Iteración 3

Juan Sebastián Prieto, Santiago Rojas Herrera
Sistemas Transaccionales
Universidad de los Andes, Bogotá, Colombia
{js.prieto10, s.rojas19}@uniandes.edu.co
Fecha de presentación: Octubre 23 de 2016

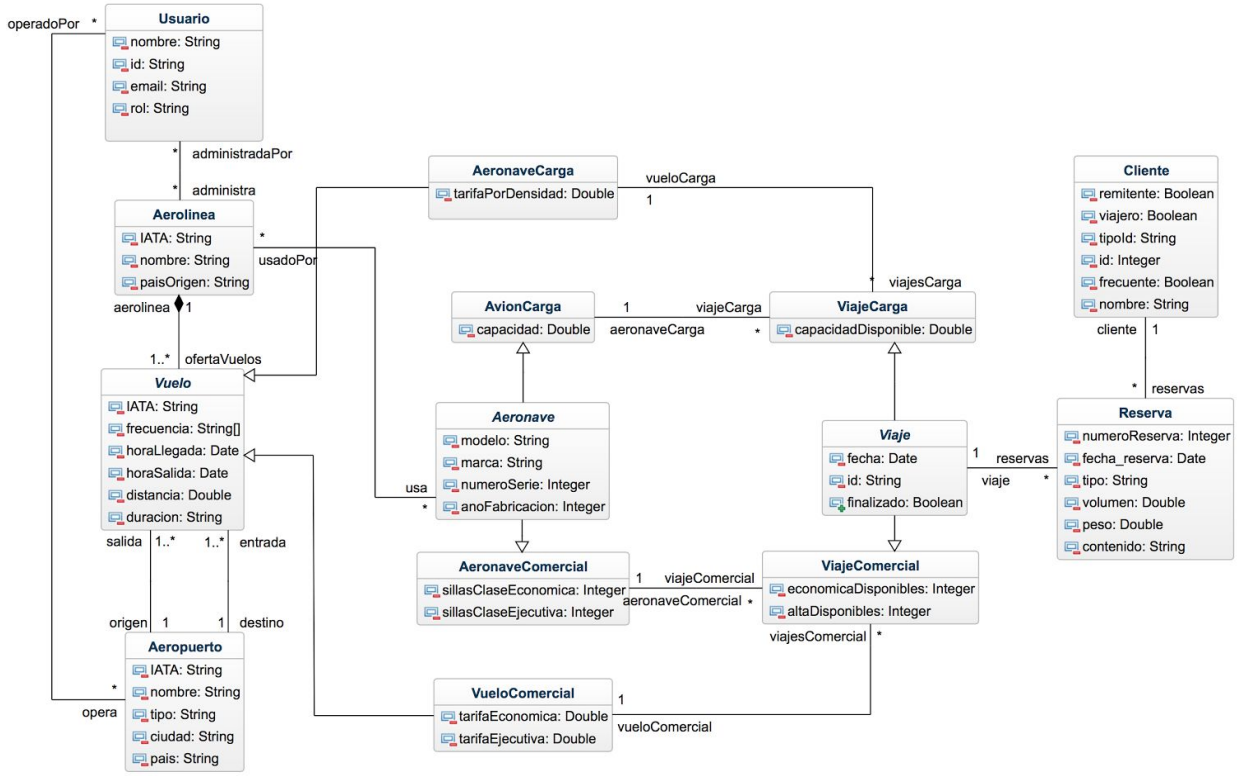
Análisis **1**

Diseño de la aplicación **2**

Construcción de la aplicación. **8**

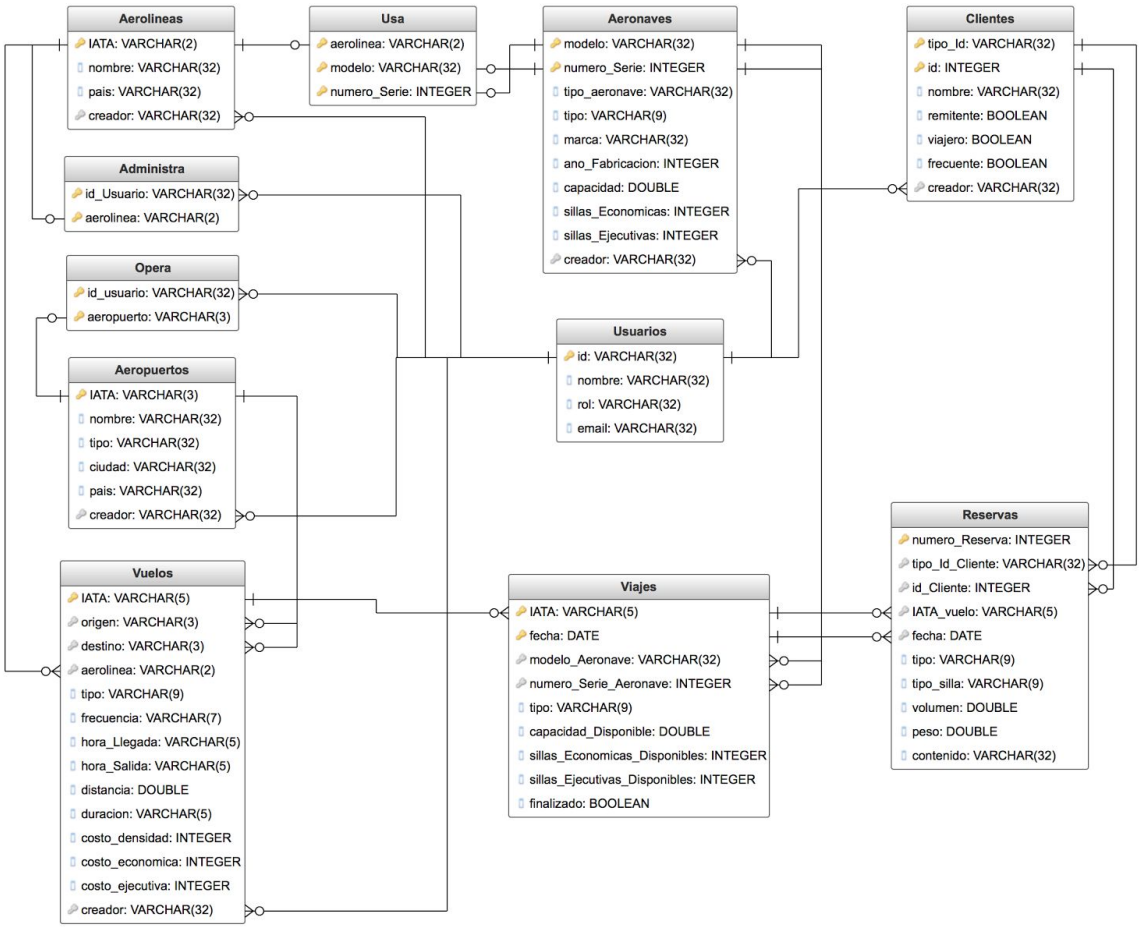
1. Análisis

Fue necesario agregar un atributo de tipo date a la reserva, para contar con la fecha en la que se realizó la reserva misma, con fin de poder realizar los requerimientos funcionales 13 y 14.



2. Diseño de la aplicación

a. Como se hizo en la iteración pasada, se basó en el modelo conceptual para crear el modelo relacional. A continuación se anexa el modelo relacional del sistema:



No existen cambios relevantes en el diseño del modelo relacional de esta iteración a comparación de la pasada.

