Ejercicios prácticos

GeneXus™ 16

Marzo 2019

CONTENIDO

CONTENIDO 2

EL PROBLEMA 4

NUEVO PROYECTO, NUEVA BASE DE CONOCIMIENTO 4

PRIMERAS TRANSACCIONES 5

Transacción "Employee" 5

Transacciones "AmusementPARK" y "country", relacionadas 7

Datos relacionados: ¿cómo se mantiene la integridad? 10

Transacción 'Game' 11

Transacción 'Category' 11

Transacciones "Employee" y "AmusementPark", relacionadas 13

Agreguemos las ciudades a la transacción 'Country' 14

Transacción "AmusementPark": agreguemos la ciudad. 16

AGREGUEMOS COMPORTAMIENTO A LAS TRANSACCIONES (RULES) 17

PATTERNS: MEJORANDO LA INTERFAZ PARA TRABAJAR CON LA INFORMACIÓN 18

TRANSACCIONES "REPAIR" Y "TECHNICIAN" Y NECESIDAD DE DEFINIR SUBTIPOS 22

FÓRMULAS 23

CREACIÓN DE SEGUNDO NIVEL 24

LISTADOS PDF 26

PASAJE DE PARÁMETROS 32

Listado de parques en un rango determinado 32

BUSINESS COMPONENTS 33

Aumento de precio de las reparaciones 33

Pantalla para eliminación de todas las reparaciones 35

PROCEDIMIENTOS PARA ACTUALIZAR REGISTROS 36

Aumento de precios de las reparaciones 36

Eliminación de todas las reparaciones 36

Inicialización de la información de la base de datos [opcional] 40

WEB PANELS 42

EXTENDED CONTROLS 42

OBJETO QUERY 43

PARTE PARA SMART DEVICES 44

GENEXUS SERVER 45

EL PROBLEMA

Una multinacional encargada de gestionar parques de diversiones lo contrata para que desarrolle un sistema para almacenar y manipular la información con la que trabaja. Imagine que el sistema se compone de dos módulos:

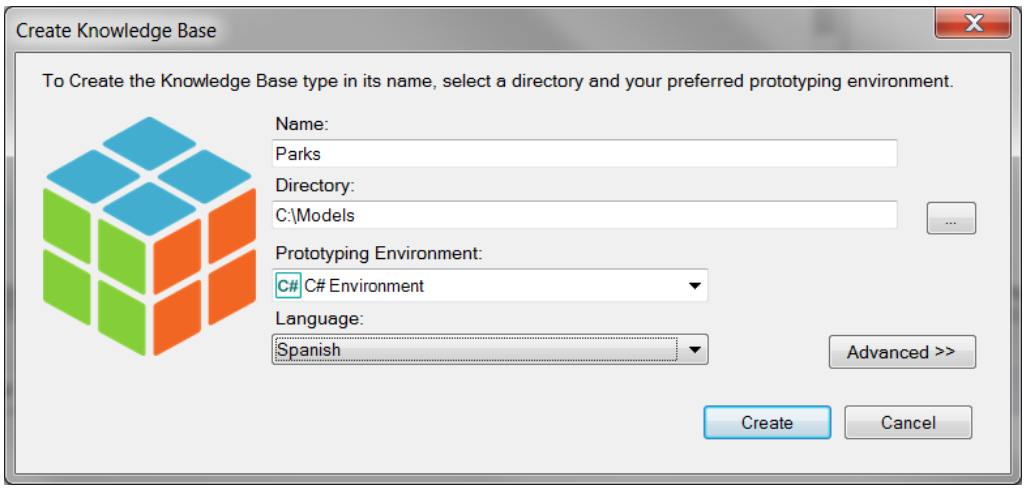
- **Backend**: parte de la aplicación que deberá correr en un servidor web, de manera tal que los empleados de la empresa puedan manipular la información desde cualquier lugar con conexión a internet.
- **Aplicación sencilla para dispositivos móviles**: parte de la aplicación que será destinada para ser descargada por los clientes de la empresa, la cual les permitirá consultar los países disponibles, así como los principales parques de diversiones que ofrece cada ciudad y sus juegos.

NUEVO PROYECTO, NUEVA BASE DE CONOCIMIENTO

Entrar a GeneXus y crear una base de conocimiento de nombre *Parks* para comenzar el desarrollo de la aplicación.

Sugerimos:

- Elegir como ambiente de desarrollo C#. Asegúrese de tener instalado todo lo necesario (incluyendo SQL Server). Si usa GeneXus Trial, el ambiente de generación con C# y SQL Server ya es predefinido, prototipando en la nube de Amazon.
- No crear la base de conocimiento en la carpeta "Mis Documentos" o cualquier otra carpeta que quede bajo "Documents and Settings", debido a que estas carpetas tienen permisos especiales otorgados por Windows.



Tómese unos minutos para **familiarizarse** con el **IDE (ambiente de desarrollo integrado de GeneXus)**. Pruebe **mover ventanas**, **visualizar ventanas específicas que desee** (View y View/Other Tool Windows) y observe detenidamente el contenido de la ventana **KBExplorer** (Knowledge Base Explorer). Verá que aparecen ya inicializados **dominios**, algunos **objetos**, **imágenes**, etc.

Sugerencia: mantenga la ventana de propiedades abierta (**F4**), pues la utilizará continuamente. Dentro de la ventana **'Preferences'** donde se configura el **'Environment'**.

PRIMERAS TRANSACCIONES

En las reuniones con la empresa, le transmiten lo siguiente:

> "Nosotros registramos los datos de parques de diversiones, para realizar la gestión tanto de sus empleados como de sus juegos y actividades que ofrece a los visitantes".

Para empezar a construir la aplicación, debemos empezar por identificar los actores de la realidad, y representarlos mediante **transacciones**. ¿Qué transacciones debemos crear entonces en la base de conocimiento (KB)?

TRANSACCIÓN "EMPLOYEE"

Preguntamos: ¿qué datos registran de los empleados de la empresa? La respuesta es la siguiente:

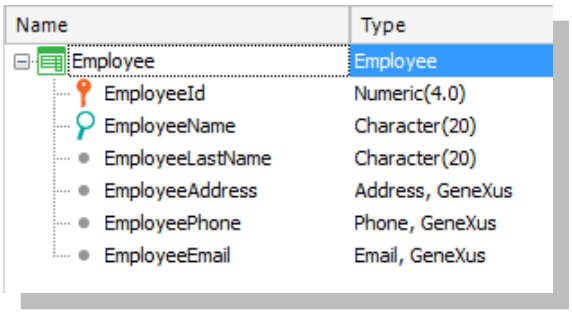
> El **nombre** (que no supera los 20 caracteres), **apellido** (que tampoco los supera), **dirección**, **teléfono** y el **e-mail**.

Con estos datos ya puede crear la transacción *Employee*.

Recordar que:

- Para crear objetos existen varias alternativas:
  - Hacerlo desde el menú: File/New/Object
  - Ctrl+N
  - Ícono de la barra de herramientas
- Necesitará un atributo que identifique a cada empleado (EmployeeId).
- Digitando punto (".") cuando va a ingresar un nuevo atributo, éste se inicializa con el nombre de la transacción.

La estructura de la transacción debería haberle quedado como se muestra:



Recordar que:

- **Address, Phone** e **Email** son **dominios semánticos** que se asignan automáticamente a los atributos que se definen conteniendo en su nombre los textos Address, Phone o Email respectivamente.
- Cuando esté definiendo el tipo de datos del atributo identificador, en vez de utilizar directamente Numeric(4.0), defina el **dominio Id** con ese tipo de datos.
  Configure la propiedad **Autonumber** de ese dominio en **True**, para que todos los atributos basados en el mismo se numeren automáticamente, sin que el usuario deba preocuparse.

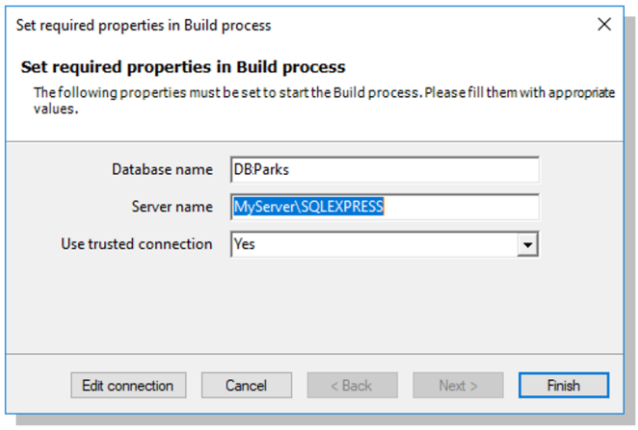
El próximo paso es probar la aplicación en ejecución. Asegúrese de tener la ventana ***Output*** de GeneXus habilitada y a la vista. **(*View/Other Tool Windows /Output*).**

Ahora sí, **pruebe** la aplicación en ejecución presionando **F5**.

¿Qué sucederá?

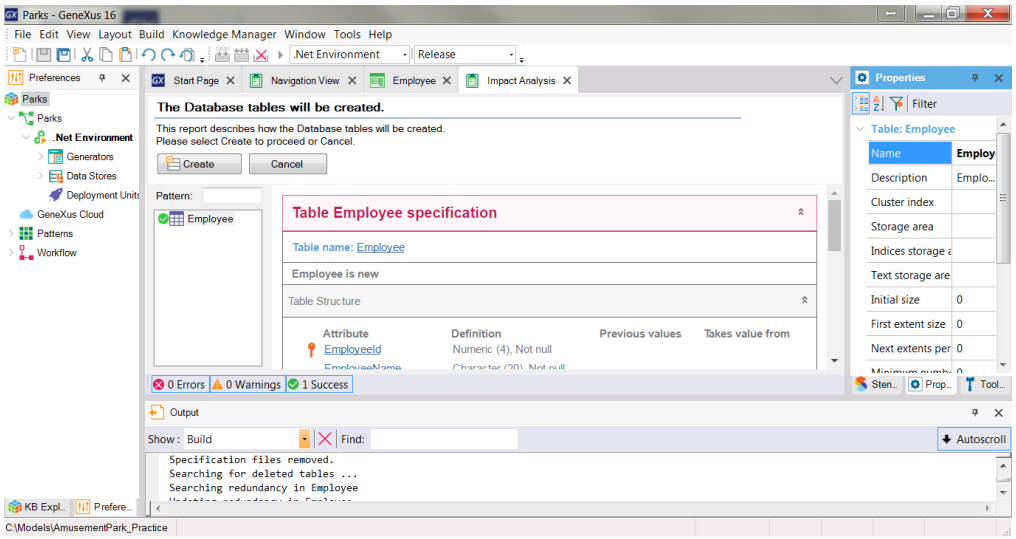
Solución

Si decide crear la base de datos y programas **localmente,** se le abrirá una ventana como la siguiente para que ingrese la información de Base de Datos, Servidor y método de conexión. Recuerde que si no existe una base de datos con el nombre que indicó en ese servidor, GeneXus la **crea**.



Si en cambio la base de datos y programas se crearán en la nube, el diálogo anterior no aparece puesto que GeneXus conoce los datos del servidor en la nube y configura automáticamente el nombre de la base datos y toda la información de conexión a la misma.

A continuación, se despliega un **Análisis de Impacto** que detalla que se creará la base de datos y la tabla *EMPLOYEE* dentro de la misma:



Si presiona el botón ***Create***, GeneXus procederá a ejecutar el programa que llevará a cabo la creación. Al finalizar el proceso, se le abrirá en el navegador que tenga configurado como predeterminado, el menú con links para ejecutar los objetos definidos. En este caso sólo uno: la transacción *Employee*.

Ingrese algunos empleados al sistema. Luego, **modifique** algún dato de alguno de los empleados previamente ingresados y **elimine** algún empleado.

También pruebe usar las flechitas ofrecidas para pasar de registro en registro de empleados y la opción *SELECT*, que ofrece una "lista de selección" para ver la lista de empleados registrados y seleccionar uno.

Ahora pasemos a identificar y crear la siguiente transacción. Recordemos lo que nos habían enunciado, a lo cual se le hicieron algunos agregados:

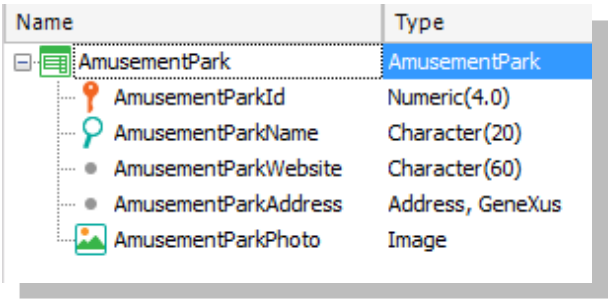
> "Nosotros registramos los datos de **parques de diversiones** de diferentes ciudades en diferentes **países**, para realizar la gestión tanto de sus empleados como de sus juegos y actividades que ofrece a los visitantes".

TRANSACCIONES "AMUSEMENTPARK" Y "COUNTRY", RELACIONADAS

Le preguntamos a los empleados de la empresa: ¿qué datos registran de los parques de diversiones con los que trabajan? La respuesta es la siguiente:

> El **nombre** (que no supera los 40 caracteres), **sitio web** (que no supera los 60 caracteres), **dirección** y **foto representativa**.

Con estos datos ya puede crear la transacción *AmusementPark*.



Recordar que:

- **Address** es un **dominio semántico** que se asigna automáticamente a los atributos que se definen conteniendo en su nombre el texto Address.
- Cuando esté definiendo el tipo de datos del atributo identificador, en vez de utilizar directamente Numeric(4.0), defina el **dominio Id** con ese tipo de datos.
- Configure la propiedad **Autonumber** de ese dominio en **True**, para que todos los atributos basados en el mismo se numeren automáticamente, sin que el usuario deba preocuparse.

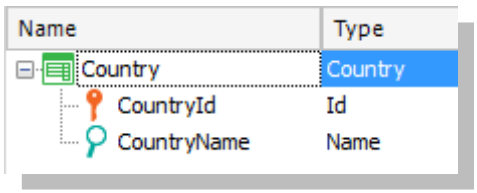
Vamos a crear una transacción para registrar los países a los que los parques de diversiones pertenecen.

Recuerde que:
- presionando punto (".") cuando está por dar nombre a un atributo en la estructura de la transacción, aparece inicializado con el nombre de la transacción.
- Necesitará un atributo identificador, CountryId.

