

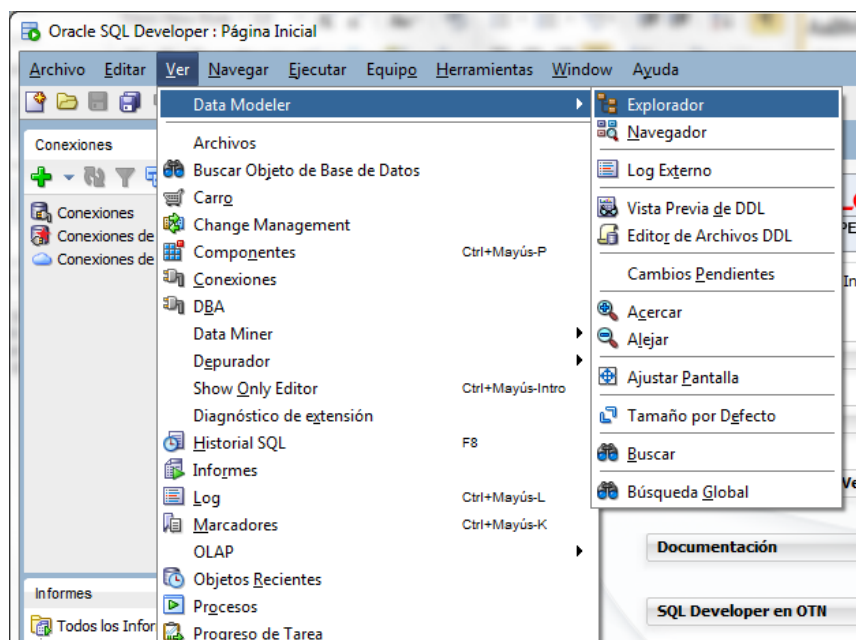
# SQLdeveloper y Datamodeler

**Datamodeler** va a dedicarse a crear lo que llamaremos el modelo lógico y relacional de nuestros problemas. Mientras que con **SQLdeveloper** vamos a introducir los datos dentro de nuestra base de datos.

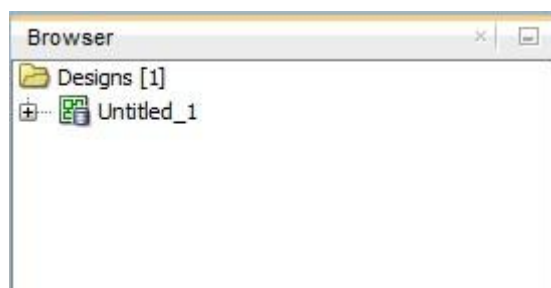
Una vez instaladas nuestras herramientas vamos a aprender a usarla para crear las relaciones que se han explicado teóricamente. Además, vamos a ver como se implementa la relación en el ordenador y que consecuencias crea. Es interesante ir mirando la teoría a medida que se sigue el ejemplo ya que es vital entender qué se está haciendo.

## Modelo lógico

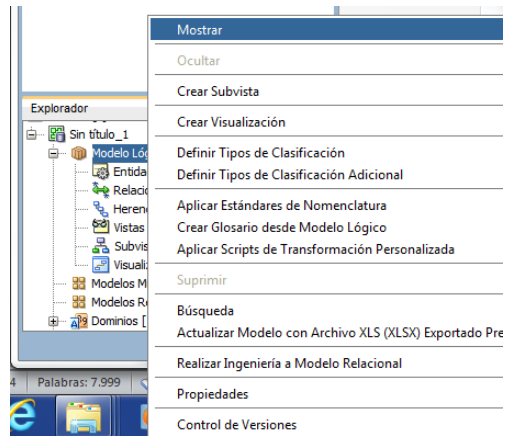
Para empezar a crear un modelo lógico lo más intuitivo sería seleccionar Archivo, Nuevo, pero debemos recordar que tenemos dos programas en uno, por lo que debemos buscar donde está Datamodeler, que será la herramienta que nos ayude a crear el modelo lógico y relacional. Por tanto, en vez darle a Archivo, Nuevo debemos ir a Ver, DataModeler, Explorador como se observa a continuación



