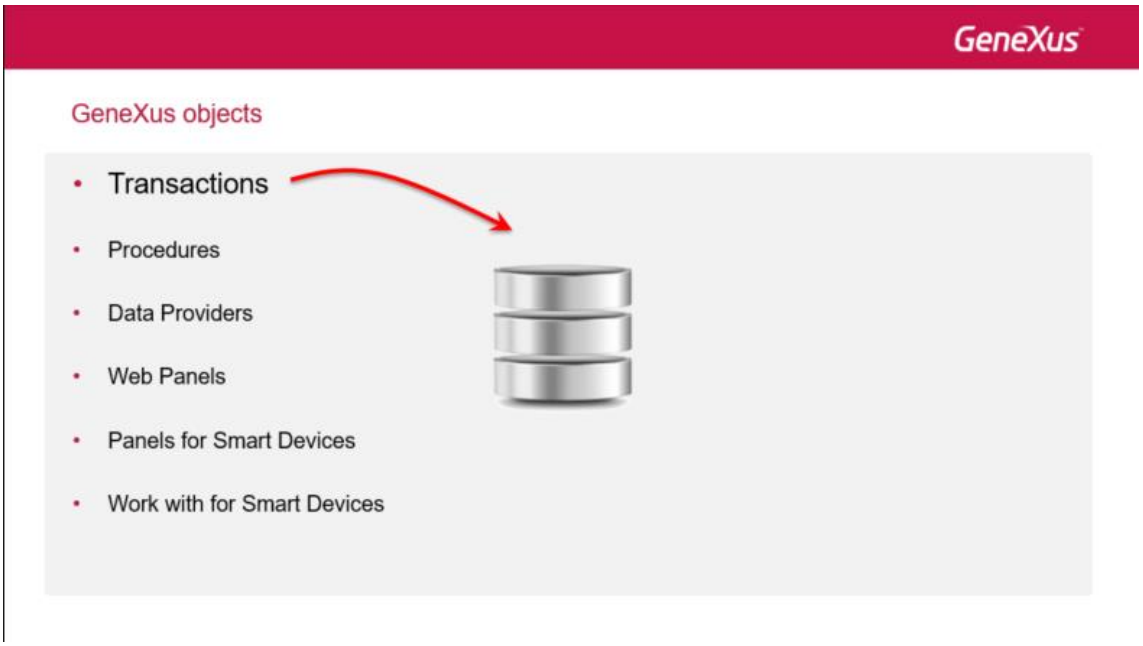
Overview final

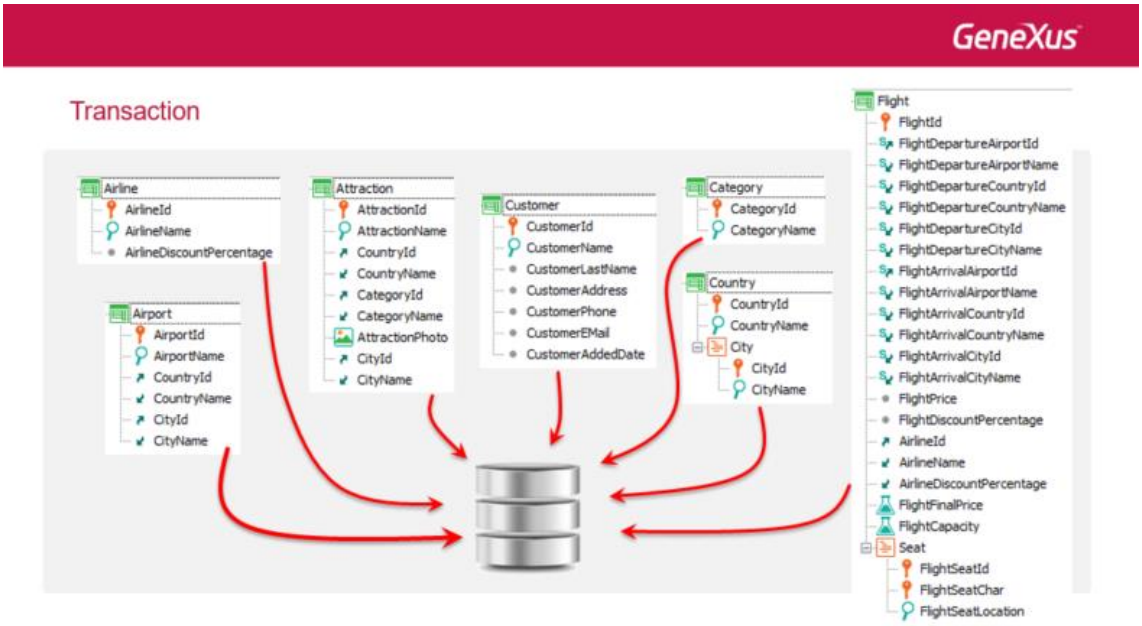
A lo largo de este curso nos hemos detenido en los principales objetos GeneXus que permiten implementar las funcionalidades más importantes que hacen a una aplicación web y mencionamos también los que implementan una aplicación para dispositivos inteligentes.

Vimos que a partir de los objetos de tipo Transacción definidos en la base de conocimiento se construía el modelo de datos…



… es decir, las entidades de la realidad con sus atributos y la forma en que se relacionan entre sí, que se reflejaba luego en la creación de una base de datos, con sus respectivas tablas. Esta creación, así como sus sucesivas reorganizaciones quedaban a cargo de GeneXus, liberando a

los desarrolladores de pensar a nivel físico, es decir, a nivel de tablas, permitiéndoles concentrarse en las visiones de los datos a más alto nivel, que es el nivel de las transacciones y sus atributos, justamente.