Cuando esté definiendo el tipo de datos del atributo identificador, en lugar de utilizar directamente *Numeric(4.0)*, defina el **dominio *Id*** con ese tipo de datos.

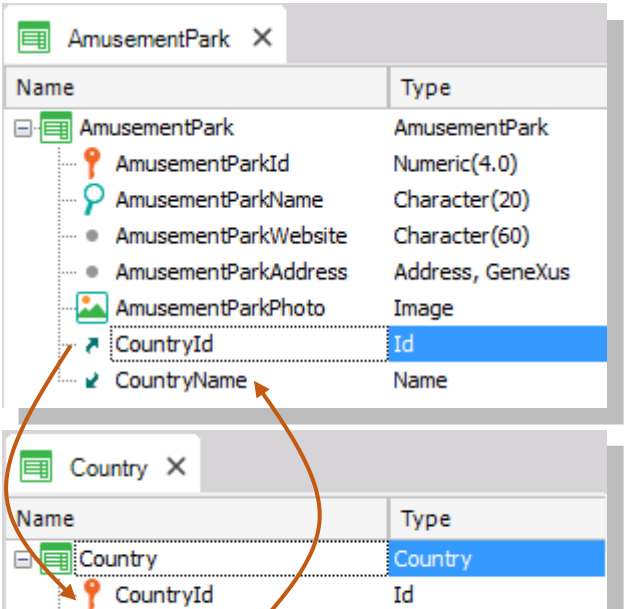
Configure la propiedad ***Autonumber*** de ese dominio en ***True***, para que todos los atributos basados en el mismo se numeren automáticamente, sin que el usuario deba preocuparse.

Defina el atributo ***CountryName***, creando y utilizando un nuevo dominio: ***Name***=*Character(50).*



Ahora, volveremos a la transacción ***AmusementPark***, para agregarle los atributos *CountryId* y *CountryName*.

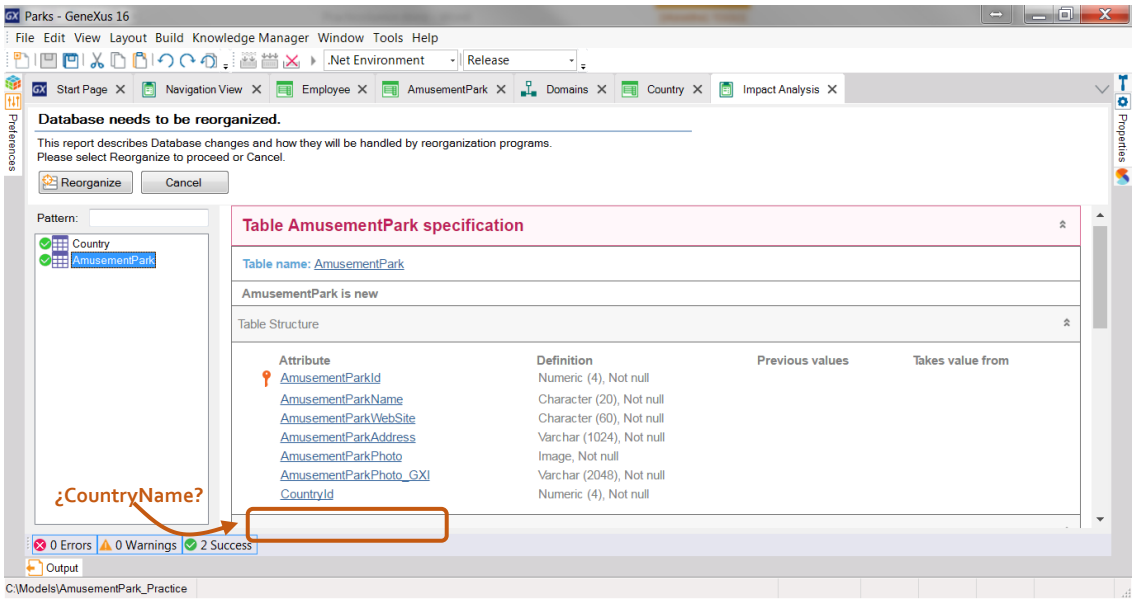
Aprovechemos para cambiar el tipo de datos de *AmusementId*, asignándole el dominio Id previamente creado si es que aún no lo ha configurado.



¿Por qué colocó además de *CountryId*, el atributo *CountryName* en *AmusementPark*?

Respuesta: El atributo *CountryName* es importante que aparezca en la pantalla de la transacción, para mostrarnos el nombre del país, que es el dato que recordamos mejor del país, en lugar de ver solamente su identificador. También es necesario agregar al atributo en la estructura de la transacción si luego lo queremos utilizar, por ejemplo, dentro de una regla.

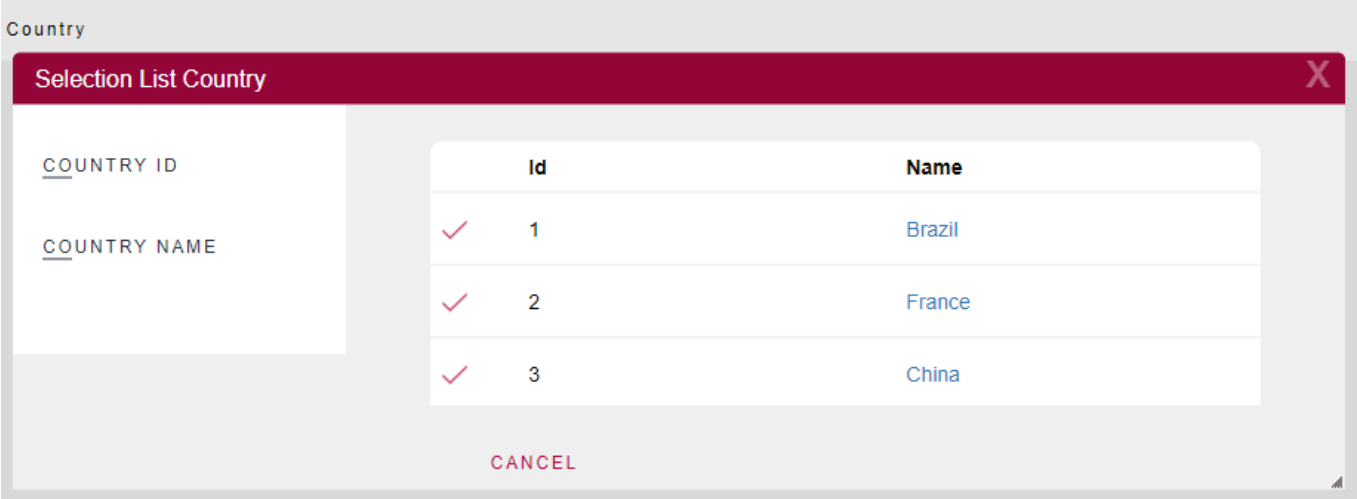
Ejecute para probar (**F5**) y le aparecerá el siguiente reporte:



¿Por qué en la tabla *AmusementPark*, que GeneXus informa que se debe alterar en la Base de Datos, no aparece el atributo *CountryName*? Es decir, ¿por qué la tabla física no lo contendrá, cuando sí está en la estructura de la transacción?

Después de estudiar el reporte, si estamos de acuerdo, presionamos ***Reorganize*** para que efectivamente se lleve a cabo eso que se informa. Se abrirá el navegador con el menú con links a los 3 programas que corresponden a cada una de las transacciones (*AmusementPark*, *Country* y *Employee*).

Ingresar como **países** a: Brasil, Francia y China. Observar que dejando el valor o como valor en el identificador, al grabar se le asigna automáticamente el número posterior al último asignado (efectivamente, se está autonumerando).



Ingresar como parque de diversiones: "Beto Carrero World", que está en Brasil. Si no recuerda el identificador de Brasil en el sistema, ¿cómo ingresa el país? Se le ofrece un ícono con una flecha al lado de *CountryId*, para abrir una "Lista de selección" de países, creada automáticamente por GeneXus. Esto es porque *CountryId* tiene el rol de llave foránea (foreign key) en esta transacción (es decir, está "apuntando" a otra tabla).

DATOS RELACIONADOS: ¿CÓMO SE MANTIENE LA INTEGRIDAD?

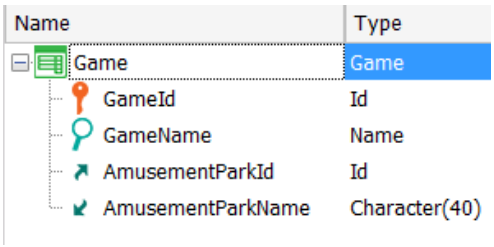
AmusementPark y *Country* están relacionados. Al colocar *CountryId* en la estructura de *AmusementPark*, por tener exactamente el mismo nombre que el atributo que es llave primaria en la transacción *Country*, GeneXus entiende que en *AmusementPark* el atributo *CountryId* es llave foránea y mantiene automáticamente la integridad de la información. Así, por ejemplo:

- Intente ingresar un parque de diversiones con un Id de país que no exista. ¿Le permite grabar ese parque?
- Elija un parque previamente ingresado (por ejemplo, 'Beto Carrero World') y cambie el país, por uno que no exista. ¿Pudo grabar la modificación?
- Intente eliminar un país (usando la transacción *Country*) que tenga algún parque asociado (por ejemplo, Brasil). ¿Se lo permite?

***Conclusión**: los programas correspondientes a las transacciones aseguran la integridad de los datos.*

TRANSACCIÓN 'GAME'

Como nos fue solicitado inicialmente, el sistema debe brindar la posibilidad de ingresar los juegos disponibles en cada parque, por lo que debemos crear una transacción que contenga su nombre y el parque de diversiones al que pertenece.

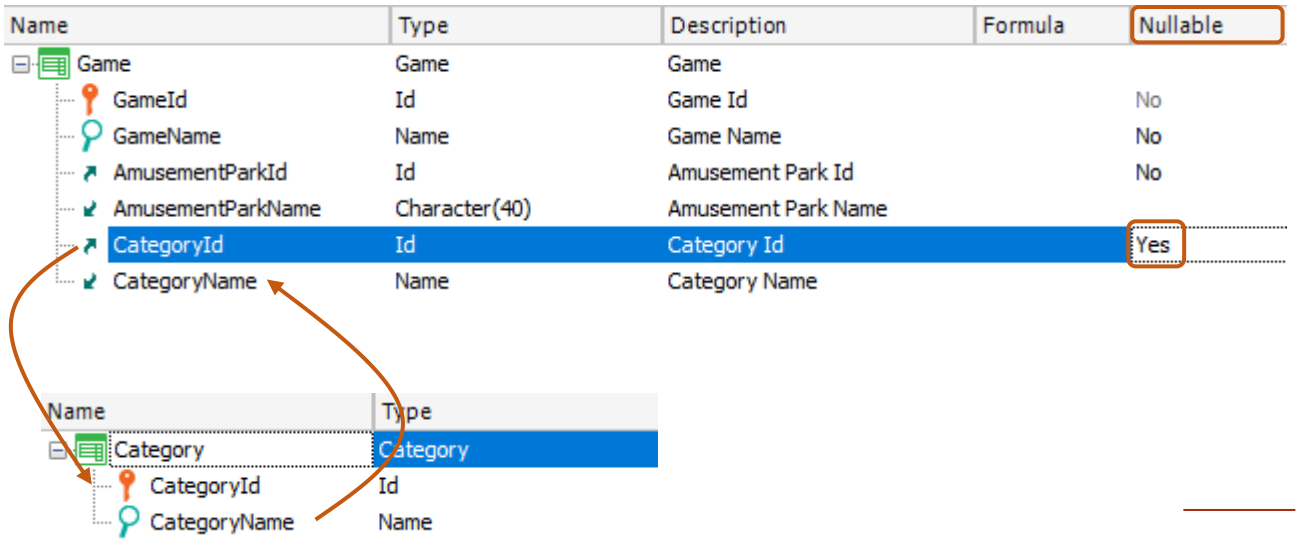


TRANSACCIÓN 'CATEGORY'

Nos falta completar la información de la transacción *Game*. Los empleados describieron que también registran de cada juego la **categoría** (infantil, radical, recreativo, etc.) a la que pertenece. Así que necesitaremos crear una transacción para registrar esta información, y agregar la categoría a la transacción *Game*.

Pero, además, nos han informado que no es obligatorio registrar indefectiblemente la categoría a la que pertenece un juego dado que se está manipulando. Se puede dejar vacía. Si sabemos que GeneXus controla automáticamente la integridad, ¿cómo lo conseguimos?

Solución:



Para terminar la definición de la transacción *Game*, agreguemos el dato que nos está faltando: la foto.

Para ello, cree el atributo ***GamePhoto*** de tipo de datos ***Image***.

Pídale a GeneXus que construya la aplicación, así puede probarla en ejecución (**F5**).