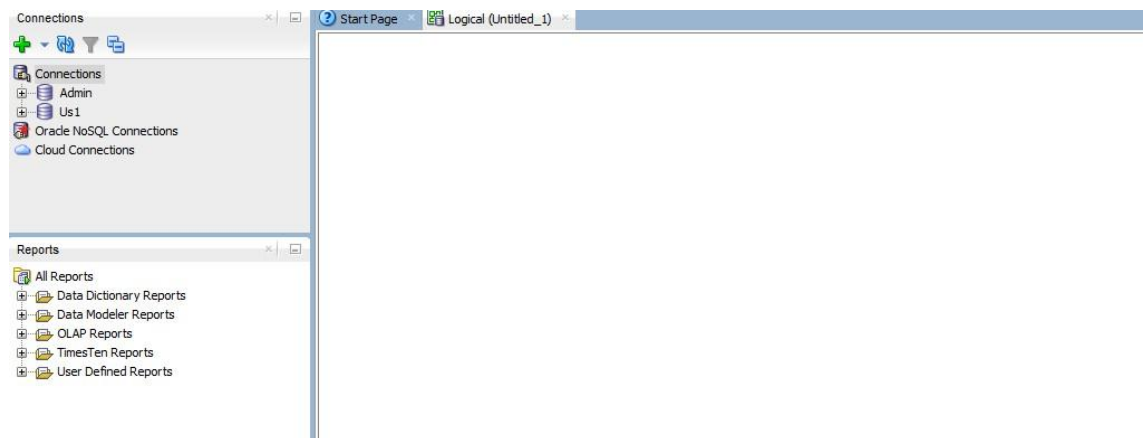
Gracias a esta operación se nos ha creado la zona 4 que se comentaba en el documento de instalación.



En la imagen anterior podemos ver que al lado del Untitled\_1/Sin Titulo\_1 hay un + que nos desplegará un menú. En este menú, dentro de Modelo Lógico le damos a Mostrar

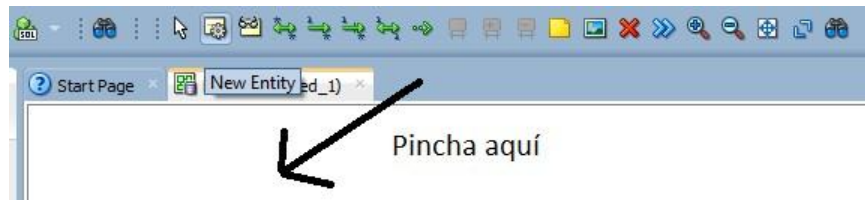


Al darle se nos crea una subventana en la zona 5 con el nombre de logical (Sin título).

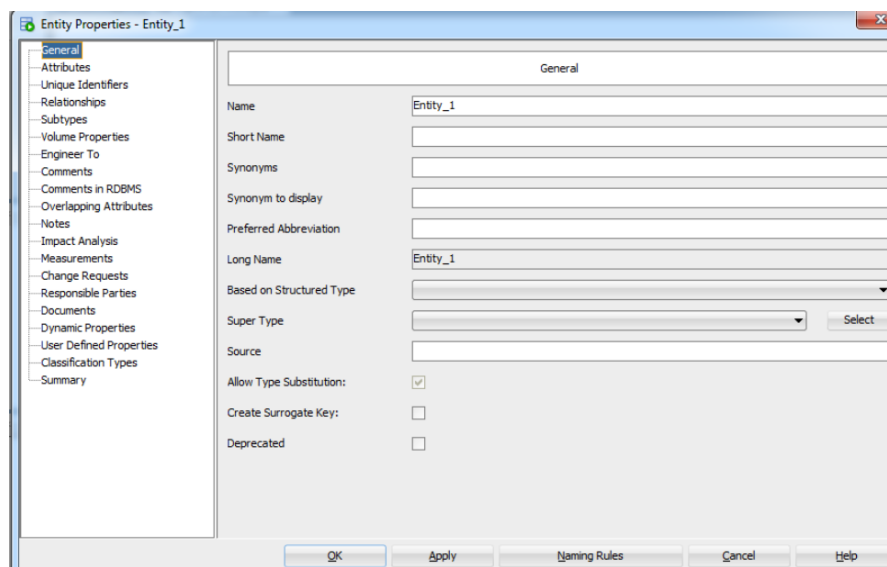


## Crear Entidades

Antes de poder crear relaciones, tenemos que tener tipos de entidades que relacionar. Vamos a aprender cómo crear una entidad. Para ello pinchamos en el sobrecito con una rueda dentada que pone **Entidades**. A continuación pinchamos en cualquier lugar de la ventana



Y se nos abrirá la subventana siguiente.