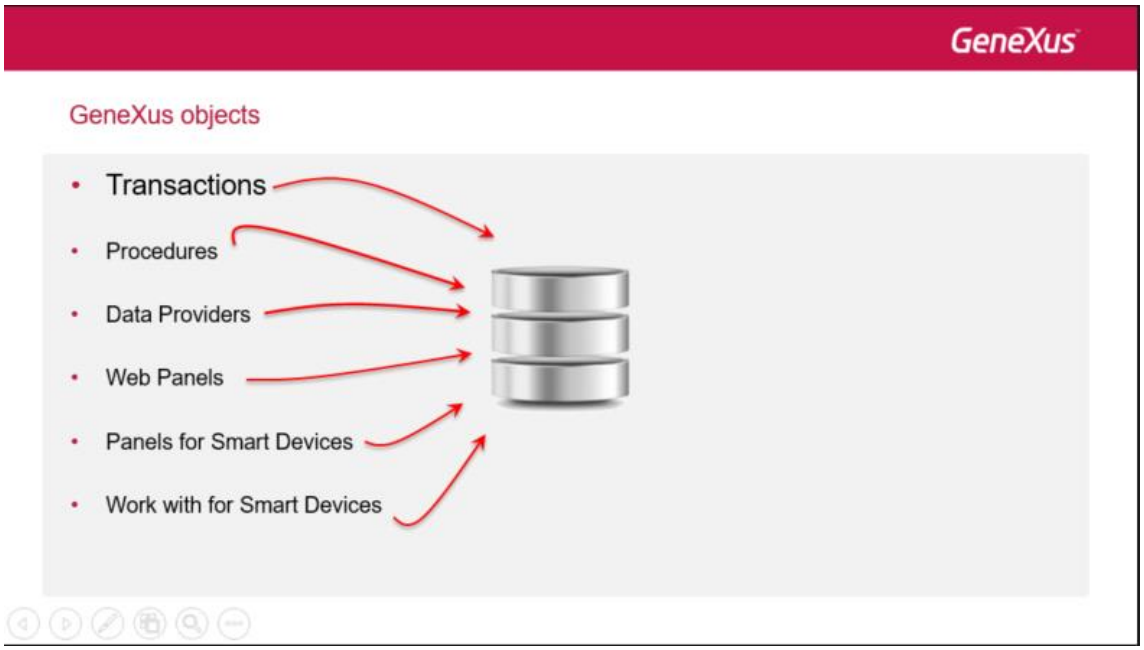


Todos los demás objetos que aquí podemos ver, podían acceder luego a las tablas de la base de datos, para consultar o para además actualizar su información. Pero los detalles de este acceso quedan enmascarados en GeneXus. Él es quien se encarga de relacionar atributos y transacciones con tablas y campos de base de datos. En este curso mostramos cómo lo hace, pero no es necesario saberlo en absoluto.

Así, si bien acabamos de decir que la filosofía de GeneXus es olvidarnos de las tablas y pensar a nivel de atributos y niveles de transacción, a lo largo del curso hemos ido a las tablas de la base de datos una y otra vez para explicar cada uno de los conceptos que hemos abordado. Esto contraviene esta filosofía, que consiste, insistimos, justamente en lo contrario: independizar al desarrollador de los detalles del almacenamiento y de la implementación, para permitirle pensar y expresarse a más alto nivel, como lo haría cualquier persona que no sabe demasiado de informática pero sí puede expresar una realidad que está queriendo modelar. Esa forma de expresión sólo requiere de las transacciones y de los atributos, nunca de las tablas. ¿Por qué entonces fuimos a las tablas una y otra vez? Fue por motivos básicamente didácticos.

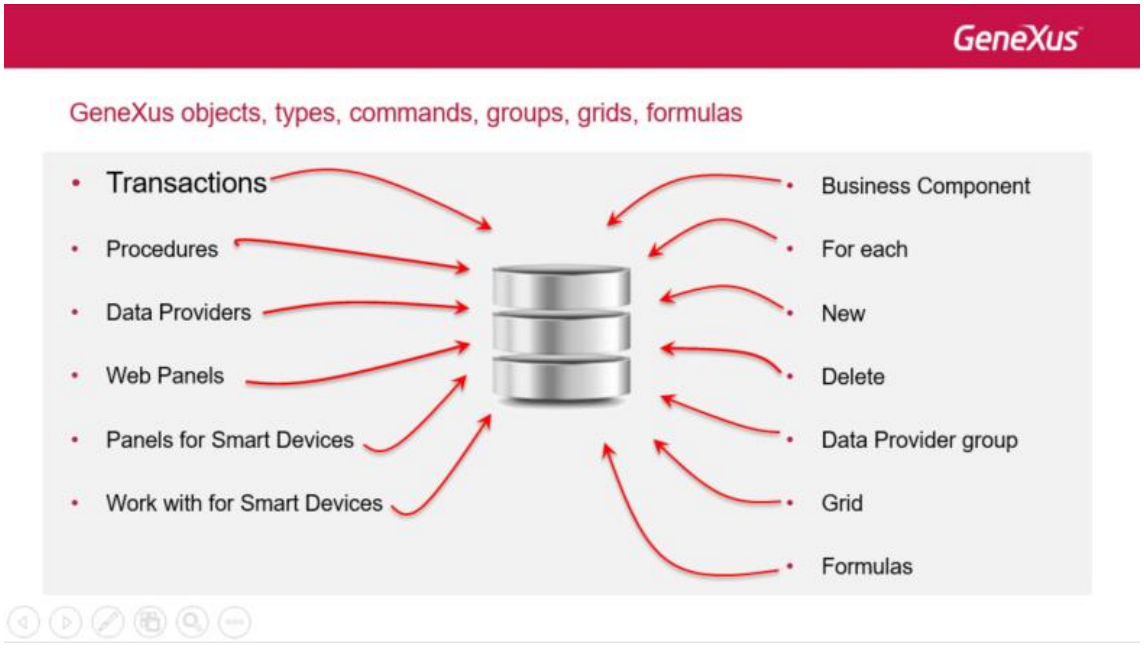
Pero llegó el momento de cuestionar esto.