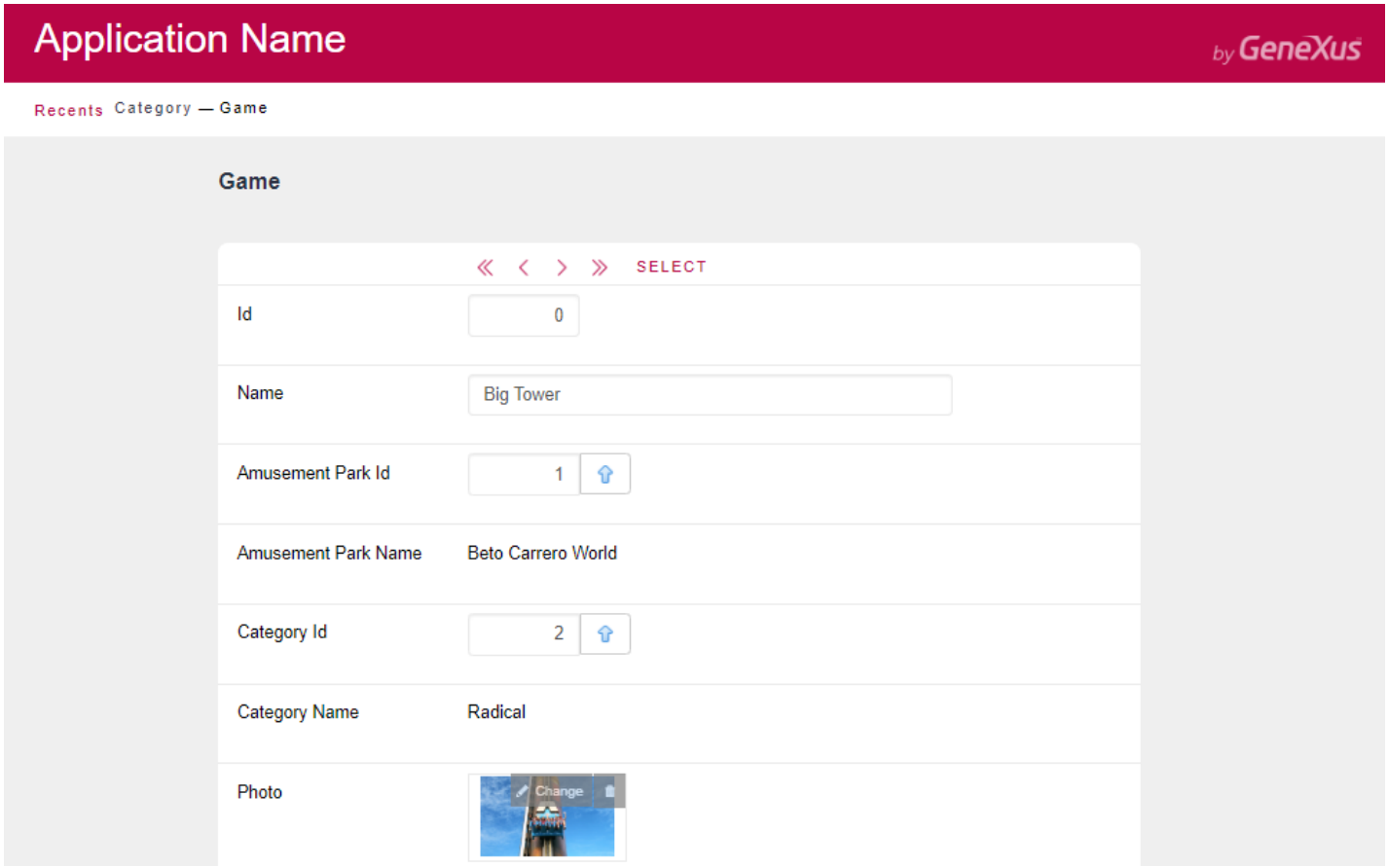
> **Observe** lo que le informa el reporte de Análisis de Impacto. Deberán crearse las tablas *Category* y *Game* (no se preocupe en entender por qué requiere almacenar dos valores por imagen).

Reorganice y **ejecute**.

Ingrese categorías (como infantil y radical) y juegos (como montañas rusas y carruseles).

Observe que en este caso puede dejar la categoría vacía (debido a que configuró la propiedad **Nullable** en **Yes** en la estructura de la transacción).

Sin embargo, si intenta poner como valor de *CategoryId* para el juego un valor inexistente, no le dejará grabar.

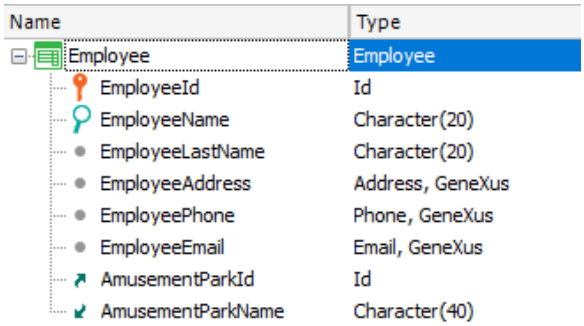


TRANSACCIONES "EMPLOYEE" Y "AMUSEMENTPARK", RELACIONADAS

Como nos dijeron al comienzo del desarrollo de la aplicación, el sistema gestionará los parques de diversiones y **sus** empleados. Por lo tanto, necesitamos vincular a los empleados que son registrados con el parque en el cual trabajan.

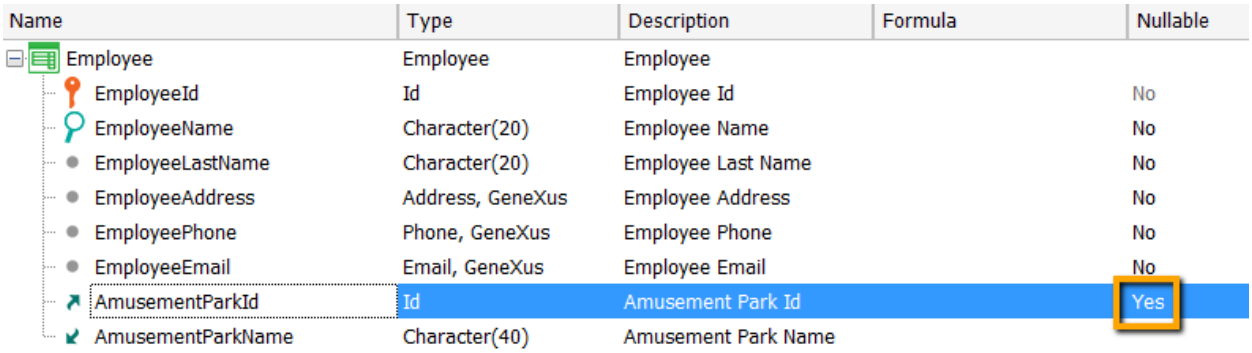
Para eso, vayamos a la transacción ***Employee*** y agreguemos los atributos *AmusementParkId* y *AmusementParkName*.

Aprovechemos para cambiar el tipo de datos de *EmployeeId*, asignándole el dominio *Id* previamente creado si todavía no lo ha configurado.

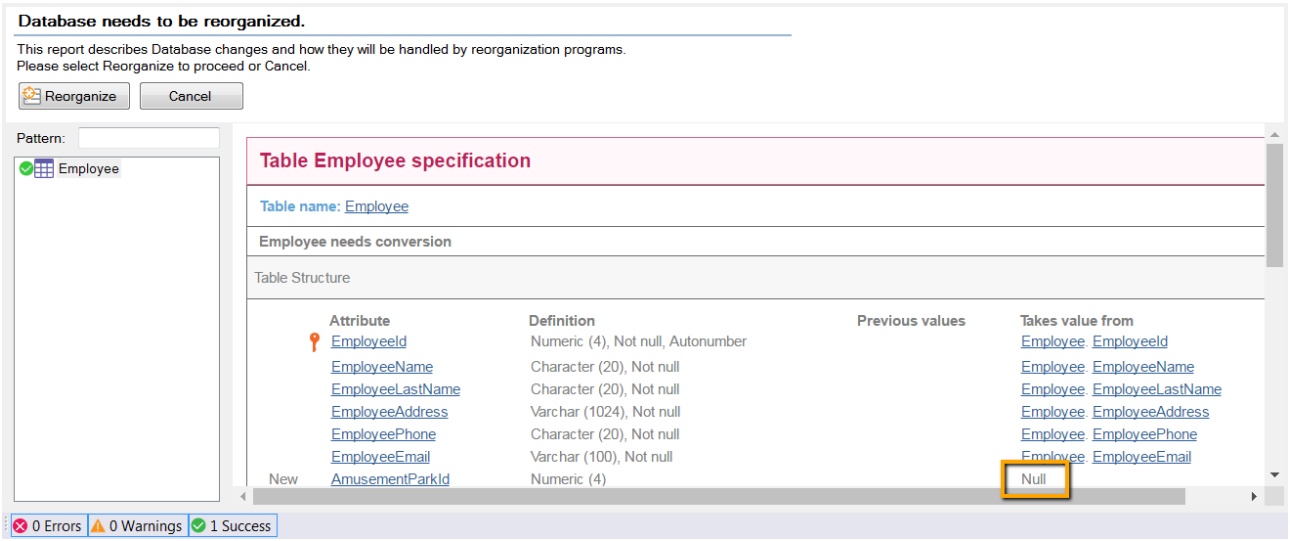


Nos han informado que no es obligatorio registrar en el momento el parque en donde trabaja el empleado, es decir, se puede dejar vacío. ¿Qué debemos hacer?

Solución:

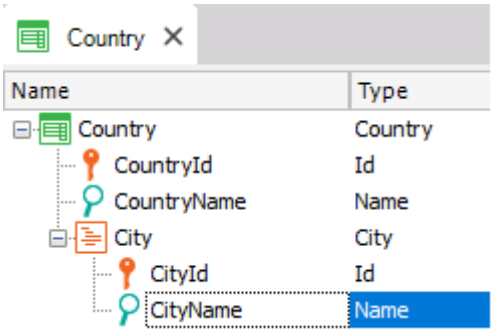


Ejecute para probar (**F5**) y le aparecerá el siguiente reporte indicándonos que el atributo AmusementParkId ahora permite dejar un valor no especificado:



AGREGUEMOS LAS CIUDADES A LA TRANSACCIÓN 'COUNTRY'

Además de los países, necesitamos registrar la información de sus ciudades. Por lo tanto, debemos agregar un segundo nivel a la transacción *Country*, con el identificador y el nombre de ciudad.

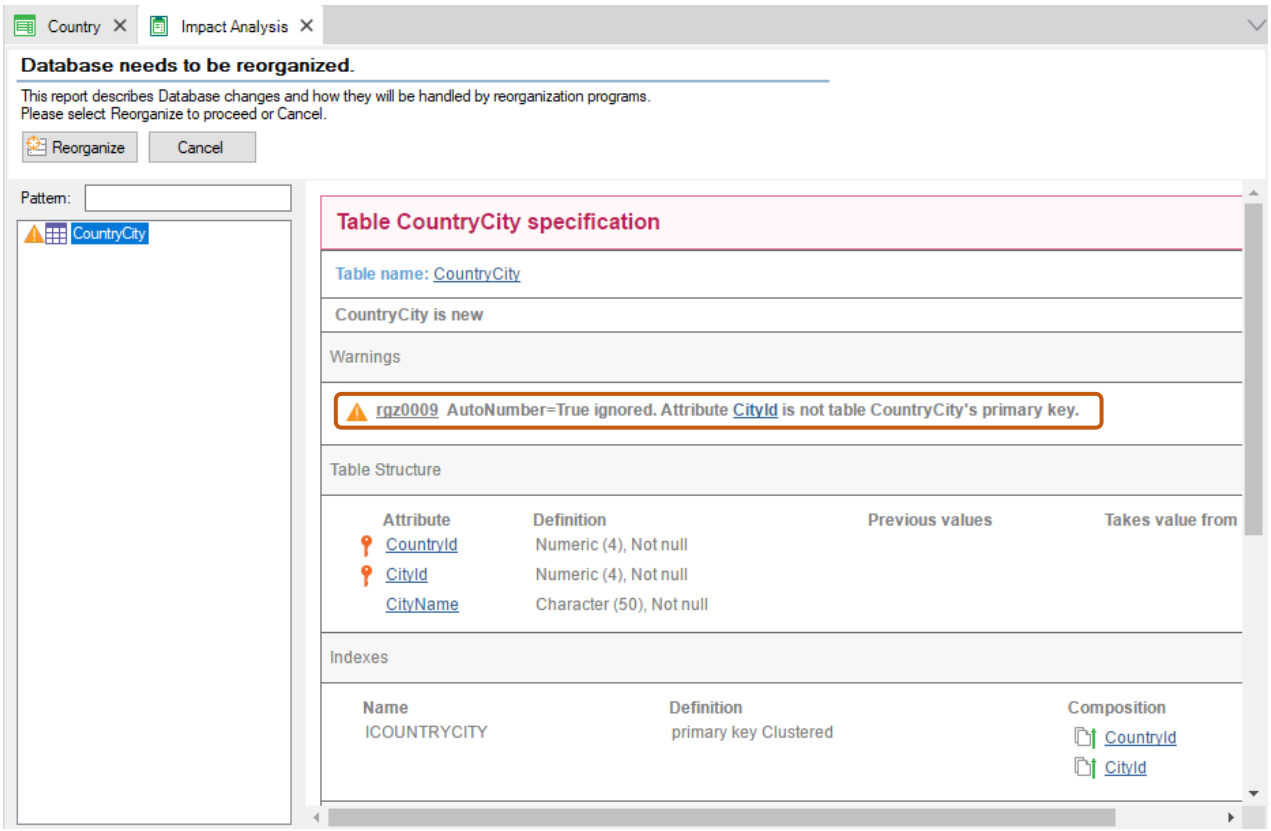


Recuerde que:

- Posicionado en el atributo *CountryName*, con botón derecho, ***Insert Level***, agrega el subnivel.
- Una vez que le dé un nombre al nuevo nivel, digitando comillas (") en lugar de punto, el atributo que defina se inicializará con el nombre del nivel.
- Las ciudades se identificarán por su propio Id en combinación con el del país. Es decir, no podrá identificar a una ciudad sin brindar antes la información del país del que se trata. Así, podría haber una ciudad 1 Rosario tanto para Uruguay como para Argentina:
  País: 1 (Uruguay) – Ciudad: 1 (Rosario)
  País: 2 (Argentina) – Ciudad: 1 (Rosario)

- O incluso podría ser que Rosario para Argentina se identificara con otro número:
  País: 2 (Argentina) – Ciudad: 4 (Rosario)

Reorganice y **ejecute** (**F5**).



Observe que el Listado de Navegación le informará que:

- La propiedad ***Autonumber*** para el caso de ***CityId*** será ignorada. Esto significa que en ejecución el usuario deberá ingresar manualmente los identificadores de ciudad. La explicación es que la propiedad *Autonumber* solamente autonumera llaves primarias simples y en este caso *CityId* es el segundo componente de una llave compuesta.
- Se creará una nueva tabla ***CountryCity*** para almacenar la información correspondiente a las ciudades.

