# PRÁCTICA 2 - Diseño de Datos en un Sistema de Información

## 1 - Modificaciones en la práctica 1

Se ha añadido la fecha y la hora actual a los requisitos de entrada de RF2.3,RF2.4,RF3.1,RF3.3, RF3.4

## 2 - Esquemas externos DFD1

Para cada subsistema se han elaborado los correspondientes esquemas externos DFD1. Esto es, los diagramas E/R que modelan los datos que los procesos (en este caso los procesos son los distintos requisitos funcionales que pusimos de nuestro subsistema) leen o escriben en un almacén, teniendo en cuenta las restricciones semánticas asociadas. Además, se ha hecho el esquema externo del almacén relativo a cada subsistema (diagrama E/R que modela los datos que almacena un almacén, teniendo en cuenta las restricciones semánticas asociadas).

En resumen, para cada uno de los requisitos funcionales, se ha elaborado un esquema externo en el cual se proporciona una visión conceptual de la parte de la Base de Datos con la que trabaja el proceso (cada requisito funcional lo vemos como un proceso).

## 2.1 - Explicación de algunos esquemas externos DFD1

### Gestión de usuarios

> **Consultar historial de vuelos del usuario**:

A partir del número de pasaporte del usuario, que es lo único que nos interesa en este proceso del usuario, accedemos a las reservas asociadas a dicho número de pasaporte, identificadas todas por un código de reserva. Entre usuario y reserva existe una relación que hemos denominado "Reservas usuario", que tiene una cardinalidad 1 a muchos y una navegabilidad hacia usuarios, es decir, que un usuario puede tener asociadas muchas reservas pero una reserva solo puede corresponder a un único usuario. No solo puede, si no debe corresponder a un usuario; no puede haber una reserva que no sea de ningún usuario, de ahí la obligatoriedad;

la doble línea que sale de reserva hacia usuario. Para cada una de las reservas asociadas al usuario, se accede al número y datos asociados del vuelo (fecha y hora de llegada y salida). Existe una relación muchos a 1 con navegabilidad hacia Vuelos y obliatoriedad en Reservas (toda reserva debe tener un único vuelo asociado y un vuelo puede estar asociado a varias reservas). A partir del vuelo, se consultan finalmane los códigos de los códigos de aeropuertos de origen y destino (existe una relación muchos a 1 entre vuelos y aeropuertos, con navegabilidad hacia aeropuertos y obligatoriedad en vuelos (muchos vuelos pueden y deben estar asociados a un único aeropuerto de origen y de destino, pero un aeropuerto puede ser origen o destino de varios vuelos)).

### Gestión de pagos

> **Realizar pago**:

A partir del número de la reserva que se desea pagar, accedemos al número de asientos reservados y al estado de la reserva (recordemos que para poder pagar una reserva, era necesario que su estado fuera "Pendiente"). A partir del número de reserva podemos acceder al usuario que a hecho la reserva (existe una relación entre Reserva y Usuario, muchos a 1 con navegabilidad hacia Usuario y obligatoriedad en Reserva (toda reserva debe pertenecer a un usuario y un usuario puede haber realizado una o varias reservas)), con objeto de cosultar el saldo disponible para poder pagar la reserva. A su vez a partir del número de reserva se accede al vuelo reservado y se consulta el precio del asiento (existe una relación entre Reserva y Vuelo de muchos a 1 con navegabilidad hacia Vuelo y obligatoriedad en Reserva (una reserva debe estar asociada a un único vuelo y un vuelo puede haberse reservado varias veces)). Finalmente se crea un nuevo pago; a partir de los datos obtenidos, se crea el pago, que será de una cantidad igual al precio del asiento por el número de asientos reservados (existe una relación 1 a 1 entre reserva y pago con obligatoriedad en pago (un pago debe estar asociado a una única reserva y una reserva puede ser pagada una única vez (la relación de obligatoriedad no está en reserva porque una reserva puede estar Pendiente o Pagada))).