Si bien todos estos objetos podían acceder a los datos de la aplicación, en ninguno de ellos, para hacerlo, nombrábamos tablas explícitamente. Lo que hacíamos era nombrar atributos y transacción base. Así se espera que el desarrollador trabaje.



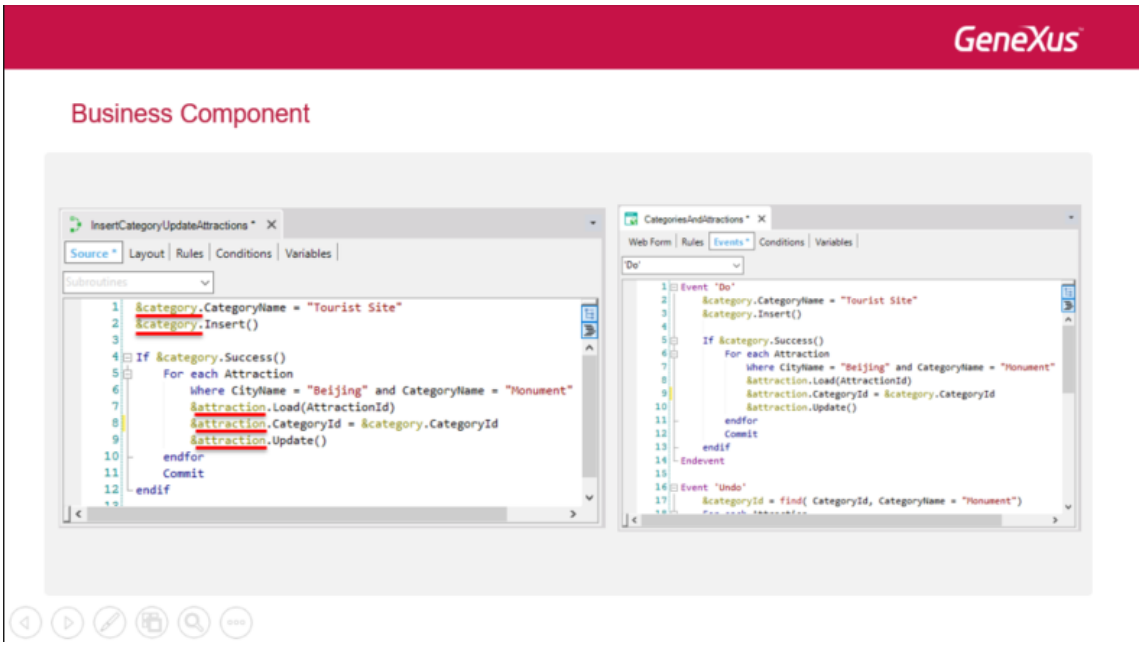
Aquí agregamos los tipos de datos, comandos, etc. que permitían en los distintos objetos acceder a los datos, tanto para sólo consultarlos, como para también actualizarlos.

Los **Business Components** eran una suerte de tipos de datos estructurados que se construían a partir de las transacciones, quedándose con su estructura de datos y con sus reglas y eventos, pero no con la pantalla ni nada de lo asociado a ella.

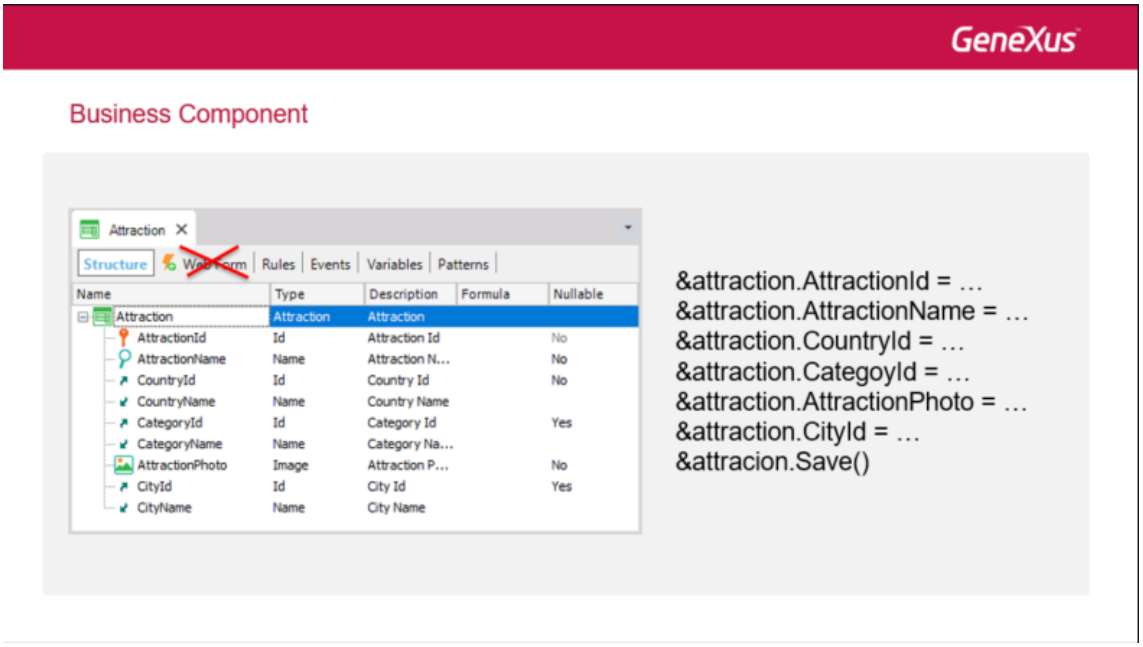


De esta manera, los Business Components se asociaban a variables y permitían a través de métodos el trabajo con la base de datos sin necesidad de estar pensando en las tablas físicas. Eran como un espejo de las transacciones, pero operando por código.