Para la creación de las tablas son necesarias las siguientes sentencias:

```
/*Creacion tablas*/

CREATE TABLE USUARIOS
(
  ID VARCHAR2(32) NOT NULL,
  NOMBRE VARCHAR2(32) NOT NULL,
  ROL VARCHAR2(32) NOT NULL CHECK(ROL IN ('ADMINISTRADOR', 'AEROPUERTO',
'AEROLINEA')),
  EMAIL VARCHAR2(32) NOT NULL,
```

```
  CONSTRAINT USUARIOS_PK PRIMARY KEY (ID)
);


CREATE TABLE AEROPUERTOS
(
  IATA VARCHAR2(3) NOT NULL,
  NOMBRE VARCHAR2(32) NOT NULL,
  TIPO VARCHAR2(13) NOT NULL CHECK(TIPO IN('INTERNACIONAL', 'NACIONAL',
'LOCAL')),
  CIUDAD VARCHAR2(32) NOT NULL,
  PAIS VARCHAR2(32) NOT NULL,
  CREADOR VARCHAR2(32) NOT NULL,
  CONSTRAINT AEROPUERTOS_PK PRIMARY KEY (IATA),
  CONSTRAINT AEROPUERTOS_CREADOR_FK FOREIGN KEY (CREADOR) REFERENCES
USUARIOS(ID)
);



CREATE TABLE AEROLINEAS
(
  IATA VARCHAR2(2) NOT NULL,
  NOMBRE VARCHAR2(32) NOT NULL,
  PAIS VARCHAR2(32) NOT NULL,
  CREADOR VARCHAR2(32) NOT NULL,
  CONSTRAINT AEROLINEAS_PK PRIMARY KEY (IATA),
  CONSTRAINT AEROLINEAS_CREADOR_FK FOREIGN KEY (CREADOR) REFERENCES
USUARIOS(ID)
);


CREATE TABLE CLIENTES
(
  TIPO_ID VARCHAR2(32) NOT NULL,
  ID NUMBER NOT NULL,
  NOMBRE VARCHAR2(32) NOT NULL,
  REMITENTE CHAR NOT NULL CHECK(REMITENTE IN (0,1)),
  VIAJERO CHAR NOT NULL CHECK(VIAJERO IN (0,1)),
  FRECUENTE CHAR NOT NULL CHECK (FRECUENTE IN (0,1)),
  CREADOR VARCHAR2(32) NOT NULL,
  CONSTRAINT CLIENTES_PK PRIMARY KEY (TIPO_ID, ID),
  CONSTRAINT CLIENTES_CK CHECK((VIAJERO=0 AND FRECUENTE=0) OR VIAJERO=1),
  CONSTRAINT CLIENTES_CREADOR_FK FOREIGN KEY (CREADOR) REFERENCES USUARIOS(ID
);


CREATE TABLE AERONAVES
(
  MODELO VARCHAR2(32) NOT NULL,
  NUMERO_SERIE NUMBER NOT NULL,
  TIPO_AERONAVE VARCHAR2(32) NOT NULL,
  TIPO VARCHAR2(9) NOT NULL CHECK(TIPO IN ('CARGA','COMERCIAL')),
```

```
  MARCA VARCHAR2(32) NOT NULL,
  ANO_FABRICACION NUMBER(4,0) NOT NULL,
  CAPACIDAD NUMBER,
  SILLAS_ECONOMICAS NUMBER,
  SILLAS_EJECUTIVAS NUMBER,
  CREADOR VARCHAR2(32) NOT NULL,
  CONSTRAINT AVIONES_PK PRIMARY KEY (MODELO, NUMERO_SERIE),
  CONSTRAINT AVIONES_CK CHECK ((TIPO='CARGA' AND CAPACIDAD IS NOT NULL AND
SILLAS_ECONOMICAS IS NULL AND SILLAS_EJECUTIVAS IS NULL)
    OR (TIPO='COMERCIAL' AND CAPACIDAD IS NULL AND SILLAS_ECONOMICAS IS NOT
NULL AND SILLAS_EJECUTIVAS IS NOT NULL)),
  CONSTRAINT AERONAVES_CREADOR_FK FOREIGN KEY (CREADOR) REFERENCES
USUARIOS(ID)
);

CREATE TABLE ADMINISTRA
(
  ID_USUARIO VARCHAR2(32) NOT NULL,
  AEROLINEA VARCHAR2(2) NOT NULL,
  CONSTRAINT ADMINISTRA_PK PRIMARY KEY (ID_USUARIO, AEROLINEA),
  CONSTRAINT ADMINISTRA_FK1 FOREIGN KEY (ID_USUARIO) REFERENCES USUARIOS(ID),
  CONSTRAINT ADMININISTRA_FK2 FOREIGN KEY (AEROLINEA) REFERENCES