# 3 - Esquemas externos DFD0

Para cada subsistema, viendo cada uno de los mismos como un proceso, se ha elaborado un esquema externo (una visión conceptual de la parte de la base de datos con la que el proceso trabaja (lee y/o escribe)). Acompañado además del correspondiente esquema externo del almacén relativo a cada subsistema.

El proceso de elaboración de los esquemas externos de los procesos y almacenes de DFD0 se ha llevado a cabo mediante la integración de los procesos y almacenes correspondientes a su refinamiento en DFD1.

Básicamente se ha codigo el esquema externo DFD1 más grande y a él se le han ido añadiendo relaciones y atributos que no tenía y que estaban en otros esquemas externos DFD1; se han ido integrando los esquemas de los procesos en DFD1.

# 4 - Diagrama entidad-relación

Para obtener el diagrama entidad-relación de nuestro sistema, simplemente se han unido cuidadosamente los esquemas externos de DFD0. Se han añadido a las diferentes instancias (una por cada subsistema + aeropuertos) sus correspondientes atributos (añadiendo los que aparecían en los distintos esquemas), así como las distintas relaciones entre las mismas, conservando la cardinalidad, navegabilidad y obligatoriedad.

# 5 - Tablas resultantes y fusión de tablas

Para cada una de las entidades y para cada una de las relaciones que conforman el modelo E/R elaborado, se ha de crear una tabla. El paso a tablas se ha hecho de la siguiente forma:

En primer lugar, para cada una de las entidades, se crea una tabla en la que se incluyen sus atributos y se identifican claves primaria y claves candidatas.
Para cada una de las relaciones, se crea la tabla correspondiente cuyas columnas serán las claves primarias de las entidades que se relacionan y cuya clave primaria dependerá del tipo de relación del que se trate: si es muchos a muchos, la clave primaria será una combinación de las claves primarias de las entidades que se relacionan, si es 1 a muchos, o muchos a uno, la clave primaria será la de la entidad que está en el lado de muchos, y si es 1 a 1, se elige una de las dos, y la otra se pone como candidata.

Finalmente, una vez creadas todas las tablas, se establecen las claves externas de las tablas, que son esenciales para mantener la integridad referencial en una base de datos relacional; es una columna, o conjunto de columnas que hace referencia a la clave primaria de otra tabla. En las tablas de las relaciones, cada una de las columnas de las tablas es una clave externa que hace referencia a la clave primaria de la entidad correspondiente.

La fusión de tablas se refiere a la combinación de datos de dos o más tablas en una sola. Por lo general, significa identificar un campo que exista en dos tablas diferentes. En este caso se pueden hacer 3 fusiones:

- Fusión Reserva-ReservaUsuario y Reserva-ReservaVuelo: Al mostrar el listado de reservas, y para consultar cada reserva, se muestra el número de pasaporte del usuario que ha realizado la reserva y el número de vuelo asociado a la reserva. Por tanto, dichas operaciones se aceleran si se realiza dicha fusión.
- Fusión Pago-ReservaPago: Cada vez que se muestran los pagos de un usuario, se muestra el número de reserva a los que están asociados dichos pagos. Además, se evita la operación JOIN de la tabla Pago y ReservaPago al mostrar la información de un pago, dado un número de reserva. Nótese que la fusión de la tabla Reserva con ReservaPago no tendría sentido, ya que tendría muchos valores nulos.
- Fusión Vuelo-Origen y Vuelo-Destino: como en los casos anteriores, la fusión de estas tablas evita dos operaciones JOIN por cada operación de mostrar la información de un vuelo. Por ello, las operaciones de mostrar la información de los vuelos en los que ha estado el usuario, o los vuelos con un origen y destino determinados se aceleran mucho.

# 6 - Normalización de tablas

La normalización de tablas es un proceso en el diseño de bases de datos relacionales que busca organizar la información de manera eficiente, minimizando la redundancia y evitando problemas de actualización e inserción de datos. El objetivo principal de la normalización es mejorar la integridad de los datos y reducir la posibilidad de anomalías en la base de datos. Se logra dividiendo las tablas grandes en tablas más pequeñas y relacionadas.