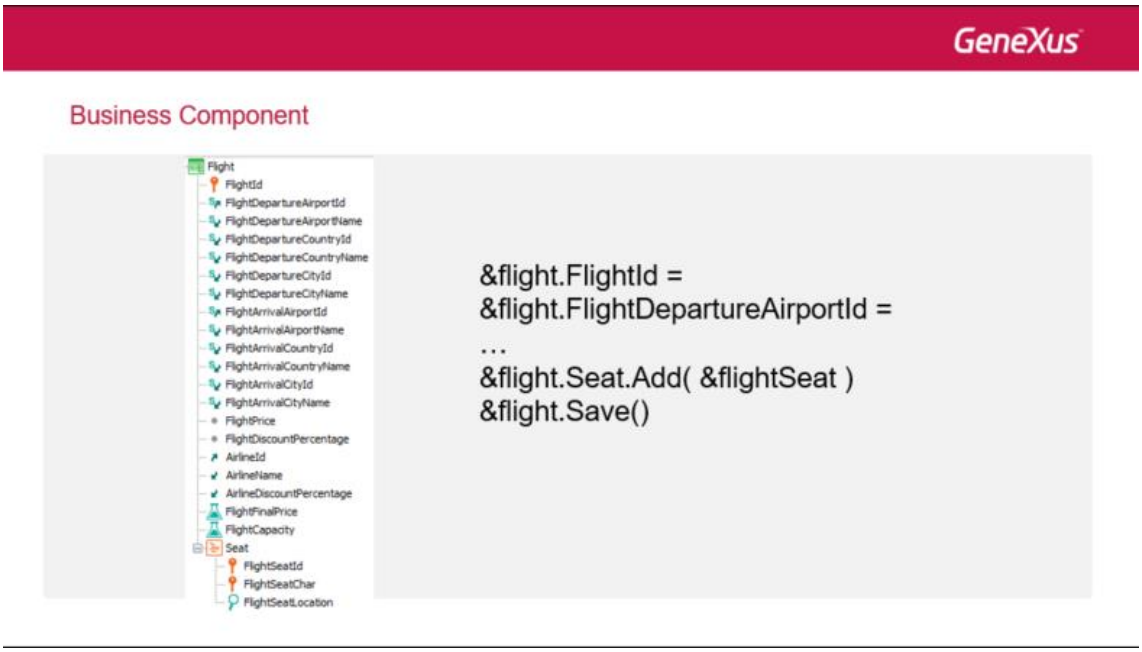
Por ser utilizados a través de variables, podían usarse en varios objetos GeneXus (por ejemplo en el Source de procedimientos o en eventos de Web Panels).



Insertar información utilizando un business component es exactamente igual a hacerlo a través de la transacción asociada (aunque sin su pantalla). Recordemos que se controlará la integridad de los datos, y también se ejecutarás las reglas de la transacción.

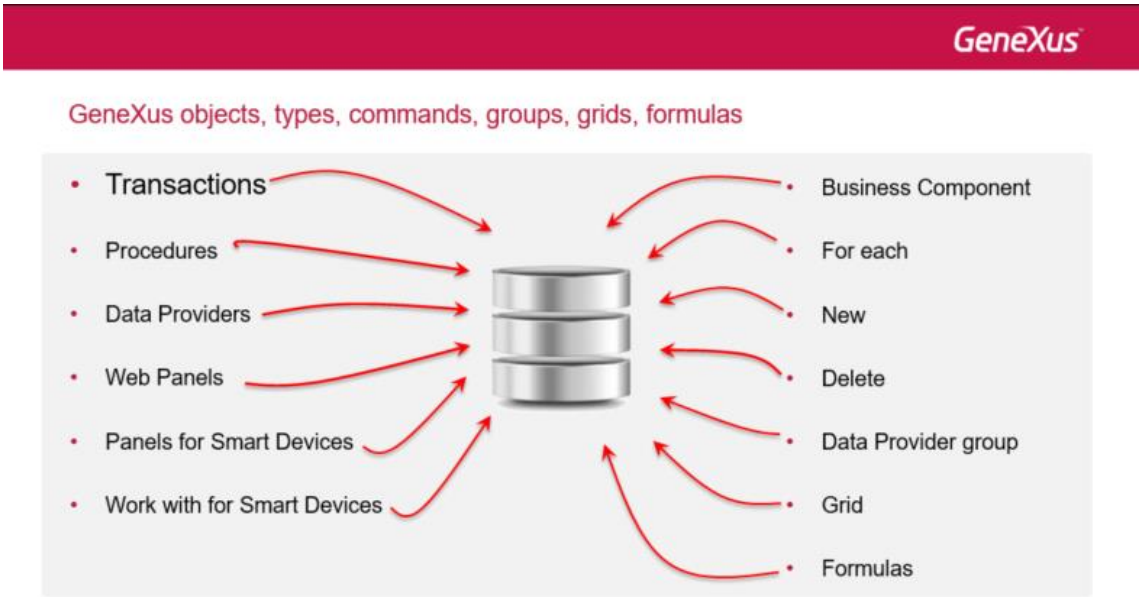


Si bien no lo vimos en este curso, un business component puede ser de dos niveles, si la transacción lo es. Como ejemplo, pensemos en la transacción Flight.