### CONSEJOS Y BUENAS PRÁCTICAS

Las entidades se nombran en singular. Los nombres deben ser cortos y claros. No deben contener caracteres extraños o exclusivos del alfabeto español (ñ, tildes...).

Se debe establecer un criterio del uso de mayúsculas y minúsculas, guiones y demás, al nombrar los distintos elementos (entidades, atributos, relaciones, etc.), y mantener ese criterio en toda la base de datos. Por ejemplo, un criterio que se puede utilizar es el de Java, en donde se comienzan los nombres en minúsculas y si hay más de una palabra, se escriben todas juntas escribiendo la primera letra de las siguientes palabras en mayúsculas. Ejemplo: UnoDosTres, telefonoFijo.

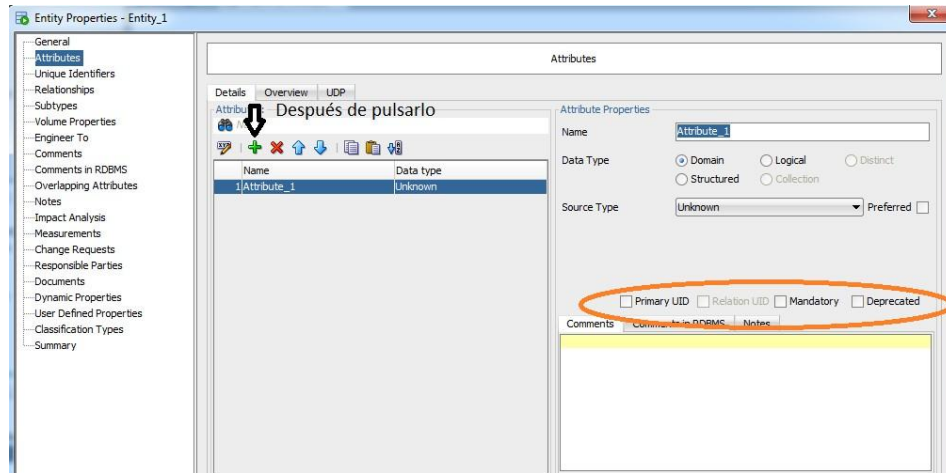
A la hora de realizar varias prácticas es interesante añadir un identificador de dicha práctica. Ejemplo: P1\_Alumno. Sería la entidad alumno de la práctica 1.

Es aconsejable utilizar nombres generales para las entidades, para poder utilizar la misma entidad para el mayor número de instancias posibles. Por ejemplo en lugar 3 entidades: empleado + socio + directivo, se puede utilizar una sola: persona.



Para nuestro proyecto vamos a seguir el criterio de Java salvo para las entidades que siempre irán en mayúsculas.