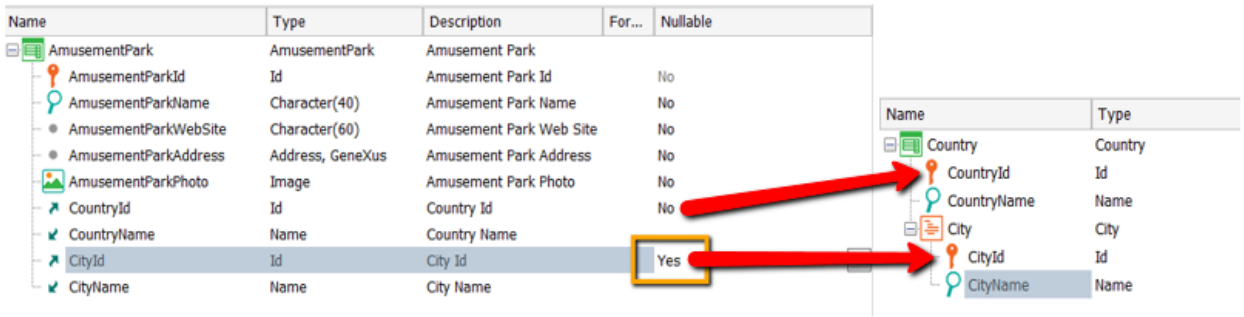
Ingrese ciudades para los países que ya tenía registrados.

TRANSACCIÓN "AMUSEMENTPARK": AGREGUEMOS LA CIUDAD.

En la transacción *AmusementPark* agreguemos la ciudad del país a la que el parque pertenece. **¿Qué debe hacer si la empresa nos informa que ese valor puede no ser conocido o relevante para un parque dado en un momento dado?**

Construya la aplicación y pruébela (**F5** y **Reorganize**).

Solución:



AGREGUEMOS COMPORTAMIENTO A LAS TRANSACCIONES (RULES)

Después de probar con nosotros la aplicación que venimos desarrollando, en la empresa nos cuentan que para los empleados hay algún comportamiento específico que debemos hacer cumplir a la hora de manipular la información a través del programa (transacción *Employee*).

¿Cuál es este comportamiento?

Nos dicen:

- "El sistema no debe permitir ingresar empleados sin nombre, ni sin apellido".
- "Debe advertirse al usuario si está dejando el teléfono sin asignar, por si fue un descuido".
- "Se debe registrar la fecha de ingreso del empleado al sistema (**EmployeeAddedDate**) y se debe proponer como valor predeterminado para ese atributo, la fecha de hoy".

Especifique ese comportamiento y pruébelo (**F5** y **Reorganize**).

Recuerde que:

- Las reglas finalizan con punto y coma ";".
- El método **IsEmpty()** aplicado a un atributo devuelve True cuando el atributo está vacío y False en caso contrario.
- La variable **&Today** es del sistema y tiene cargado el valor de la fecha del día.

Para escribir una variable dentro de la pantalla **Rules**, cuando digita "&" se le despliegan todas las variables definidas hasta el momento para que seleccione la que necesita. La otra posibilidad es utilizar **Insert** / **Variable.**

Pruebe ingresar un nuevo empleado dejando vacío el nombre. ¿Le permite grabar o pasar al siguiente campo?

Ídem con el apellido. ¿Sucede lo mismo con el teléfono?

Si luego le informan que la fecha de ingreso al sistema no debería ser manipulada por el usuario, sino únicamente visualizada, ¿cómo establece este comportamiento?

Especifíquelo y pruébelo en ejecución.

PATTERNS: MEJORANDO LA INTERFAZ PARA TRABAJAR CON LA INFORMACIÓN

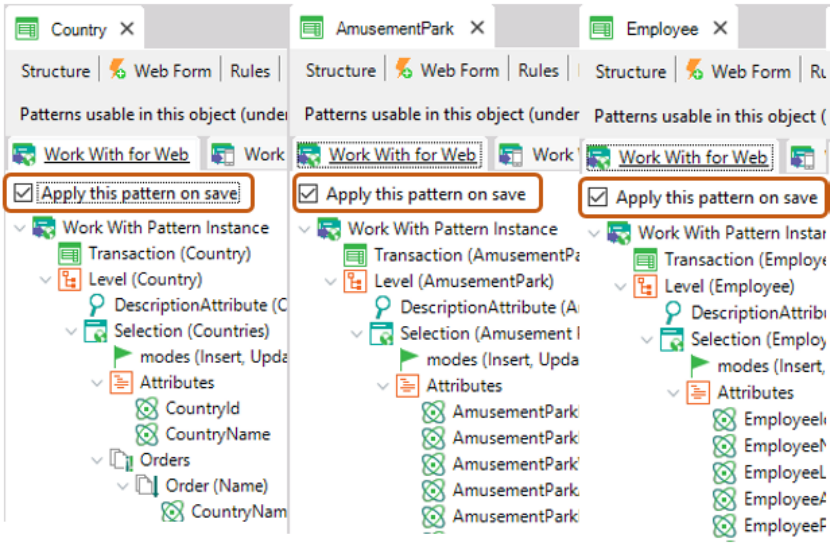
Al mostrarle al cliente lo realizado hasta ahora, nos dice que quisiera poder manipular la información de países, parques de diversiones, empleados, categorías y juegos de un modo más potente y vistoso (que ofrezca consulta, posibilidad de filtrar, así como insertar, modificar y eliminar datos, etc.).

Para ello deberá aplicar el patrón *Work With for Web* a las tres transacciones. Pruébelo y véalo en ejecución.

Observar que:

- existe un Work With para Smart Devices, también. Pero el que usted deberá aplicar es el que corresponde a la aplicación web que está construyendo.
- GeneXus creará automáticamente varios objetos por transacción, para implementar el "Trabajar con" esa entidad.

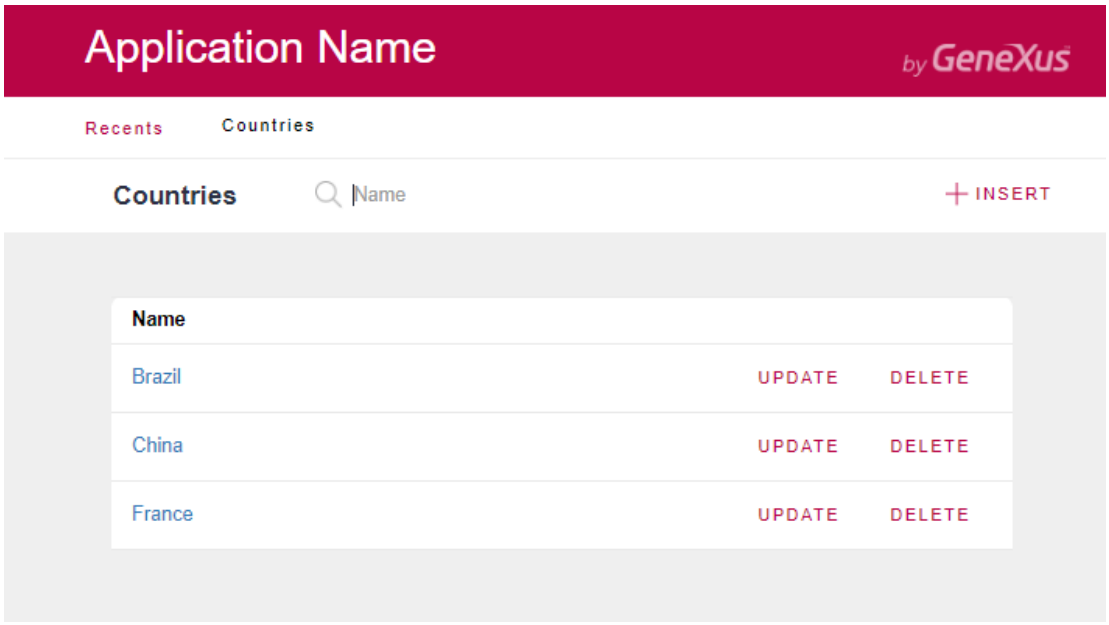
Solución:



¿Por qué no aparecen más en el *Developer Menu* las transacciones *Country, AmusementPark y Employee?*

Pruebe hacer lo siguiente:

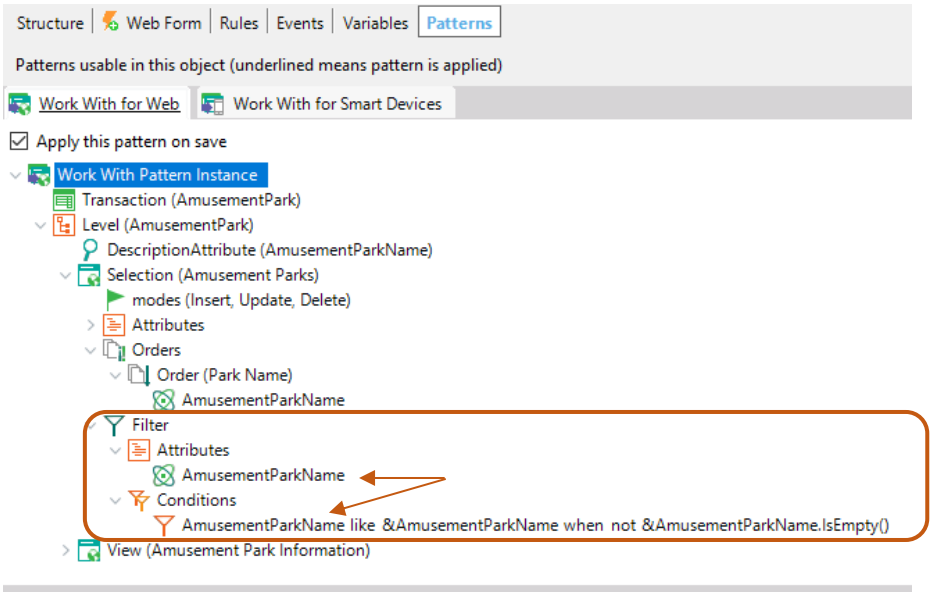
1. Ingresar un nuevo país.
2. Modificar un país existente (por ejemplo, agregándole una ciudad).
3. Eliminar un país existente.
4. Visualizar la información de un país.
5. Realizar una búsqueda por nombre de país.



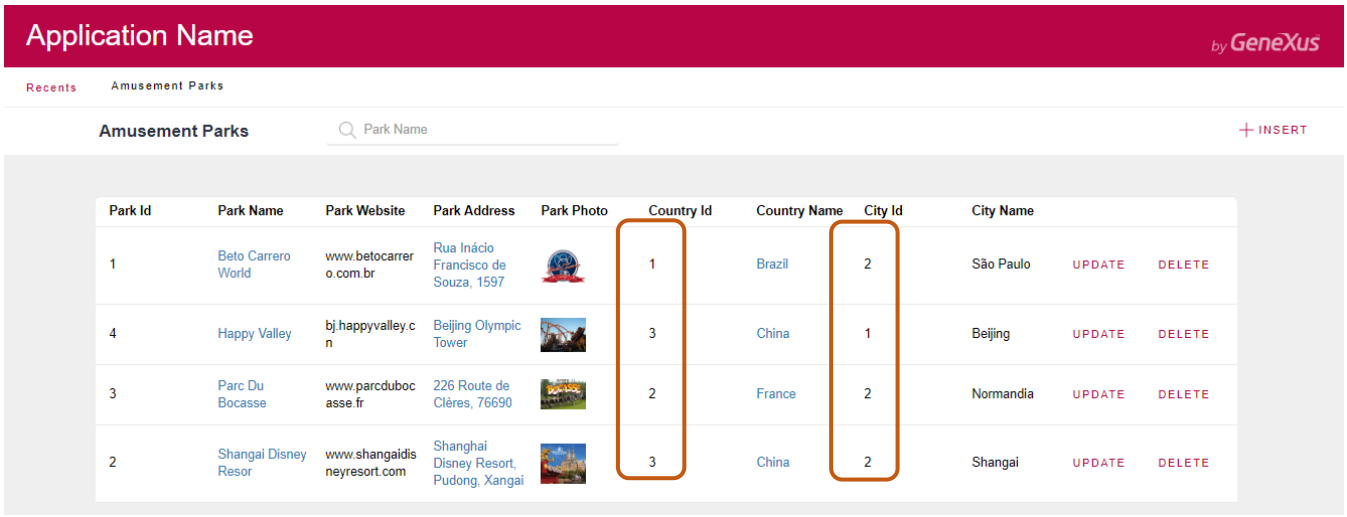
6. Ingrese un par de parques de diversiones (Ej: Shangai Disney Resort, de China/Shangai, Parc Du Bocasse de Francia/Normandía, Happy Valley, de China/Beijing).
7. Filtre los parques de diversiones cuyo nombre empiece con P. ¿Y si ahora quiere poder visualizar todos los parques de China? No está incluida esta posibilidad, por lo que deberemos personalizar el pattern *Work With* de esta transacción, para agregárselo. Hágalo **en GeneXus** y **pruebe** en **ejecución**.

Solución:

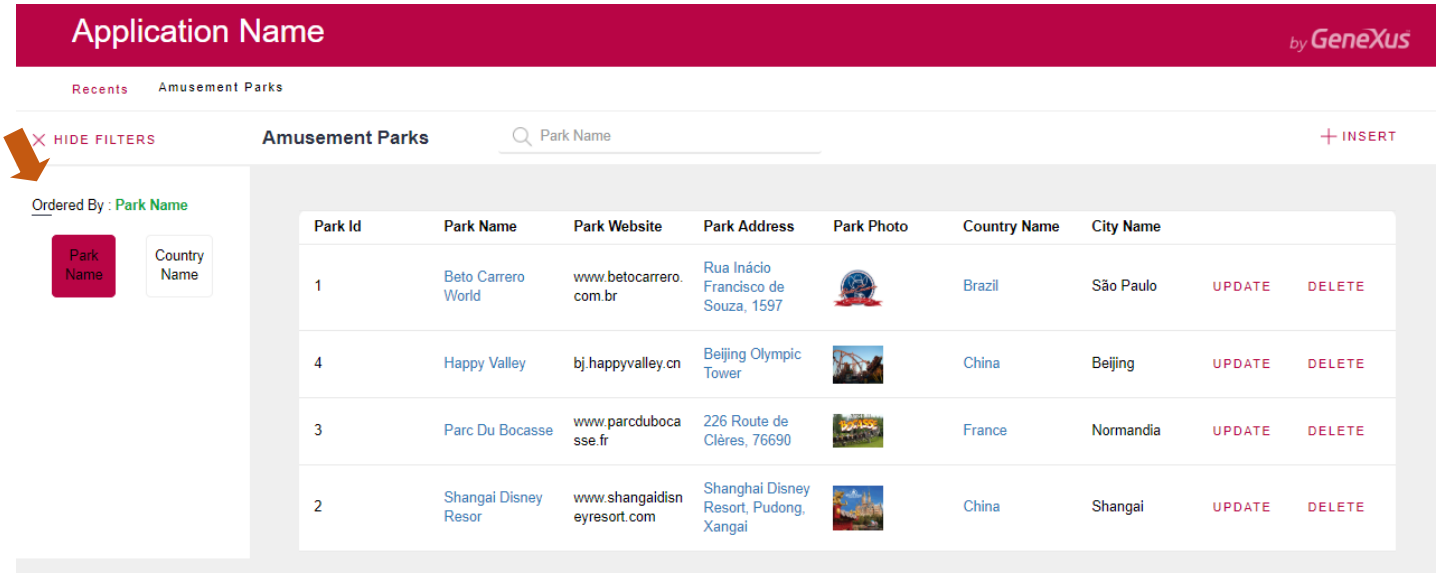
Para ello primero observe cómo está especificado el filtro que sí existe, por nombre del parque de diversiones:



8. Ahora quite los identificadores de país y ciudad de la pantalla del *Work With* y pruébelo en ejecución.



9. Si ahora quiere brindar la posibilidad de que el usuario elija si quiere ver los parques de diversiones ordenado por nombre del parque o por nombre de país**, impleméntelo** y **pruebe**.