Hay varias formas normales que se utilizan en el proceso de normalización, siendo las más comunes la Primera Forma Normal (1NF), la Segunda Forma Normal (2NF) y la Tercera Forma Normal (3NF). Aquí hay una breve explicación de cada una:

- Primera Forma Normal (1NF):
  > Un conjunto de datos está en 1NF si no hay conjuntos anidados ni valores repetidos en las filas. Cada columna debe contener un solo valor, y cada

> fila debe ser única. La 1NF elimina la redundancia a nivel de columna.

- Segunda Forma Normal (2NF):
  > Un conjunto de datos está en 2NF si ya está en 1NF y si todos los atributos no clave dependen completamente de la clave primaria. Esto implica la eliminación de dependencias parciales, dividiendo la tabla en dos o más tablas relacionadas.

- Tercera Forma Normal (3NF):
  > Un conjunto de datos está en 3NF si ya está en 2NF y si no existen dependencias transitivas. Las dependencias transitivas se refieren a situaciones en las que un atributo no clave depende de otro atributo no clave a través de un atributo clave.

El proceso de normalización implica identificar las dependencias funcionales

> Dependencia funcional: Dada una relación R con n>0 atributos y dos subconjuntos de atributos de R llamados $\alpha$ y $\beta$, se dice que a determina funcionalmente a $\beta$ (o equivalentemente que $\alpha$ depende funcionalmente de $\alpha$) si para cualquier instancia válida r de R y cualquier pareja de tuplas s y t de r que tengan los mismos valores para los atributos del subconjunto $\alpha$, se verifica que tienen los mismos valores para los atributos del subconjunto $\beta$.

Sobre las tablas ya fusionadas, identificamos las dependencias funcionales que hay. En nuestro caso, las únicas dependencias funcionales que hay en cada tabla es la del conjunto de atributos de la tabla de la clave primaria: la clave primaria de toda tabla determina funcionalmente al resto de atributos. Además esta dependencia es completa (una ependencia funcional se dice completa si $\alpha$ determina funcionalmente a $\beta$ y no hay ningún subconjunto de atributos de $\alpha$ que determine funcionalmente a $\beta$). Aunque en Pago, sí hay dos dependencias funcionales, que son del mismo tipo que antes (Número de pago determina funcionalmente a todos los atributos de Pago y, dado que Número de reserva es CC, también hay una dependencia funcional del conjunto de atributos de Reserva, con respecto a Numero de reserva).

Analicemos a continuación la normalización de las tablas:

- Las tablas están en primera forma normal, ya que todos los dominios son atómicos (todos los valores en la tabla son indibisibles; la tabla se encuentra libre de duplicados y de valores compuestos o multivaluados).

- Las tablas están en segunda forma normal, ya que están en 1ª y todas las claves candidatas de las tablas están formadas por un único atributo, por tanto todos los atributos no primos dependen de forma completa de ellas.
- En todas las dependencias funcionales que tenemos, el atributo que se encuentra en la izquierda es una clave candidata, y en particular, una superclave, por lo que todas las tablas se encuentran en tercera forma normal (un conjunto de datos está en 3NF si ya está en 2NF y si no existen dependencias transitivas. Las dependencias transitivas se refieren a situaciones en las que un atributo no clave depende de otro atributo no clave a través de un atributo clave.).
- Como todas las claves candidatas son simples, las tablas están en forma normal de Boyce-Codd (es una variante de la 3FN, que tiene deficiencias cuando hay varias claves candidatas, las claves candidatas son compuestas o las claves candidates se solapan).

# 7 - Sentencias SQL de creación de tablas e inserción de tuplas

Hemos aprovechado la base de datos Oracle ya creada en el Seminario 1, para trabajar en esta práctica. Hemos borrado las tablas que en su momento creamos para la elaboración del Seminario 1 y procedimos a crear e insertar tablas y tuplas nuevas relativas a nuestro SI.