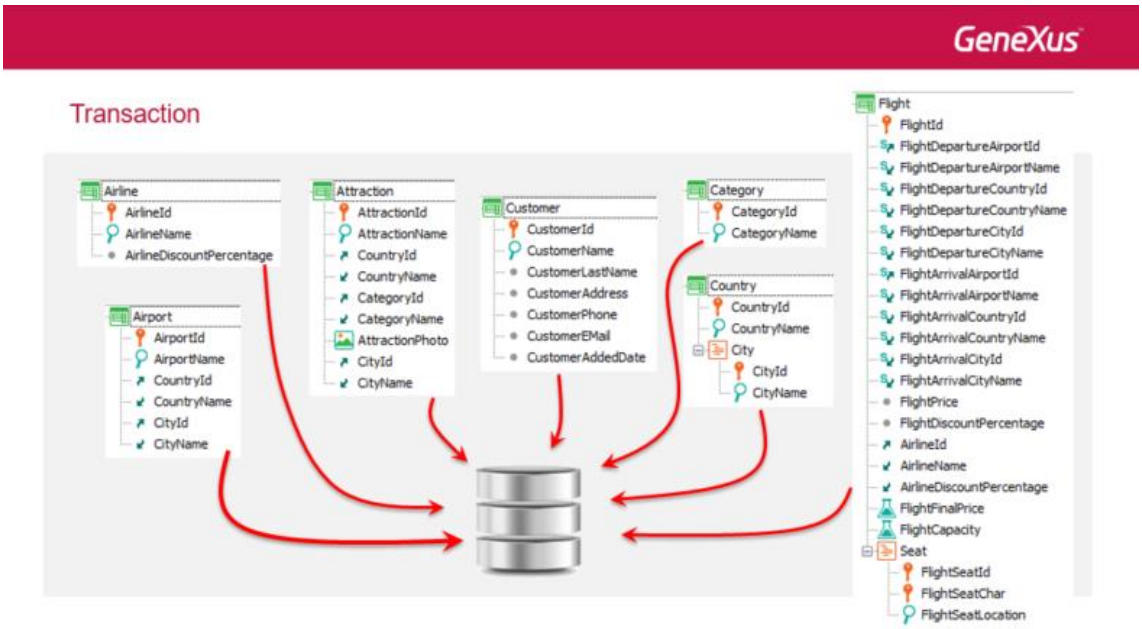


El comando For each es uno de los principales comandos de GeneXus, pues puede utilizarse prácticamente en cualquier objeto.

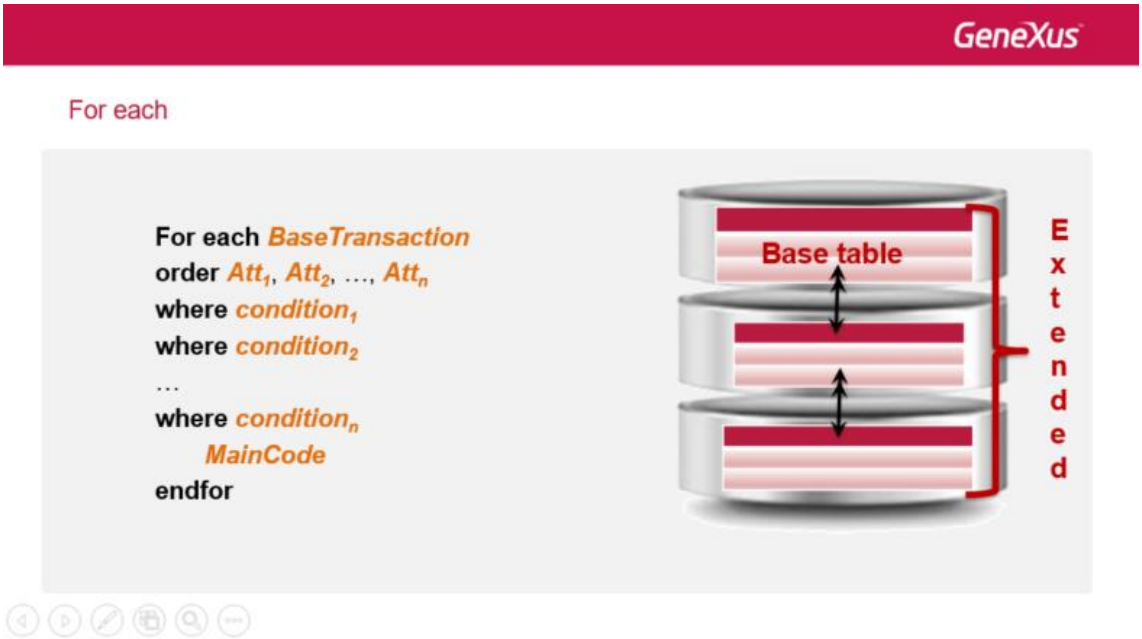
Su propósito es recorrer cada elemento del nivel de una transacción, esto es, de un actor de la realidad, si por actor pensamos en cada nivel de la transacción.



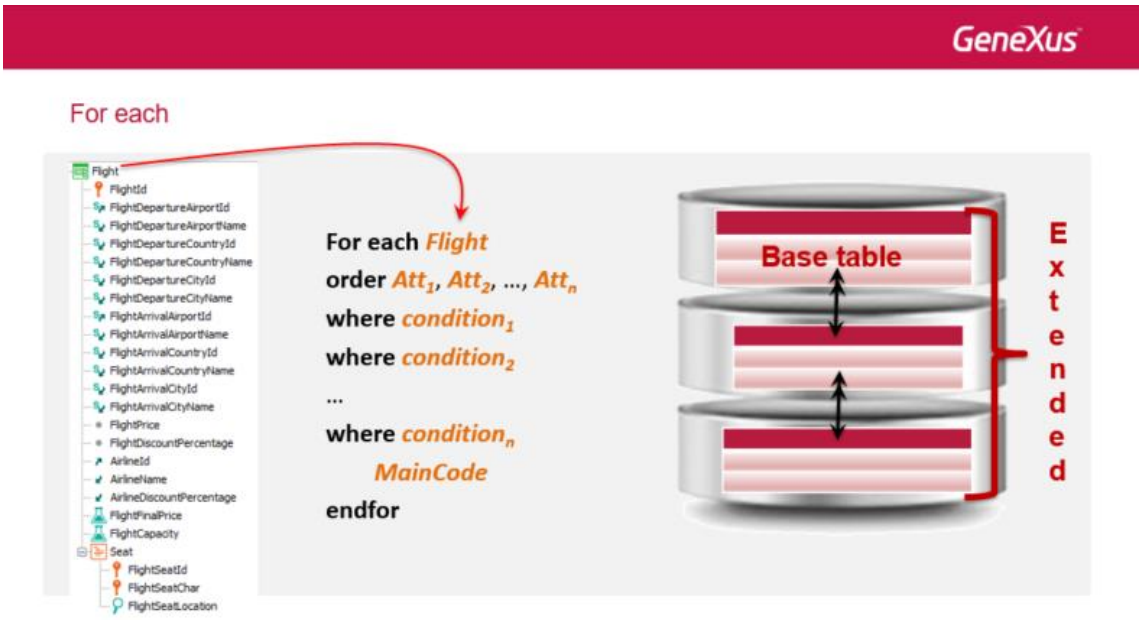
Por ejemplo, cada atracción turística, o cada vuelo, o cada asiento de vuelo, o cada país, o cada ciudad, etc., para hacer algo con su información.