TRANSACCIONES "REPAIR" Y "TECHNICIAN" Y NECESIDAD DE DEFINIR SUBTIPOS

Se necesita ahora registrar los juegos que entran en estado de reparación. Cada reparación tiene un identificador, una fecha desde la cual el juego deja de estar disponible para su uso, la cantidad de días estimada para su reparación, el identificador del juego, su nombre, el técnico titular y el técnico suplente. También cada reparación tiene un costo. Para el costo cree un dominio llamado *Cost*, del tipo Numeric(8.2).

Cree una transacción para registrar a los técnicos que trabajarán en las reparaciones. Cada técnico tiene un identificador, un nombre y apellido, un teléfono, un país y una ciudad en la cual se encuentra.

¿Cómo se define que cada reparación tiene un técnico titular y otro suplente?

Recuerde

1) Que en la estructura de la transacción:
- un ícono representando una flecha hacia arriba informa que el atributo es clave foránea (Foreign Key), es decir que apunta a otra tabla.
- Un ícono representado una flecha hacia abajo informa que el atributo es inferido de otra tabla.
- Un ícono representando una S indica que el atributo es un subtipo.

2) Sobre los grupos de subtipos:
- Se definen de la misma manera que cualquier tipo de objeto.
- Cada grupo de subtipos debe contener obligatoriamente un subtipo de un atributo primario (que es llave primaria de una tabla) o conjunto de atributos que forman una llave primaria.
- En cada grupo de subtipos, hay que incluir todos los atributos subtipos que se necesiten conocer, pertenecientes a la tabla base y/o extendida de la clave primaria del grupo.

Ejecute y **verifique** que al intentar ingresar una reparación, se dispare un error si el técnico titular que está queriendo asignarle a la reparación no existe. Ídem para el técnico suplente.

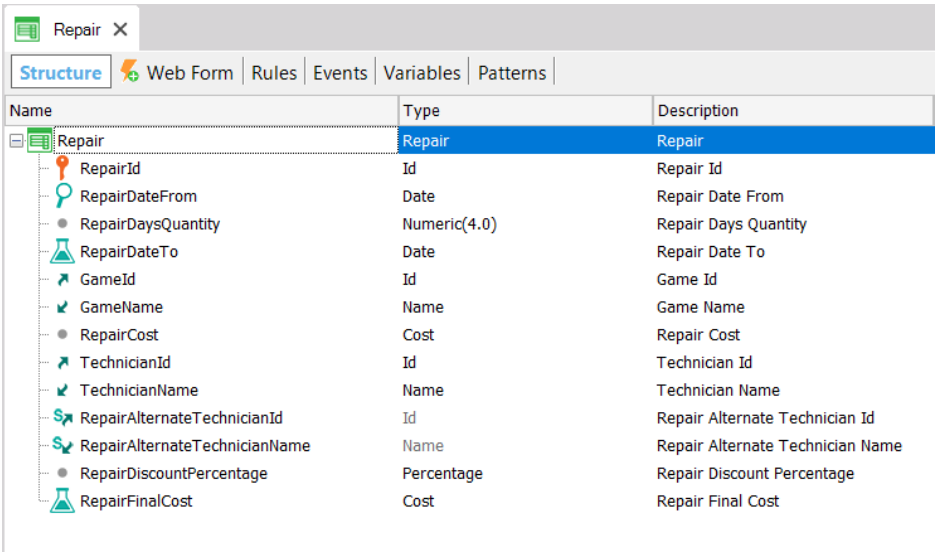
No debe permitirse ingresar una reparación cuyo técnico titular coincida con el técnico suplente. Implemente ese comportamiento y pruébelo en ejecución.

FÓRMULAS

Se necesita poder registrar **el descuento actual que tiene cada reparación.** Defina un nuevo atributo en la transacción *Repair* **para almacenar este dato.** Darle al nuevo atributo el nombre: *RepairDiscountPercentage* y que su tipo de datos sea un dominio *Percentage*, numérico de largo 3.

Se desea visualizar el precio final de la reparación con el descuento aplicado. Para resolver esto, defina otro atributo más, de nombre *RepairFinalCost*, que sea **fórmula global** y que calcule automáticamente el precio final de la reparación.

Agregue un nuevo campo llamado *RepairDateTo*, el cual será una suma entre la fecha de inicio de reparación y la cantidad de días que la misma llevará.

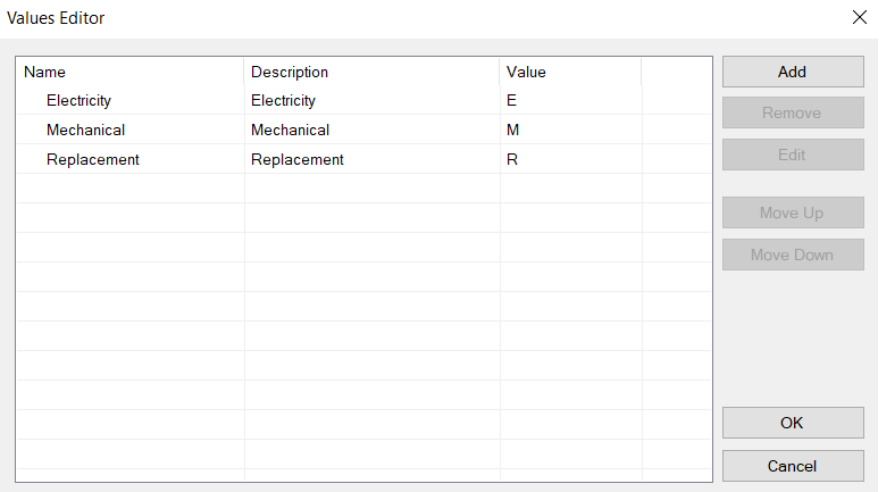


Presione F5, **observe en el Análisis de Impacto cuál atributo se creará físicamente y cuál no,** reorganice y pruebe la aplicación en funcionamiento.

CREACIÓN DE SEGUNDO NIVEL

Se desea crear un segundo nivel en la transacción *Repair* para allí guardar un detalle del tipo de problema encontrado para reparar.

Para ello, antes que nada se deberá crear un dominio llamado *KindName*, Character(1). Restrinja los valores posibles para el dominio: que sean válidos los valores "E","M" y "R" (editando la propiedad **Enum Values** del mismo como se ve a continuación).



Cree un segundo nivel en la transacción *Repair* de nombre *Kind* para registrar el tipo de reparación. Este nivel tendrá estos tres atributos:

- *RepairKindId* – Numeric(4) (será clave en este segundo nivel)
- *RepairKindName* – Basado en dominio *KindName* (Genexus lo sugerirá automáticamente).
- *RepairKindRemarks* – Character(120), contendrá observaciones sobre el problema, un pequeño detalle del problema encontrado o la pieza a remplazar por ejemplo.

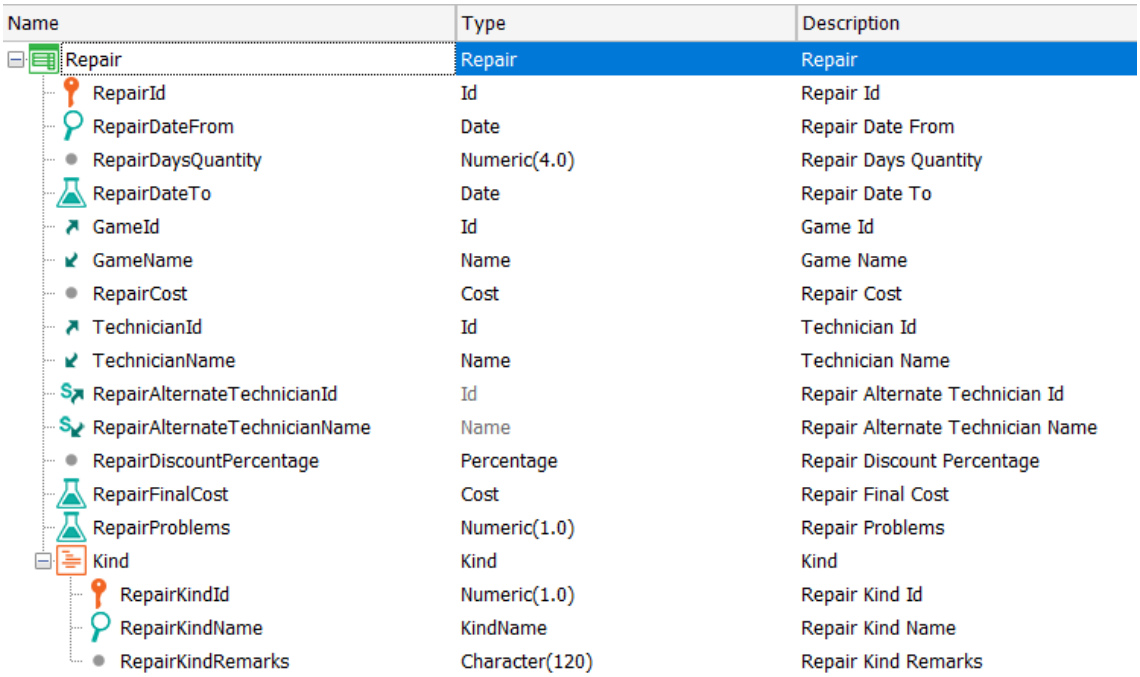
Recuerde que para definir que cierto atributo es parte de la llave primaria, debe presionar el botón derecho del mouse sobre el atributo y el menú contextual le ofrecerá la opción **Toggle Key**. En este caso, no será necesario ya que sólo el primer atributo forma parte de la clave primaria y éste ya aparece con el ícono de llave.

A efectos de conocer la cantidad de tipos de problema que implica una reparación, cree un nuevo atributo en el primer nivel de la transacción *Repair*, de nombre *RepairProblems*, Numeric(1) y defínalo como fórmula global, deberá contar los tipos de problemas encontrados.

Una reparación podría implicar problemas de electricidad, de mecánica o que sea necesario el remplazo de alguna pieza. Podría incluso tener más de un problema del mismo tipo, dos registros de electricidad, por ejemplo. Si hubiera muchos, se puede detallar algo más con el uso del atributo de observaciones, el cual permite escribir un pequeño texto.

Se desea ver la cantidad de problemas en el formulario web y se debe controlar que se ingresen entre 1 y 3 líneas de tipos de problema.

Este control deberá realizarse **cuando se termine de ingresar datos en el segundo nivel** y luego de haber presionado el botón *Confirm*.



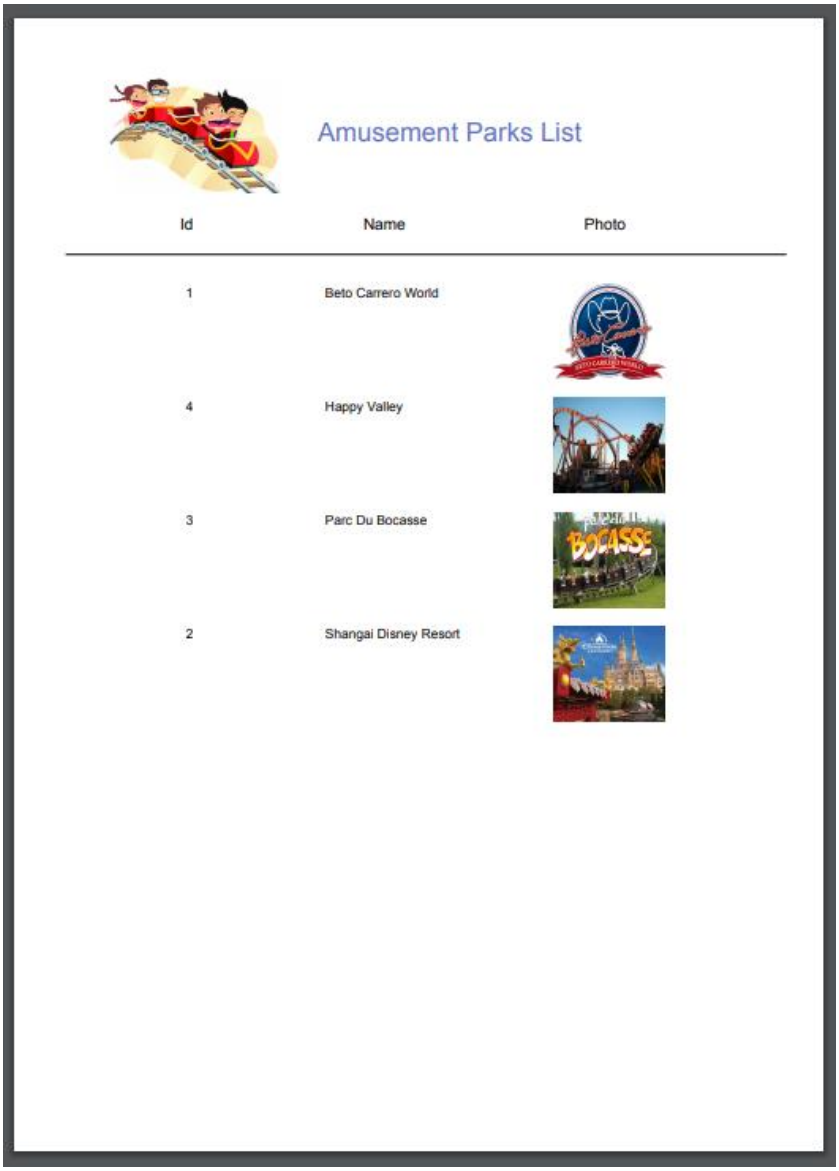
Name	Type	Description
Repair	Repair	Repair
RepairId	Id	Repair Id
RepairDateFrom	Date	Repair Date From
RepairDaysQuantity	Numeric(4.0)	Repair Days Quantity
RepairDateTo	Date	Repair Date To
GameId	Id	Game Id
GameName	Name	Game Name
RepairCost	Cost	Repair Cost
TechnicianId	Id	Technician Id
TechnicianName	Name	Technician Name
RepairAlternateTechnicianId	Id	Repair Alternate Technician Id
RepairAlternateTechnicianName	Name	Repair Alternate Technician Name
RepairDiscountPercentage	Percentage	Repair Discount Percentage
RepairFinalCost	Cost	Repair Final Cost
RepairProblems	Numeric(1.0)	Repair Problems
Kind	Kind	Kind
RepairKindId	Numeric(1.0)	Repair Kind Id
RepairKindName	KindName	Repair Kind Name
RepairKindRemarks	Character(120)	Repair Kind Remarks

El valor de *RepairKindId* se va ingresando manualmente, lo normal sería ir dándole valores a partir de 1. Recordemos que **no es posible autonumerar** utilizando la propiedad *Autonumber*.