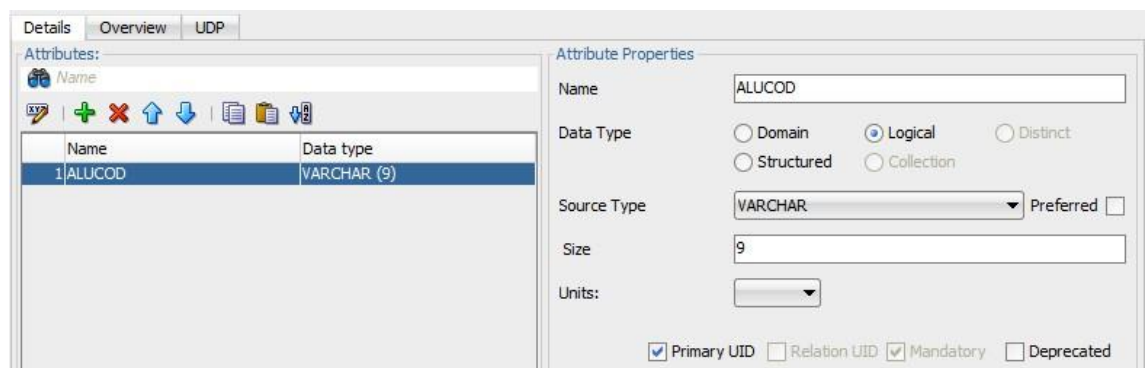
## Creación atributos

Cada entidad va a tener una serie de atributos. Para crearlos nos vamos a la pestaña attributes de la entidad. Podemos hacerlo cuando estamos creando la entidad o una vez creada haciendo doble click sobre ella.



Algunas de las opciones más son:

- **Name:** indicamos que nombre tendrá nuestro atributo.
- **Data Type:** Es el tipo de dato. Lo normal será elegir logical y dentro del logical el varchar, date, numeric, que son los más frecuentes.
- **Primary UID:** Si marcamos esta opción estamos diciendo que dicho atributo será parte de la Primary Key de nuestra identidad.
- **Mandatory:** esta opción indica que es un campo obligatorio. Si lo dejamos vacío será opcional. Debemos entender que aunque algo sea necesario, no siempre interesará hacerlo obligatorio. Cuando marcamos dicha opción estamos obligando que al introducir datos dentro de la entidad, todos esos campos no pueden quedarse vacíos. A veces interesa poder dejar parte de la información sin rellenar y posteriormente tener un programita o aplicación que nos diga que algunos campos están vacíos por acelerar el proceso.
- Las flechas   nos permite reordenar los atributos.



En la imagen anterior tenemos un atributo que representa el código de un alumno y es un dato de tipo varchar de tamaño 9 (es decir, podrá tener 9 caracteres alfanuméricos).

A continuación vemos un atributo que representa el número de asignaturas que tiene el alumno. Sería una variable numérica. Donde la precisión indica el número de cifras y la scale nos dice cuántas de esas cifras corresponden a los decimales. Por

ejemplo: un 5,2 significa 5 cifras de las cuales dos son decimales. 134,32 serviría pero 1000,1 no valdría. Ni 10,134.

The screenshot shows a software interface for defining database attributes. On the left, a table lists attributes: 1 ALUCOD (VARCHAR 9), 2 NOMBRE (VARCHAR 30), 3 APELLIDO (VARCHAR 50), 4 F\_NACIMIENTO (Date), 5 NUM\_ASIGNATURA (DECIMAL 2), and 6 CURSO (VARCHAR 30). The attribute 'NUM\_ASIGNATURA' is selected. On the right, the 'Attribute Properties' panel shows its configuration: Name is 'NUM\_ASIGNATURA', Data Type is 'Logical' (selected over Domain, Distinct, Structured, and Collection), Source Type is 'DECIMAL' (with a 'Preferred' checkbox), Precision is '2', and Scale is '0'. At the bottom, there are checkboxes for 'Primary UID', 'Relation UID', 'Mandatory', and 'Deprecated', all of which are currently unchecked.

## CONSEJOS Y BUENAS PRÁCTICAS

Los nombres deben ser cortos y claros. No deben contener caracteres extraños o exclusivos del alfabeto español (ñ, tildes...).

Todos los atributos de una entidad deben tener una relación directa con la entidad, deben referirse a ella exclusivamente. Por ejemplo, en una entidad cuentaBancaria no debe haber atributos email, telefono... referidos al cliente, en su lugar estos atributos deberían estar en otra entidad cliente.