Para cada una de las tablas derivadas de la fusión, se ha creado una tabla, mediante una sentencia SQL (sentencia CREATE TABLE NOMBRETABLA(atributos)), escrita y ejecutada en SQL Developer. En dicha sentencia se especifica: el nombre de la tabla y los atributos especificando: nombreAtributo,Tipo, si referencia a algún atributo de otra tabla, especificacion (si es clave primaria, si hay que hacer alguna comprobación previa sobre el atributo de cara a futuras inserciones/modificaciones, si tiene algún valor por defecto, etc).

> En la tabla de usuario y aeropuerto tenemos el correo (solo en usuario) y el número de teléfono (del usuario y del aeropuerto), que tienen un formato peculiar y que se ha de comprobar que se cumple previamente a la hora de insertar una tupla. Dicha comprobación se especifica en la sentencia de creación de la tabla para que se tenga en cuenta en futuras inserciones. Se usar la orden CHECK a la que se le pasa como argumento lo que quiere que se compruebe. En nuestro caso, como queremos que se compruebe el formato

> característico del correo y del número de teléfono, hemos tenido que especificarle una expresión regular que permite validar el formato de dichos atributos. Por ejemplo, la expresión regular que representa el formato que debe tener el correo es:

- ^: Indica que la coincidencia debe comenzar desde el principio de la cadena.
- \w+: Representa uno o más caracteres de palabra (letras, dígitos o guiones bajos).
    - (.\w+)*: Un grupo que indica que puede haber un punto seguido de uno o más
      caracteres de palabra, y este grupo puede repetirse cero o más veces (permitiendo múltiples subdominios).
- @: Representa el símbolo '@' en el correo electrónico.
- \w+: Otro conjunto de uno o más caracteres de palabra (dominio).
- .: El punto que separa el dominio y el dominio de nivel superior (TLD).
- \w+: El dominio de nivel superior (TLD).
- $: Indica que la coincidencia debe llegar al final de la cadena.

```
Básicamente lo que hace se hará cuando se quiera insertar una tupla es
comprobar que lo que se le pase como "correo" o "teléfono" y ver que
verifica la expresión regular.
```

Finalmente, se han insertado en cada una de las tablas creadas, dos tuplas, mediante la sentencia SQL: INSERT INTO NOMBRETABLA VALUES (atributos con sus formatos correspondientes).

> En Reserva y Pago, se ha hecho una cosa curiosa que consiste en la declaración y utilización de un objeto SEQUENCE, que permite generar automáticamente valores únicos de manera incremental, generalmente utilizados para asignar identificadores de forma única a las filas de una tabla. De esta forma, por ejemplo en Reserva, se ha hecho la inserción de las tuplas de esta manera:

```
CREATE SEQUENCE reservas_seq
START WITH 1
INCREMENT BY 1
NOCACHE ;

 Inserción

INSERT INTO RESERVA (NUMRESERVA,NUMUSUARIO,NUMVUELO,
NUMASIENTOSRESERVADOS)
VALUES (reservas_seq.nextval, '123456789R', 'AIR9345', 2);

INSERT INTO RESERVA (NUMRESERVA,NUMUSUARIO,NUMVUELO,
NUMASIENTOSRESERVADOS)
VALUES (reservas_seq.nextval, '674563880Z', 'FLY3456', 6);
```

> La secuencia "reservas_seq", genera automáticamente valores únicos para la columna "NUMRESERVA" al realizar inerciones en al tabla "RESERVA. Sin la secuencia, tendríamos que gestionar manualmente la generación de números de reserva, lo cual podría ser propenso a errores y conflictos en entornos multiusuario. La secuencia asegura la unicidad y la consistencia de los números de reserva al proporcionar automáticamente valores únicos cada vez que se realiza una inserción en la tabla RESERVA.