LISTADOS PDF

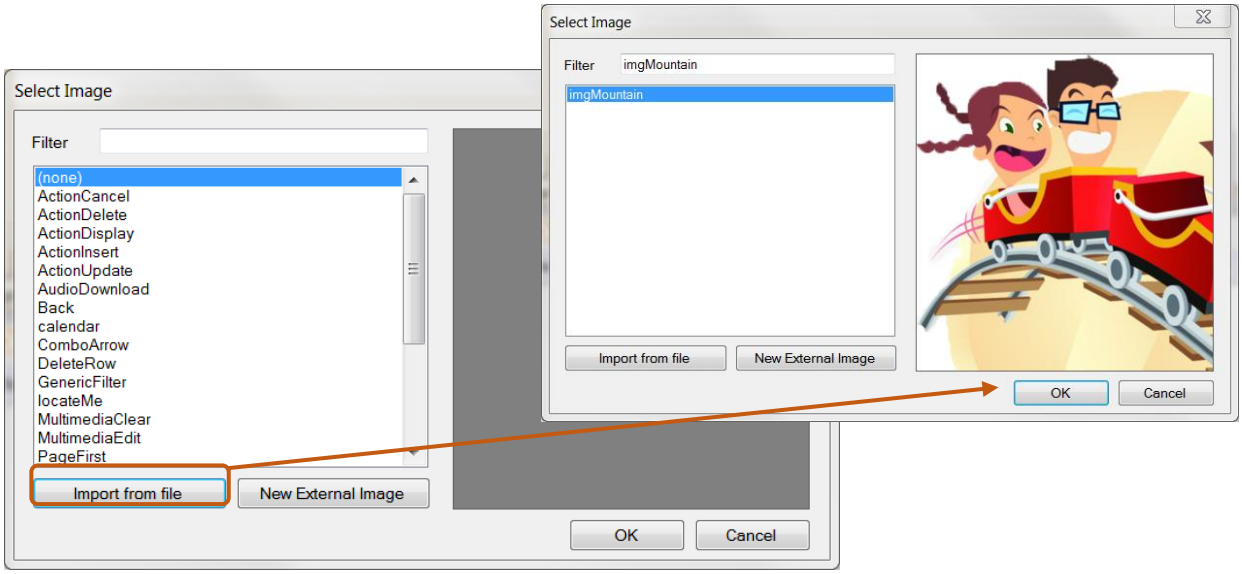
Ahora supongamos que como parte de la aplicación, deberá implementarse la posibilidad de que a pedido del usuario, se le desplieguen listados *PDF* con la información requerida. Por ejemplo, suponga que se necesita un listado que muestre en orden alfabético, los parques de diversiones almacenados en la base de datos.

Se sabe que debe lucir más o menos como sigue:



Sugerencia para poner una imagen en el título: Utilizar el control *Image* presente en la *Toolbox* para desplegar una imagen junto al título del listado.

Dicha imagen deberá ser integrada a la base de conocimiento. Para eso seleccionar la opción *Import from File*, buscar la imagen y asignarle un nombre.



Impleméntelo en GeneXus.

Recordar que para poder visualizar directamente desde el browser un listado, el mismo debe ser generado como PDF. Para esto debe configurar las siguientes propiedades del objeto procedimiento:

- Main program = 'True'
- Call Protocol = HTTP
- Report output = Only to File

Y la siguiente regla:

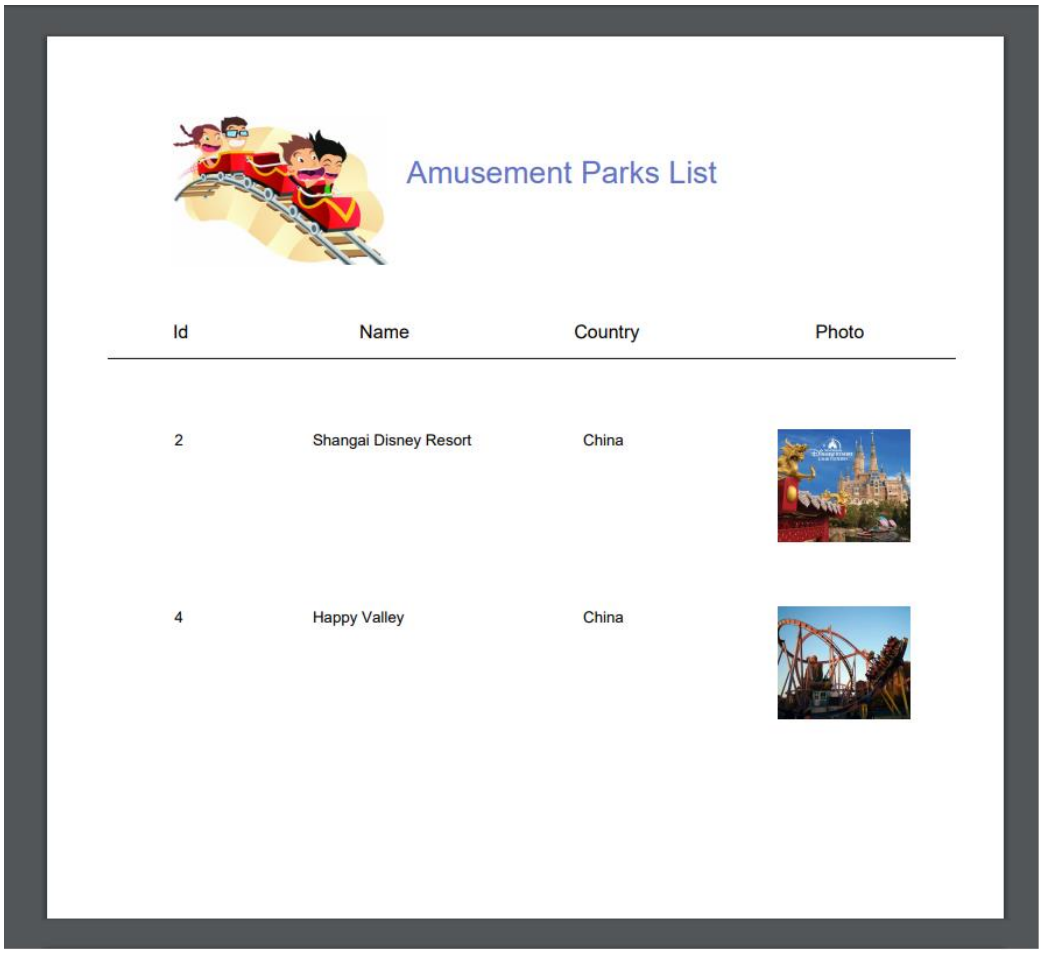
- Output_file('nombre-archivo.pdf' , 'PDF')

Para ejecutar el listado, sobre la pestaña del objeto, botón derecho / **Run with this only**

¿Reparó en lo que le informa el listado de navegación del procedimiento?

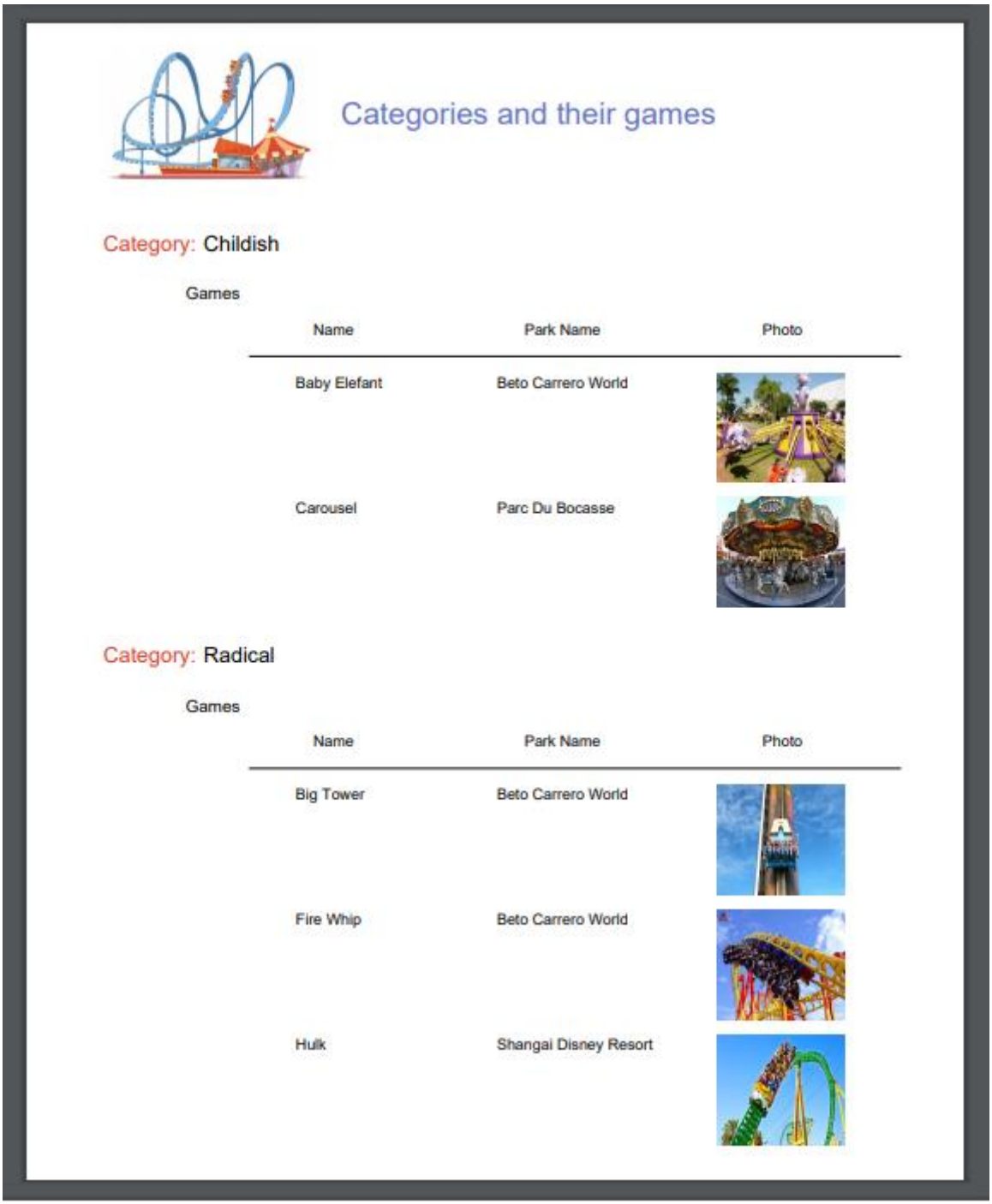
¿Y si ahora se necesita que el listado salga ordenado por nombre de país? Impleméntelo, observe lo informado en el listado de navegación y pruébelo.

¿Y si ahora necesita solamente listar los parques de diversiones de China? Pruébelo (observando el listado de navegación).



En cada caso, deduzca qué tabla de la base de datos se está recorriendo para realizar la consulta del *for each*.

También se necesita un listado como el que sigue (que muestre cada categoría, y por cada una de ellas, sus juegos). Impleméntelo y pruebe lo realizado.



Agregue una nueva categoría al sistema, por ejemplo *Aquatic* (acuáticos). Vuelva a ejecutar el listado anterior. ¿Salió listada la categoría?

Modifique el listado anterior para que no salgan listadas categorías que no tengan juegos relacionados.

¿Qué modificaciones encontró en el listado de navegación?

Respuesta: La aparición de la palabra ***break*** que indica que se está realizando un **corte de control,** mostrándose además que la tabla base del *for each* externo es igual a la del *for each* anidado.

Otra solicitud de la empresa es un listado que muestre todos los nombres de países y para cada país**, la cantidad de parques de diversiones que ofrece**:

Y nos piden otro listado también que muestre todos los países que tienen **más de 2 parques para visitar**:

PASAJE DE PARÁMETROS

Muchas veces necesitamos que un objeto reciba parámetros para que, en base a los mismos, ejecute su lógica. Por ejemplo, una cosa es realizar un listado *PDF* de todos los parques de diversiones de la base de datos, y otra es realizar un listado de aquellos cuyo nombre se encuentra dentro de un rango dado.

LISTADO DE PARQUES EN UN RANGO DETERMINADO

Grabe con otro nombre el procedimiento que había creado anteriormente para listar los parques de diversiones, y modifíquelo para que ahora solamente liste aquellos cuyo nombre se encuentra dentro de un rango recibido por parámetro (el valor inicial y el final del rango serán los parámetros recibidos).

Implemente una pantalla que pida los valores de ese rango al usuario, e invoque a este objeto, pasándole esos valores por parámetro. Pruebe en ejecución.