Cuando lo estudiamos vimos que lo que indicábamos era la transacción base, nunca la tabla asociada. Esa tabla base asociada era inferida por GeneXus y era importante únicamente para saber toda la información con la que podíamos contar a partir de ella: es decir con la tabla extendida, un concepto, este sí, fundamental.



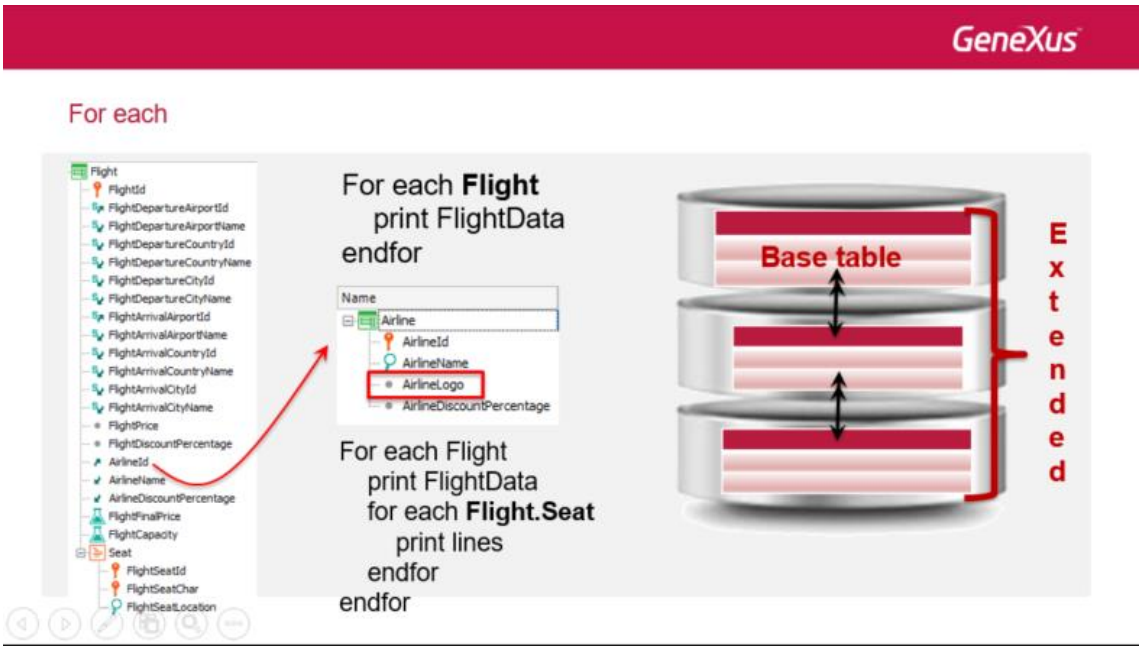
Dicho de otro modo, al For each le indicamos el nivel de la transacción con el que queremos trabajar, y de allí se deduce la información extendida, es decir, información que tal vez no se haya incluido en la propia estructura de la transacción, pero que por las relaciones entre entidades, es “alcanzable”. Por ejemplo…



…si la aerolínea tuviera un atributo que almacena el logo, aunque no lo hayamos especificado en la estructura de la transacción Flight porque allí no lo necesitábamos para nada, AirlineLogo es perfectamente "alcanzable" por un for each con transacción base Flight. Dicho de otro modo: para cada flight, se encuentra el logo de **su** aerolínea. Y por ejemplo, puede imprimirse junto con el resto de la información del vuelo. A esto, y no a otra cosa corresponde el importante concepto de **tabla extendida**.

Entonces en un for each, lo verdaderamente importante no es tanto la tabla base como la tabla extendida que se obtiene a partir de ella, que a su vez se obtiene a partir de la única indicación que realiza el desarrollador: la transacción base.

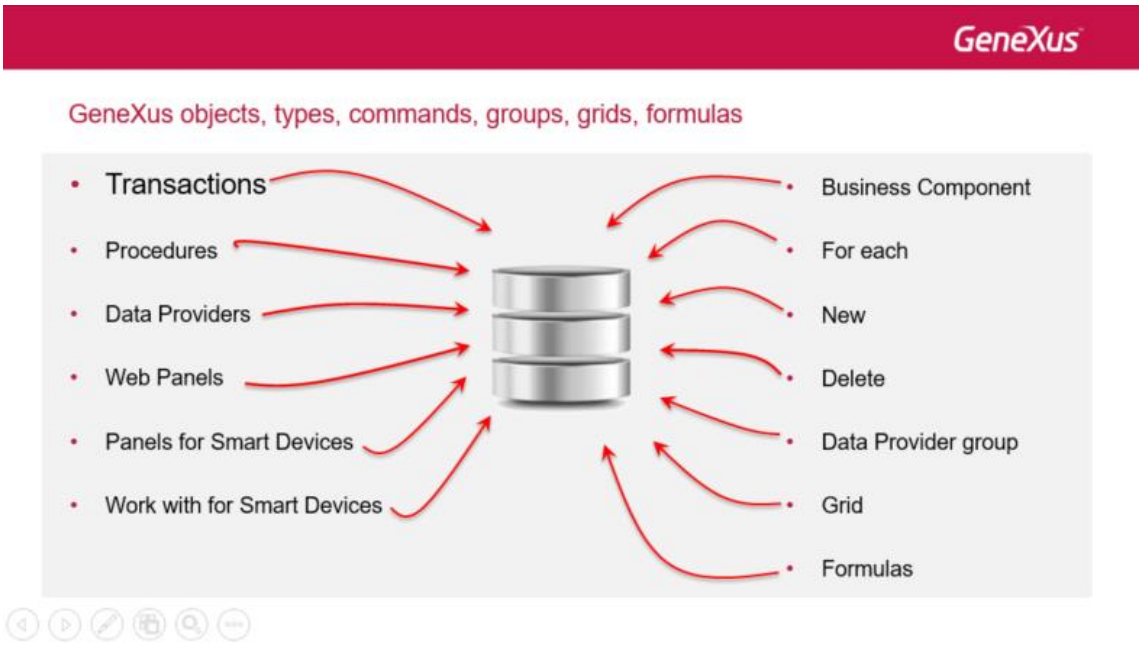
Sí es importante destacar que con un for each como el anterior sólo navegamos los cabezales de los vuelos, y no sus líneas. Las líneas no son alcanzables por el for each. Para navegar las líneas, deberemos anidar otro for each, con transacción base Flight.Seat.