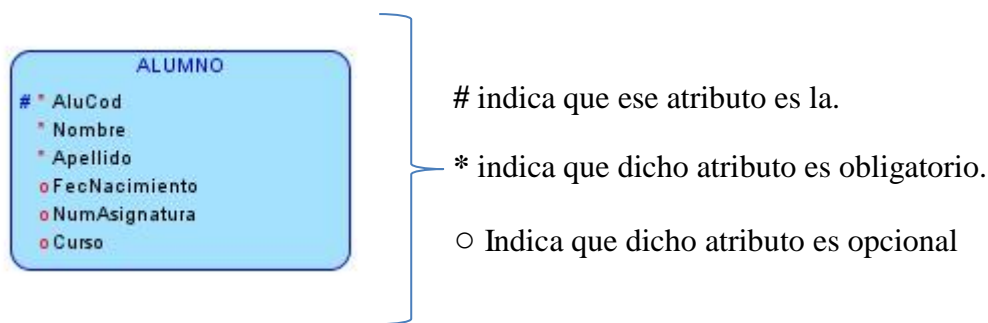
El atributo que será o los atributos que serán nuestra PK es interesante que no sean muy genéricos. Si por ejemplo es un código y la entidad es alumno. Sería interesante poner AluCod o CodAlum en vez de código

No se debe tener redundancia de atributos, con varios atributos similares dentro de una misma entidad. Cuando sucede, es una señal de que estamos diseñando la base de datos mal. Por ejemplo, no se debe tener en una entidad empleado los atributos cónyuge, hijo1, hijo2, hijo3... En su lugar debería haber otra entidad familiar relacionada con empleado.

La PK debe ser siempre el primer atributo de la entidad.

Cuando la PK está formada por más de un atributo, aparece primero el atributo más general y luego los más concretos, por orden. Por ejemplo, primero edificio, después planta y por último puerta

Tras crear los diferentes atributos le damos a ok y se nos crea la entidad.



## Creación de modelos

Vamos a ir viendo ahora como se crea cada una de las relaciones que hemos visto teóricamente. En general crearemos primero un modelo lógico, luego construiremos a partir de este un modelo relacional. Este modelo teórico serán tablas de datos en la realidad. Para crear dichas tablas y trabajar con ellas crearemos un archivo dll que contendrá las instrucciones que debemos darle a SQLdeveloper para que las construya. Luego veremos que instrucciones da internamente SQLdeveloper para introducir datos en las tablas. Por último, cuando tengamos datos introducidos aprenderemos a exportarlos por si queremos llevarlos a algún otro lugar. Veremos además, que fallos nos indicará el programa en relación con el tipo de relación que estamos trabajando. Para ello, observaremos que relaciones existen, y veremos las restricciones que se crean.

### Creación de la relación 1:N

#### *Modelo lógico*

Lo primero que hacemos es crear la entidad vuelo y la entidad pasajero.

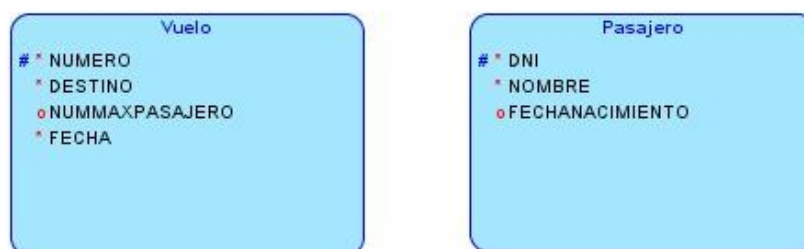


Imagen 18

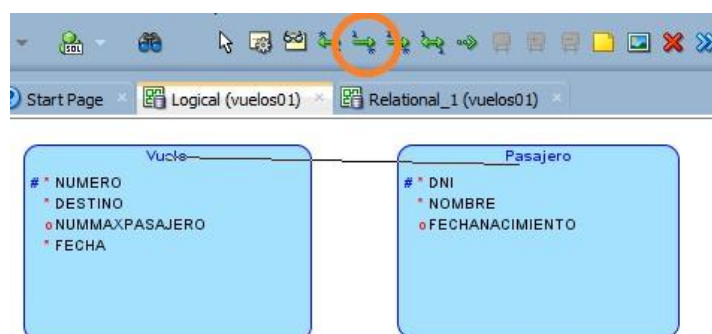


Imagen 19

Hemos pinchado en el círculo naranja de la Imagen 19. Luego pinchamos en una de las entidades y nos movemos hacia la otra. Observamos en la Imagen 19 que se

crea una línea negra. Al soltarla se nos abre una nueva ventana, Imagen 20, que interesa estudiar a fondo.