Recuerde que:

- Si define una variable basada en (**based on**) un atributo, quedará ligada al tipo de datos del atributo, es decir, si éste se modifica, el de la variable también, acorde al cambio.
- Las variables utilizadas para realizar una invocación en el objeto que llama y las utilizadas para declarar los parámetros que se reciben en el objeto llamado, no tienen por qué coincidir en nombre, pero sí deben tener tipos de datos compatibles.

Solución:

Al procedimiento (llamémosle *AmusementParksFromTo*, le agregamos la regla *parm* (y definimos ambas variables en el procedimiento):

```
parm(in: &fromName, in: &toName);
```

Y en su Source:

```
print Title
for each AmusementPark
  where AmusementParkName >= &fromName
  where AmusementParkName <= &toName
        print Data
endfor
```

Y luego creamos un web panel, en el que definimos dos variables, que pueden llamarse de cualquier manera, por ejemplo &A y &B, pero que deben tener como tipo de datos uno compatible con el de las variables &fromName y &toName. ¿Por qué? Porque será en estas variables que el usuario ingresará valores que en el evento que programemos serán enviadas por parámetro al procedimiento, así:

AmusementParksFromTo(&A, &B)

BUSINESS COMPONENTS

Realizaremos algunas operaciones sobre la base de datos, a través de *Business Components.*

AUMENTO DE PRECIO DE LAS REPARACIONES

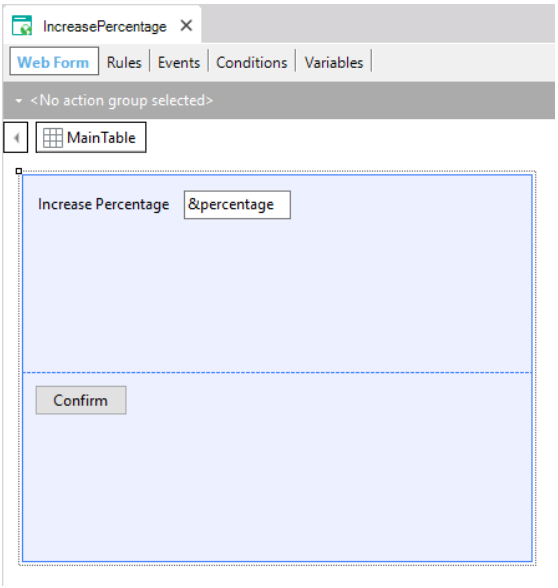
La empresa se reserva el derecho de incrementar los precios de las reparaciones en un porcentaje determinado a partir de cierto momento. A los efectos de realizar una grabación masiva probaremos esta funcionalidad. Para ello, deberemos implementar una **pantalla** que permita al usuario especificar ese **porcentaje de aumento**, y dar la **orden** de aplicarlo a todas las reparaciones de la base de datos. **Impleméntelo** y **pruebe**.

Recuerde que:

- El objeto web panel le permite implementar pantallas flexibles para entrada y salida de información.
- Para el ingreso de información (que el usuario pueda ingresar valores a la pantalla), puede insertar desde la *Toolbar* un control *Attribute/Variable* en el form y asignarle una variable
- Si quiere editar barras de menú, posicionándose en GeneXus arriba, sobre la barra, con botón derecho podrá insertar, por ejemplo, la barra *Formatting*.
- Para que el web panel tome alguna acción determinada, pueden insertarse botones y programar el "evento" asociado.
- Los *Business Components* son tipos de datos que se crean al configurar la propiedad de nombre *Business Component* del objeto transacción con valor *Yes*. Al hacerlo, para insertar, modificar o eliminar registros de las tablas correspondientes, podrá utilizarse, además de la transacción, una variable de tipo de datos *Business Component* en cualquier otro objeto (por ejemplo, un web panel) y realizar las mismas operaciones a través de los métodos *Load(), Save()* y *Delete().*
- Para que las operaciones realizadas a través del *Business Component* queden efectuadas de manera permanente, deberá ejecutar a continuación el comando *Commit*.
- Si a un valor X se le quiere incrementar un 20%, alcanza con hacer X = X*(1+20/100) = X*1,20

Solución:

Una solución posible (intente implementar Ud. lo pedido sin mirar lo que sigue): creamos un Web panel, con una variable &percentage, de tipo de datos: Numeric(3.0)



Al hacer doble click sobre el botón, nos lleva a la sección Events, editando el evento asociado al momento de definir el botón. En nuestro ejemplo es el evento Confirm.

Allí programaremos la lógica que queremos se ejecute cuando el usuario presione el botón.

Necesitamos recorrer todas las reparaciones y sobrescribir el precio que tenían, aumentándolo en ese porcentaje indicado por el usuario en la variable de pantalla.

Para recorrer todas las reparaciones de la base de datos, usamos el comando For each.

¿Y para cada reparación, cómo lo editamos para modificarle el valor de RepairCost?

Necesitaremos una variable para ello, que tenga toda la estructura de la transacción Repair, y además nos permita grabar en la base de datos. ¿Qué tipo de datos tendrá, entonces, esa variable? El Business Component Repair (observar que el tipo de datos Business Component tiene el mismo nombre que la transacción). Pero GeneXus no crea ese tipo de datos automáticamente para cada transacción. Se lo tenemos que pedir. ¿Cómo lo hacíamos? Prendiendo la propiedad correspondiente de la transacción.

Una vez hecho esto, entonces:

```
Event 'Confirm'
    For each Repair
        &Repair.Load(RepairId)
        &Repair.RepairCost = &Repair.RepairCost * (1 + &percentage/100)
        &Repair.Save()
    Endfor
    commit
Endevent
```

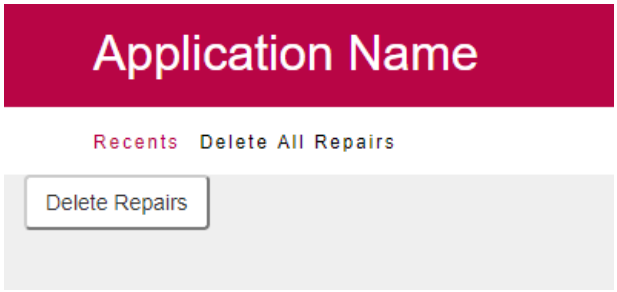
Como las operaciones de grabación o eliminación de la base de datos (a través de los métodos Save() y Delete()) pueden provocar errores, es importante saber qué ocurrió. Para ello tenemos el método Success() que devolverá True si no hubo errores, y False en caso contrario.

```
Event 'Confirm'
    For each Repair
        &Repair.Load(RepairId)
        &Repair.RepairCost = &Repair.RepairCost * (1 + &percentage/100)
        &Repair.Save()
        If &Repair.Success()
            commit
        else
            rollback
        endif
    Endfor
Endevent
```

Al insertar, modificar o eliminar registros de una tabla de la base de datos, mientras que no se diga: "todo lo hecho que quede permanente", esas modificaciones serán provisorias. ¿Qué quiere decir? Que por ejemplo, si hay un apagón, esas modificaciones se perderán. La forma de decir "lo hecho, que quede permanente" es ejecutar el comando Commit. Si por el contrario, queremos que lo que hicimos se deshaga, ejecutamos el comando Rollback.

PANTALLA PARA ELIMINACIÓN DE TODAS LAS REPARACIONES

Grabe el web panel anterior con otro nombre (para ello alcanza con posicionarse sobre la pestaña y con botón derecho, hacer ***Save as***) y modifique el Form para que sólo contenga un botón con el texto *Delete Repairs*, de modo que en ejecución este web panel se vea así:



Cuando el usuario presione el botón, deberán eliminarse todas las reparaciones de la base de datos.

¿Qué debe modificar del evento *Confirm* que tenía programado?

Nota: Si se posiciona sobre el botón y ve sus propiedades, en la de nombre *Caption* puede modificar el texto del botón.

Solución:

```
Event 'Confirm'
    For each Repair
        &Repair.Load(RepairId)
        &Repair.Delete()
        If &Repair.Success()
            commit
            msg("Successful deletion")
        else
            msg("Deletion failed")
            rollback
        endif
    Endfor
Endevent
```

Con *Msg* logrará que se despliegue ese mensaje en la pantalla del web panel.

PROCEDIMIENTOS PARA ACTUALIZAR REGISTROS

AUMENTO DE PRECIOS DE LAS REPARACIONES

Suponga que las reparaciones a las que debe aumentar el precio en un porcentaje dado son miles. Sabiendo que el aumento de precio es un procedimiento sencillo que no hará fallar la integridad de ningún modo, ejercite resolverlo con un procedimiento, sin utilizar *Business Components*.

Recuerde que:

- Con el comando *for each* dentro de un procedimiento, puede actualizar los atributos de su tabla extendida a través de simples asignaciones.
- La actualización "directa" a través de procedimientos no controla la integridad de la información.
- Todo objeto debe declarar los parámetros que recibe y los parámetros que devuelve. Si no los declara, ni recibirá ni devolverá valores.
- Los parámetros se declaran a través de la regla ***parm***.
- Las variables son locales al objeto donde se utilizan. Esto significa que si quiero recibir un valor como parámetro en una variable &X, debo declararla en el objeto.

ELIMINACIÓN DE TODAS LAS REPARACIONES

¿Y si ahora quisiera eliminar todas las reparaciones, tal como se hizo anteriormente en el práctico, pero esta vez a través de un procedimiento?

Solución:

Crear un procedimiento *RepairsDeletion* que no reciba parámetros, con el siguiente código:

```
for each Repair
   delete
endfor
```

Haga un *Save as* del web panel que tenía implementado para hacer la eliminación a través de *Business Component*), y cambie el evento *Enter*:

```
Event Enter
    RepairsDeletion()
EndEvent
```

¿Y si ahora quiere borrar toda la información de la base de datos?

Solución

Crear un procedimiento DeleteAll con el siguiente Source:

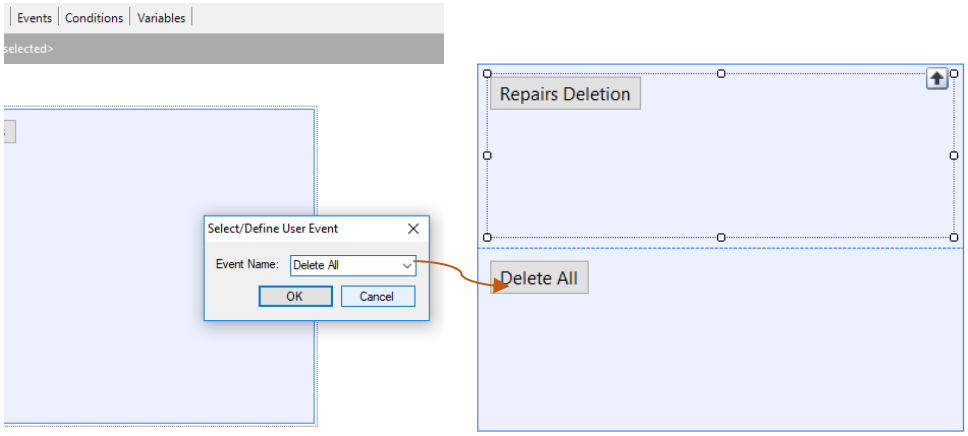
```
for each Repair
    for each Repair.Kind
      delete
    endfor
    delete
endfor

for each Technician
    delete
endfor

for each Game
    delete
endfor

for each Category
    delete
endfor

for each Employee
    delete
endfor

for each AmusementPark
    delete
endfor

for each Country
    for each Country.City
      delete
    endfor
    delete
endfor
```

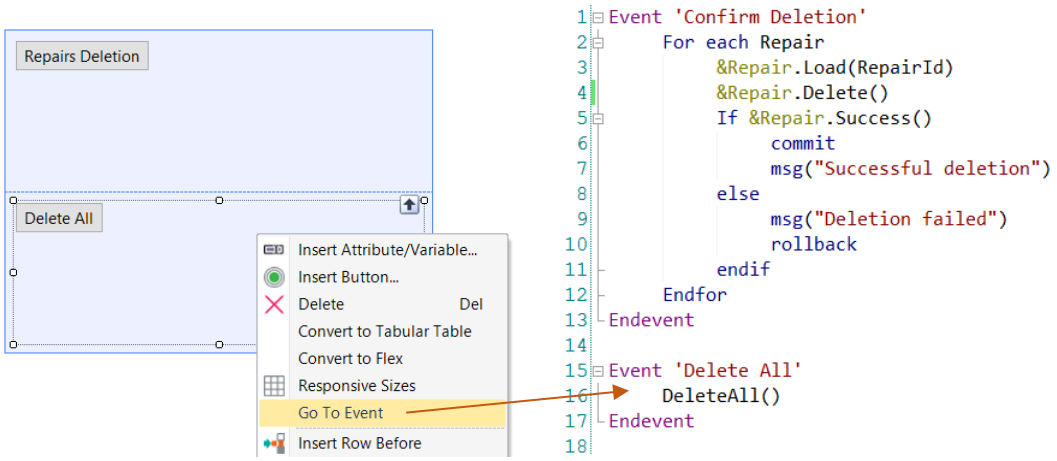
Recuerde que los procedimientos no validan la consistencia de los datos, pero la base de datos sí lo hace. Es decir, la base de datos valida la consistencia de los datos interrelacionados, por lo tanto, el orden en el que Ud. intente eliminar los datos, es importante.

Por ejemplo, si intenta eliminar los países antes que los parques, la base de datos lo impedirá, el programa cancelará y no resultará amigable para el usuario.

En el mismo web panel que se utilizó para eliminar todos los registros de Repair utilizando Business Components, agregue un nuevo botón, al cual le asociaremos un "evento de usuario", y haremos la llamada al procedimiento.

Arrastre el control button sobre el form, y defina el nuevo evento.





Luego, presionando el botón derecho del mouse sobre el botón y seleccionando Go To Event, nos posicionamos en el evento asociado al botón y le definimos dentro la invocación al procedimiento.

INICIALIZACIÓN DE LA INFORMACIÓN DE LA BASE DE DATOS [OPCIONAL]

Cuando la aplicación que usted está desarrollando se ponga en producción (es decir, empiece a ser utilizada por la empresa) deberán cargarse todos los datos de países, parques, categorías, juegos, técnicos, reparaciones. Utilice las facilidades que el objeto *Transacción* ofrece para esto e inicialice esas tablas con información que nos ha brindado la empresa y pruébelo.