Los comandos New y Delete fueron apenas introducidos y son los de más bajo nivel, puesto que no trabajan a nivel de tabla extendida, sino de tabla base. El new inserta un registro en una tabla y el Delete lo elimina. Es por ese motivo, por su bajo poder de abstracción y expresividad, así como por lo poco integrados que se mantienen en relación a las reglas de las transacciones, que aconsejamos utilizar business components en lugar de estos comandos.

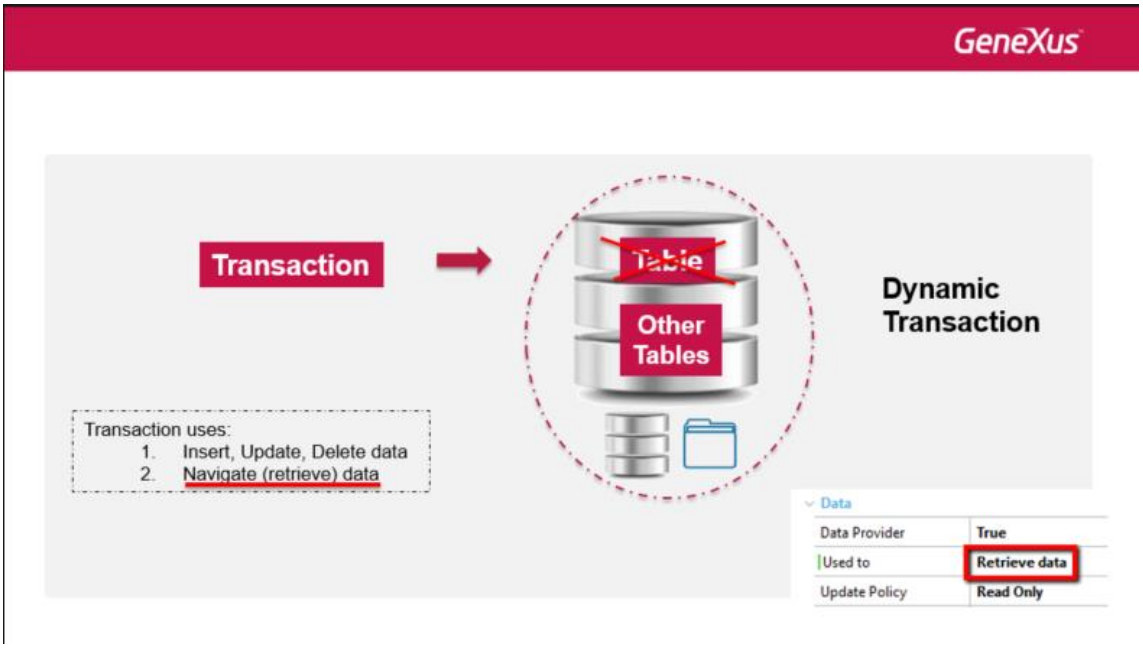
Luego, un grupo de Data Provider es análogo, en todo, a un for each. También lo es un grid con atributos.

Respecto a las fórmulas, también en ese caso vimos que jamás nombramos una tabla de la base de datos. Utilizamos siempre atributos, y será a partir de ellos que GeneXus encuentre qué información es la que se quiere navegar y cómo. Los desarrolladores debemos seguir pensando en alto nivel, de acuerdo a nuestra realidad, que es la creada por las transacciones y sus atributos.



Pero hay más: ya habíamos visto que las transacciones pueden crear tablas físicas, lo que será el comportamiento usual, o podrán no crearlas, y tomar los datos declarados en su estructura de otras tablas, de otras bases de datos externas, o de otras fuentes. Solamente mencionamos el caso de las transacciones dinámicas. Estas transacciones serán en todo como las usuales, sólo que los datos que veremos en la pantalla, y por los que podremos navegar del modo usual, serán cargados a través de un Data Provider, cada vez. Para el usuario esto será transparente.

Pero en alguna medida también lo será para el desarrollador….



…si Attraction fuera una transacción dinámica cuyos datos se toman a partir de servicios externos, por ejemplo, nada nos impedirá seguirla usando como si fuera una transacción común y corriente, con tabla física asociada.