Imagen 20

1. Indica el **nombre** que tendrá la relación. Al principio solemos dejar el nombre por defecto pero en un proyecto grande, es interesante darle un nombre a cada relación de forma que sepamos cuál es su función.
2. Indica la **entidad origen**. Vuelo.
3. Indica la **entidad destino**. Pasajero.
4. Marca si el **origen** puede contener **uno o varios** elementos de la entidad **destino**. El símbolo que observamos significa uno o varios. Un vuelo puede tener uno o varios pasajeros.
5. Marca si el **destino** puede contener **uno o varios** elementos de la entidad **origen**. El símbolo que observamos significa uno. Un pasajero debe ir en un vuelo.
6. Indica si el **origen puede o debe contener elementos de la otra entidad**. Si está marcada significa puede.
7. Indica si el **destino puede o debe contener elementos de la otra entidad**. En este caso al estar desmarcado, sería obligatorio.
8. Esta casilla convertiría la relación en una que veremos más adelante. Será la relación 1:N identificadora.

Cuando termines de marcar las opciones que nos interesen le damos a Ok. Al hacerlo se nos crea la relación que podemos observar en la Imagen 21 donde he remarcado los elementos de interés. Si más adelante vemos que la relación es diferente podemos hacer doble click en la relación para cambiarla.



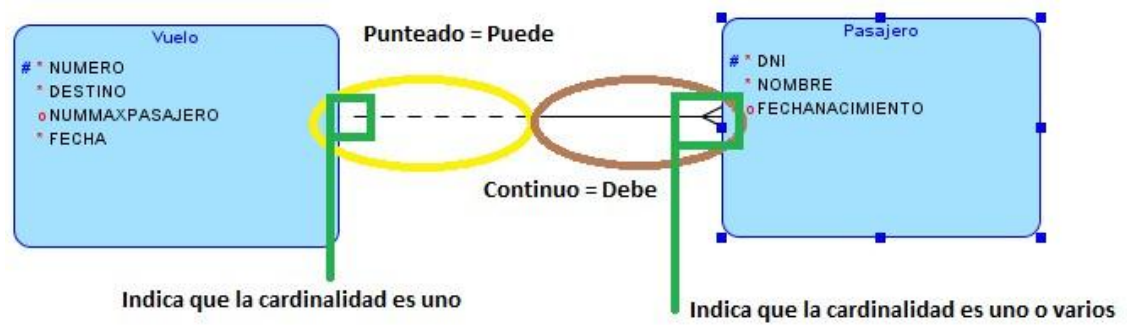


Imagen 21

## CONSEJOS Y BUENAS PRÁCTICAS

Aunque en la realidad sea lógico que ambas relaciones sean un debe. Lo cierto es que siempre usaremos puede para la entidad 1 y debe o puede para la N.

Esto no es un capricho ya que el programa no añadirá nada si ponemos debe en vez de puede en la entidad 1. Para poder controlar que un vuelo no puede ir vacío, es decir, que debe tener pasajeros necesitaremos un programa que haga dicha comprobación.

El programa no añade nada por un buen motivo. Cuando creamos las tablas y vayamos a meter datos nos encontraríamos en un bucle infinito. Si quiero añadir vuelos necesito tener pasajeros para dicho vuelo así que tendría que crear pasajeros que meter al vuelo. Pero cuando voy a introducir un pasajero tendré que asignarle un vuelo que no he podido crear.

## Modelo Relacional

Para crear el modelo relacional pinchamos en la doble flecha azul que vemos en la Imagen 22 y luego Engineer.