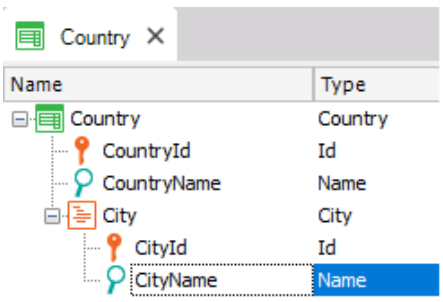
Tener en cuenta que:

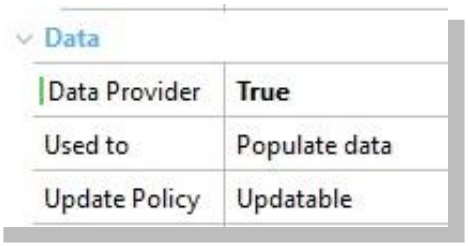
- Las transacciones cuentan con una propiedad ***Data Provider*** que nos permite que dispongamos de un data provider asociado a la transacción, de modo que al crearse la tabla en la base de datos, se inicialice con los valores que definamos en dicho data provider. Para eso deberá configurar también la propiedad ***Used to*** y ***Update Policy***.
- Hay que insertar las imágenes en la KB para poder utilizarlas. Una forma de hacerlo es en la ventana *KB Explorer* ir al nodo *Customization* y eligiendo *Images*, donde se listan todas las imágenes actualmente almacenadas en la *KB*, insertar una nueva (desde un archivo) y darle un nombre.

Solución

A modo de probar diferentes opciones, poblaremos los países y las categorías utilizando la propiedad Data Provider de las transacciones.

- Abra la transacción Country y configure el valor True a la propiedad Data Provider:





- Cargue los datos deseados en Country_DataProvider (puede verlo bajo la transacción Country en la ventana KBExplorer). Al momento de crearse, la tabla se cargará con los datos indicados.

Recordar:
El *DP* asociado a la transacción se volverá a disparar

- Cada vez que se reorganice la tabla correspondiente.
- Cada vez que se edite el contenido del *DP*.

Por lo tanto, es necesario controlar que la información no se duplique. Esto se puede lograr:

- Si la llave primaria es autonumerada, se puede definir un índice *unique* sobre el atributo descriptor.
- Dejando la llave primaria sin autonumerar. De esta forma su valor se asignará en forma fija al definir el *DP* y no se duplicará.

- En forma análoga cargue las categorías.

WEB PANELS

Se solicita una página que muestre todos los países y para cada uno, la cantidad de parques de diversiones que ofrece visitar.

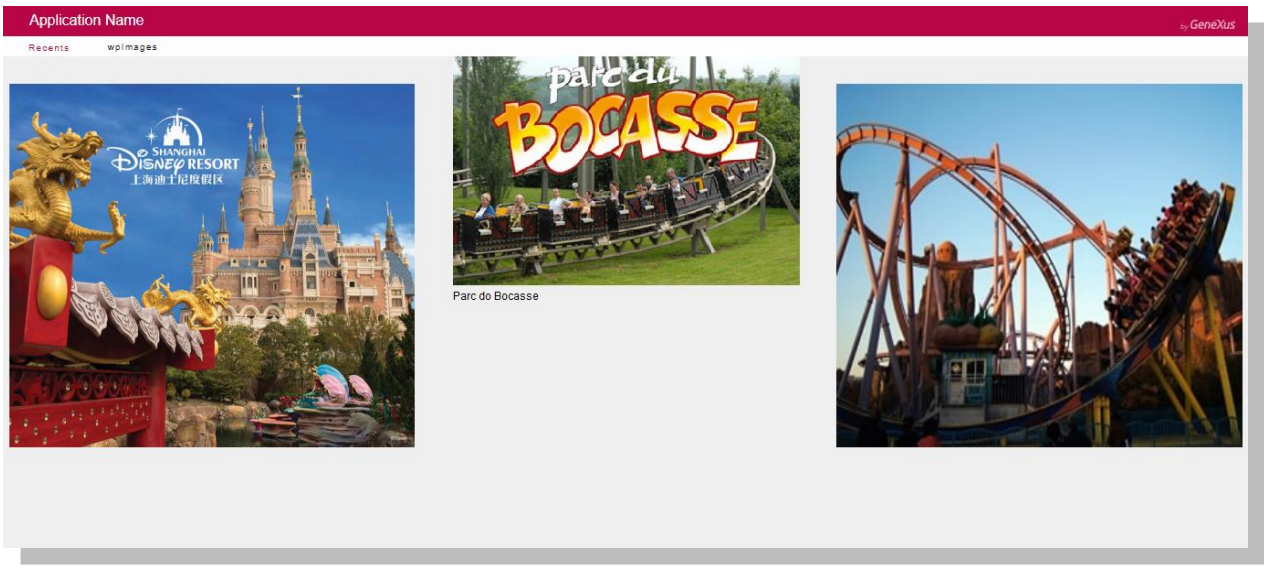
Recuerde que el **evento *Load* en un web panel con tabla base que tiene un grid,** se ejecuta justo antes de cargar cada línea en el grid. Este evento es adecuado para asignar a la variable, el cálculo que devuelve la cuenta de ciudades de cada país que se está navegando y a punto de cargar en una línea del grid.

Agregue ahora al web panel definido, dos variables (&*CountryNameFrom* y &*CountryNameTo*) y defina las condiciones necesarias para filtrar los países incluidos en dicho rango.

EXTENDED CONTROLS

Utilizando el *extended control ImageGallery*, diseñe un web panel que muestre la galería de fotos de todos los parques de diversiones.

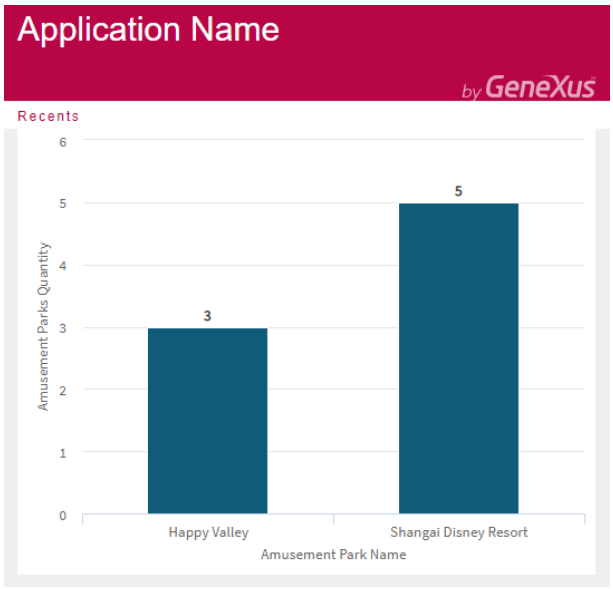
Personalice las propiedades del *extended control*, estableciendo Width=1000, Height= 500, y Type=Slider:



OBJETO QUERY

Defina un objeto *Query* que devuelva solamente los parques de diversiones de China, ordenados por orden alfabético, cada uno de ellas con su respectiva cantidad de juegos.

Defina un web panel y utilizando el extended control *QueryViewer*, visualice la consulta anterior como un gráfico.

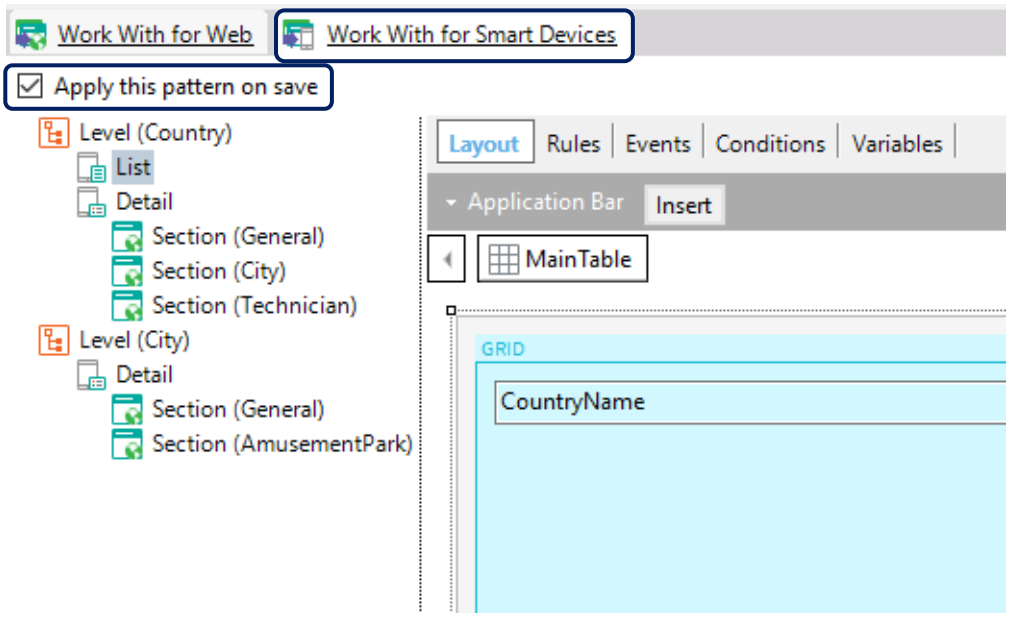


PARTE PARA SMART DEVICES

La empresa desea ofrecer también una pequeña aplicación para dispositivos inteligentes para ser utilizada por los usuarios finales.

El objetivo es que cualquier persona pueda consultar desde su smart device todos los países que puede visitar, y para cada uno de ellos sus parques de diversiones y juegos.

Para ello, hay que aplicar el patrón *Work With for Smart Devices* a la transacción *Country*:



Y simplemente grabar, para posteriormente crear un objeto *MenuforSmartDevicces* y agregarle el objeto de nombre: *WorkWithDevicesCountry* generado por el patrón.

Recuerde que por el hecho de haber creado dentro de nuestra base de conocimiento, objetos propios para Smart Devices, al presionar F5 automáticamente se ejecutará el emulador para *Android*, pudiendo también acceder a nuestra aplicación web desde el *Developer menu*.

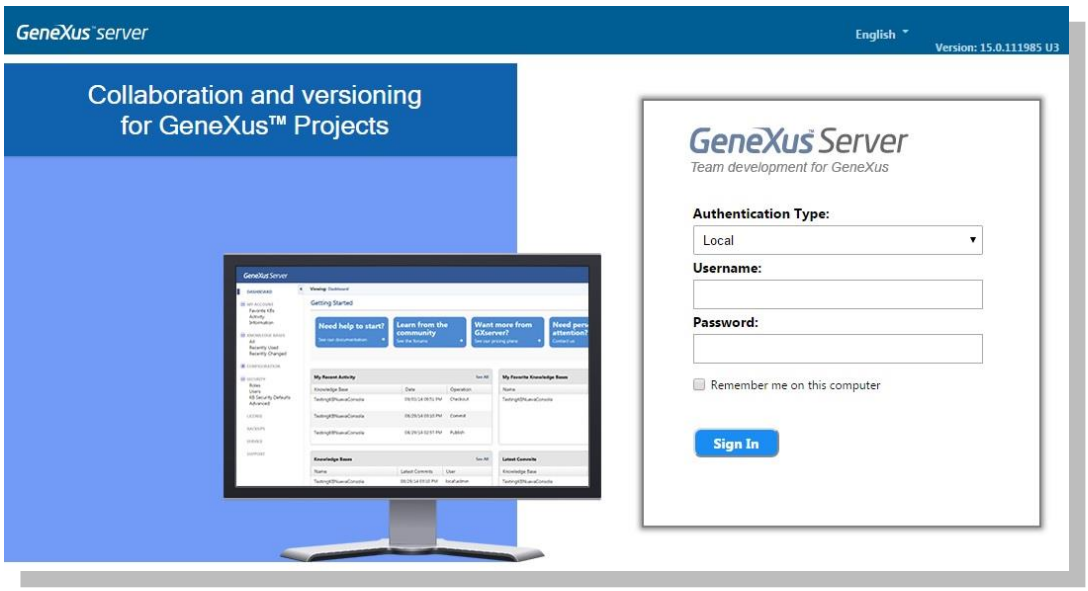
GENEXUS SERVER

Publique la base de conocimiento en el servidor http://sandbox.genexusserver.com/v16

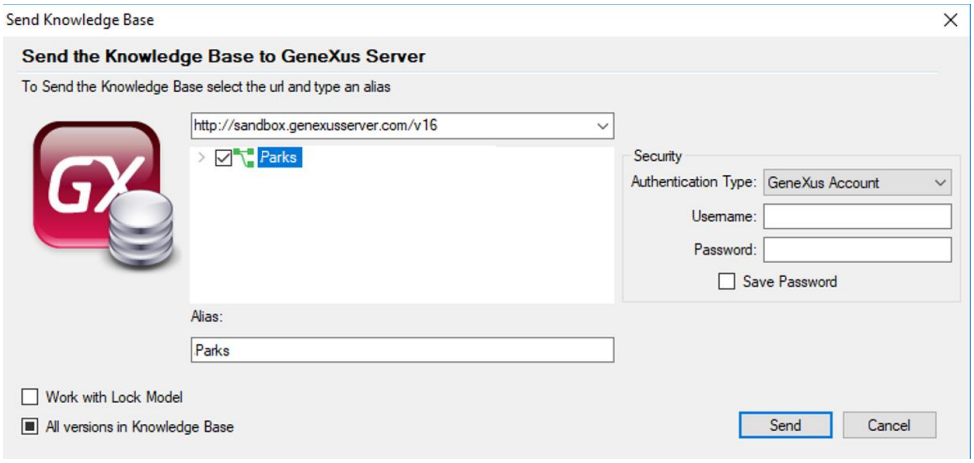
Cree un nuevo web panel que muestre la lista de parques de diversiones registrados.

Envíe este nuevo objeto al server para que se integre a la base de conocimiento centralizada.

Ingrese a la consola web y verifique el estado final de la *KB*.



Cierre la base de conocimiento anterior, y cree una nueva sincronizándose con la *KB* previamente publicada. De esta forma se recibe localmente una copia de la *KB* administrada por GeneXus Server.



En esta nueva copia local, edite la transacción *Country* y defina el nuevo atributo *CountryFlagImage*, de tipo *Image*. Envíe este cambio al servidor.

Cierre esta *KB* y abra nuevamente la *KB* inicial. Realice la operación *Update* para recibir el cambio realizado anteriormente.