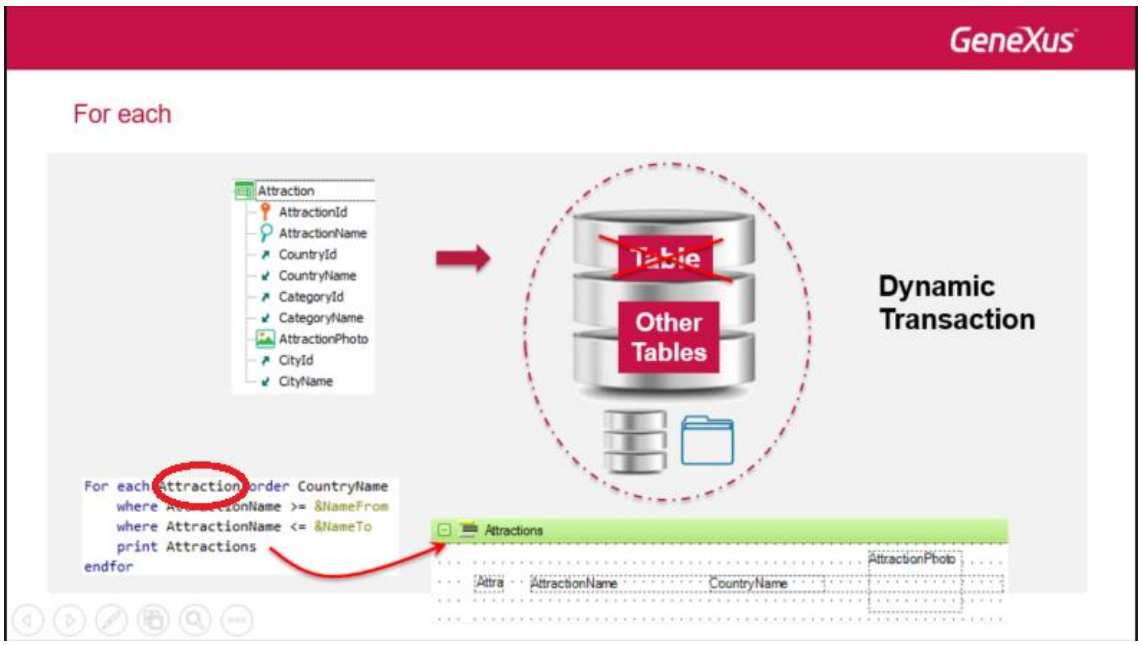
Así, si queremos un for each para listar las atracciones, seguiremos implementándolo exactamente igual que antes.

Con esto ya podemos entender mejor a qué nos referimos cuando decimos que en GeneXus nos independizamos de la parte de la implementación física. Trabajamos a un nivel más alto, con la información desde un punto de vista más conceptual, esté almacenada donde esté almacenada.

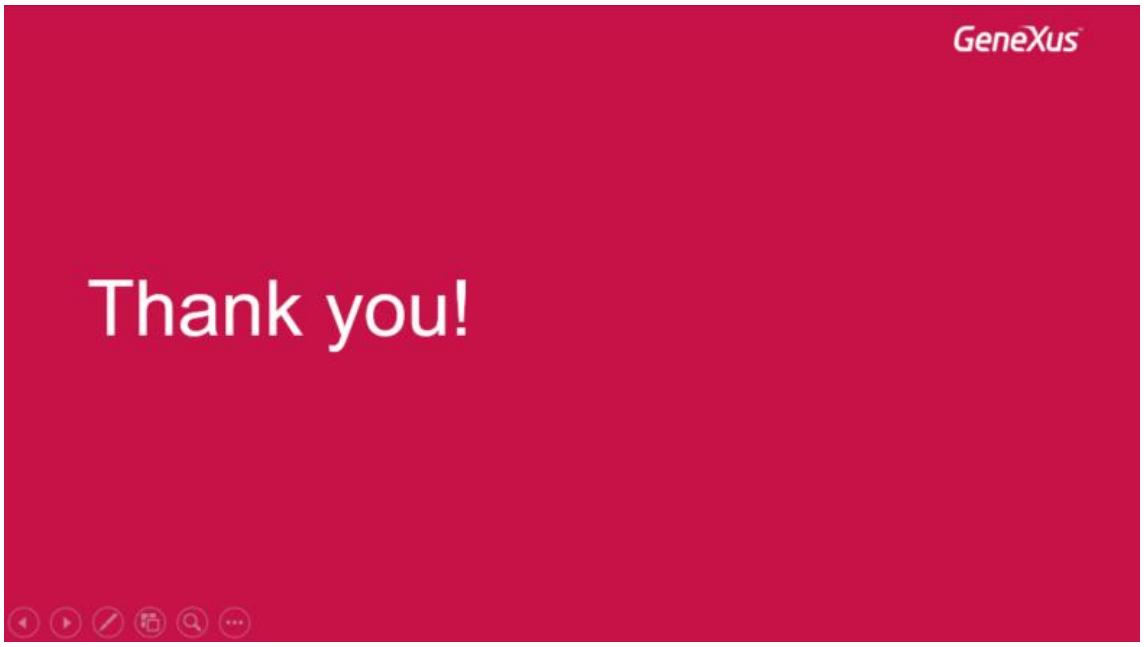
Por tanto, cada vez que hablamos en el curso de "tabla física" fue para facilitar, en ese momento, el entendimiento del tema. Ahora sabemos que esto no tiene por qué ser así, y que las tablas pueden ser una entidad virtual, que nos sirve para pensar, pero cuya implementación es variable. Dónde están los datos en realidad, poco nos importa a este nivel.

De hecho, gran parte de las aplicaciones que desarrollemos crearán y manejarán su propia base de datos, pero otra parte no. Hay aplicaciones que se construyen para trabajar con datos provistos por bases de datos externas, o incluso por datos provistos por terceros, sin saber cuáles son sus fuentes de almacenamiento.

GeneXus está preparado para todo eso, pues su lenguaje es de muy alto nivel. Y eso es lo que quisimos mostrar en este video.



Gracias por acompañarnos. Le sugerimos continuar con los materiales más avanzados. Lo esperamos.


GeneXus
Thank you!


GeneXus™
Videos
training.genexus.com
Documentation
wiki.genexus.com
Certificactions
training.genexus.com/certifications