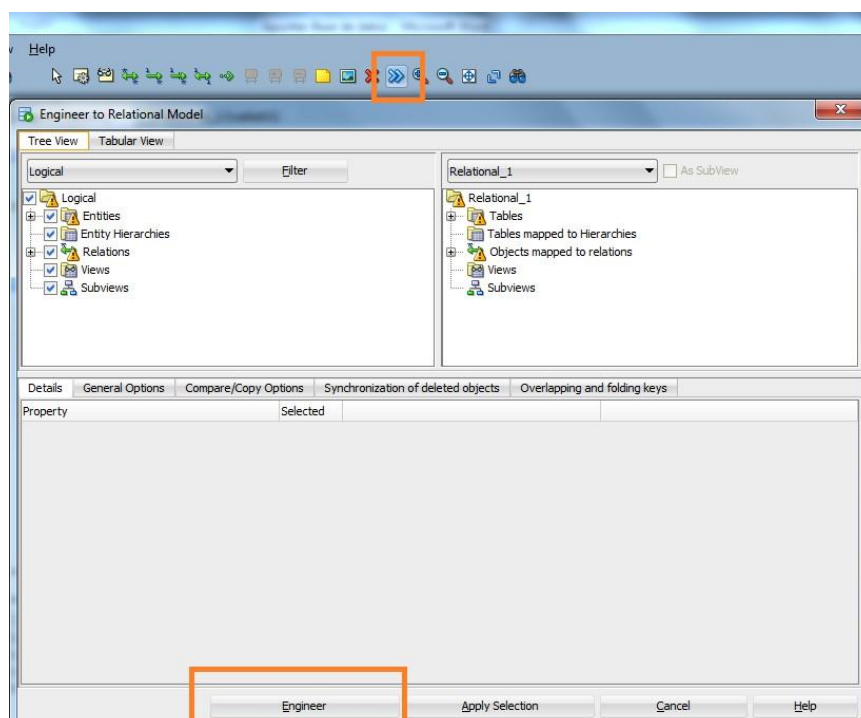


Imagen 22



Una vez creado se abrirá una nueva hoja con el modelo relacional. Lo normal es que el diagrama relacional no se vea completo.

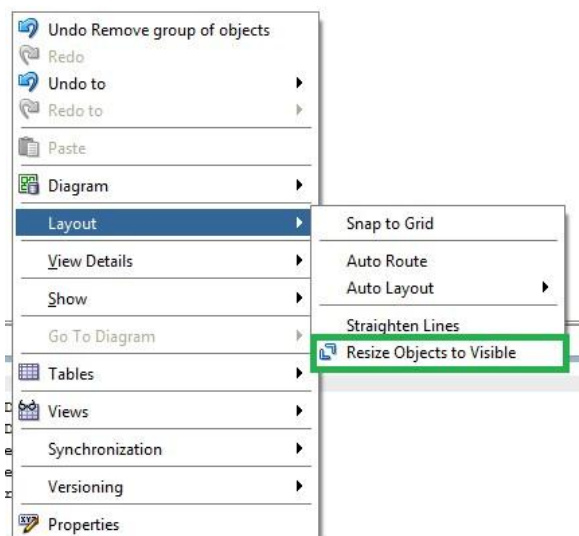


Imagen 23 Redimensionar el modelo relacional

Si pinchamos en la pantalla con el botón derecho y nos vamos a Layout, ResizeObjects to visible. Nos ajusta el diagrama para que se lea todo el texto. Ojo, si tenemos muchas entidades, puede ocurrir que unas entidades se pisen con otras y debamos desplazarlas. Basta con pinchar encima de una y mover el ratón.

Echemos un vistazo al modelo relacional.

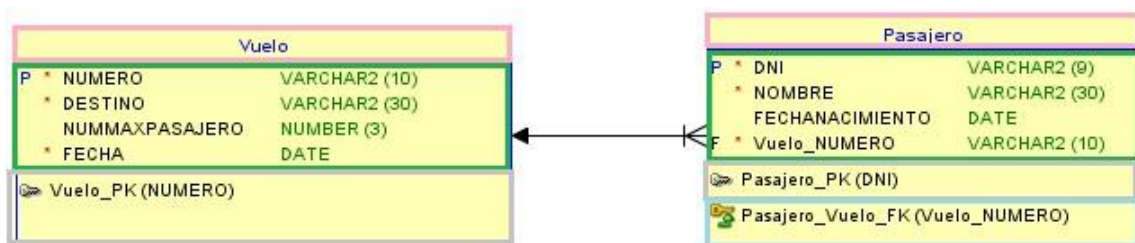


Imagen 24

En la Imagen 24 hemos coloreado cada recuadro de un color para indicar que estamos observando en cada uno.

