentar cambiarle el valor a AttracionName de ese registro, para que asuma el valor "Eiffel Tower" el índice Unique indicará que ya existe un registro con ese valor y la actualización no se realizará. Pero si queremos tomar alguna acción ante ese caso, programamos la cláusula **when duplicate**…



El for each puede utilizarse tanto en procedimientos, como en eventos de otros objetos, en los que esté permitido consultar la base de datos.

La misma lógica del for each se encuentra en otras formas de consulta de la base de datos, como:

- Los grupos de Data Providers, y
- los grids con tabla base

Por lo que haber aprendido esta lógica, significará haber aprendido buena parte del corazón de GeneXus.

GeneXus training

training.genexus.com