- **Recuadro rosa:** contiene el nombre de la entidad.
- **Recuadro verde:** contiene los atributos que tiene cada entidad. Debemos observar que en la entidad Pasajero se ha creado un nuevo atributo debido a la relación. Dicho atributo lleva una F para indicar que es una foreign key. Por tanto, hemos corroborado que cuando creamos una relación 1:N, la PK de 1 viaja a la entidad N.
- **Recuadro gris:** indica el nombre de la restricción o constrain que crea una primary key. Cuando estemos introduciendo datos, si pasamos por alto rellenar el número o el dni, nos dará un error. En el primer caso aparecerá vuelo\_pk violated y en el segundo saldrá pasajero\_pk violated.
- **Recuadro azul:** indica el nombre de la restricción o constrain que crea la relación 1:N. Que indica que el campo FK es un campo obligatorio (podría no serlo, depende las opciones que hayamos puesto) y que además lo que coloquemos en el recuadro debe existir como PK en la entidad de origen. Es decir, si en FK ponemos 001. Es porque existe el vuelo número 001.

Hemos dicho que es interesante que la FK tenga el mismo nombre que la PK, pero hemos visto que al crear la relación, esto no sucede de forma natural. Vamos a ver cómo podemos cambiar este valor, además del nombre de las restricciones.

Si hacemos doble click en un recuadro se nos abre la Imagen 25. Vamos a indicar que es lo más interesante que podemos hacer.

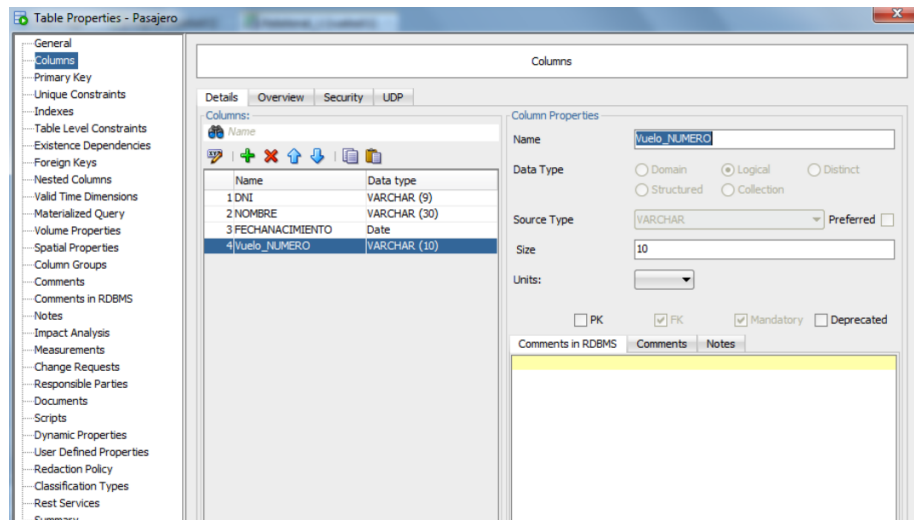


Imagen 25

- **General:** nos permitiría cambiar el nombre de la entidad. Esta opción no parece muy interesante pero veremos que en la relación N:M si es muy útil.
- **Columns:** nos permite ver los atributos. Aquí es donde cambiaremos el nombre a nuestra Foreign Key para que coincida con el nombre de nuestra Primary Key.
- **Primary Key:** Aquí podemos cambiar el nombre de la restricción de la primary key. Cuando olvidemos escribir algo en la columna de la primary key nos saltará un nombre relacionado con el valor que pongamos.
- **Foreign Keys:** Aquí podemos cambiar el nombre de la restricción de la foreign key. Es interesante poner un valor que no sea excesivamente grande y que esté relacionado con la relación que la ha creado. El límite de caracteres es de 30. A veces, el programa automáticamente la crea con una extensión mayor, así que debemos estar atentos si nos da un error al crearse el DDL, ya que el error podría venir de aquí.

El resto de opciones o no tienen interés o se escapan de nuestros dominios para este curso.