AEROLINEAS(IATA)
);


CREATE TABLE OPERA
(
  ID_USUARIO VARCHAR2(32) NOT NULL,
  AEROPUERTO VARCHAR2(3) NOT NULL,
  CONSTRAINT OPERA_PK PRIMARY KEY (ID_USUARIO, AEROPUERTO),
  CONSTRAINT OPERA_FK1 FOREIGN KEY (ID_USUARIO) REFERENCES USUARIOS(ID),
  CONSTRAINT OPERA_FK2 FOREIGN KEY (AEROPUERTO) REFERENCES AEROPUERTOS(IATA)
);

CREATE TABLE USA
(
  AEROLINEA VARCHAR2(2) NOT NULL,
  MODELO VARCHAR2(32) NOT NULL,
  NUMERO_SERIE NUMBER NOT NULL,
  CONSTRAINT USA_PK PRIMARY KEY (AEROLINEA, MODELO, NUMERO_SERIE),
  CONSTRAINT USA_AEROLINEA_FK FOREIGN KEY (AEROLINEA) REFERENCES
AEROLINEAS(IATA),
  CONSTRAINT USA_AERONAVE_FK FOREIGN KEY (MODELO, NUMERO_SERIE) REFERENCES
AERONAVES(MODELO, NUMERO_SERIE)
);

CREATE TABLE VUELOS
(
  IATA VARCHAR2(5) NOT NULL,
  ORIGEN VARCHAR2(3) NOT NULL,
  DESTINO VARCHAR2(3) NOT NULL,
  AEROLINEA VARCHAR2(2) NOT NULL,
```

```
  TIPO VARCHAR2(9) NOT NULL CHECK(TIPO IN('CARGA', 'COMERCIAL')),
  FRECUENCIA VARCHAR2(7) NOT NULL,
  HORA_LLEGADA VARCHAR2(5) NOT NULL,
  HORA_SALIDA VARCHAR2(5) NOT NULL,
  DISTANCIA NUMBER NOT NULL,
  DURACION VARCHAR(5) NOT NULL,
  COSTO_DENSIDAD NUMBER,
  COSTO_ECONOMICA NUMBER,
  COSTO_EJECUTIVA NUMBER,
  CREADOR VARCHAR2(32) NOT NULL,
  CONSTRAINT VUELOS_PK PRIMARY KEY (IATA),
  CONSTRAINT VUELOS_AEROLINEA_FK FOREIGN KEY (AEROLINEA) REFERENCES
AEROLINEAS(IATA),
  CONSTRAINT VUELOS_ORIGEN_FK FOREIGN KEY (ORIGEN) REFERENCES
AEROPUERTOS(IATA),
  CONSTRAINT VUELOS_DESTINO_FK FOREIGN KEY (DESTINO) REFERENCES
AEROPUERTOS(IATA),
  CONSTRAINT VUELOS_CK CHECK((TIPO='CARGA' AND COSTO_DENSIDAD IS NOT NULL AND
COSTO_ECONOMICA IS NULL AND COSTO_EJECUTIVA IS NULL)
    OR (TIPO='COMERCIAL' AND COSTO_DENSIDAD IS NULL AND COSTO_ECONOMICA IS NO
NULL AND COSTO_EJECUTIVA IS NOT NULL)),
  CONSTRAINT VUELOS_CREADOR_FK FOREIGN KEY (CREADOR) REFERENCES USUARIOS(ID)
);


CREATE TABLE VIAJES
(
  IATA VARCHAR2(5) NOT NULL,
  FECHA DATE NOT NULL,
  MODELO_AERONAVE VARCHAR2(32),
  NUMERO_SERIE_AERONAVE NUMBER,
  TIPO VARCHAR2(9) NOT NULL CHECK(TIPO IN ('CARGA', 'COMERCIAL')),
  CAPACIDAD_DISPONIBLE NUMBER,
  SILLAS_ECONOMICAS_DISPONIBLES NUMBER,
  SILLAS_EJECUTIVAS_DISPONIBLES NUMBER,
  FINALIZADO CHAR DEFAULT 0 NOT NULL CHECK (FINALIZADO IN (0,1)) ,
  CONSTRAINT VIAJES_PK PRIMARY KEY (IATA, FECHA),
  CONSTRAINT VIAJES_AERONAVE_FK FOREIGN KEY (MODELO_AERONAVE,
NUMERO_SERIE_AERONAVE) REFERENCES AERONAVES(MODELO, NUMERO_SERIE),
  CONSTRAINT VIAJES_VUELO_FK FOREIGN KEY (IATA) REFERENCES VUELOS(IATA),
  CONSTRAINT VIAJES_CK CHECK ((TIPO='CARGA' AND CAPACIDAD_DISPONIBLE IS NOT
NULL AND SILLAS_ECONOMICAS_DISPONIBLES IS NULL AND
SILLAS_EJECUTIVAS_DISPONIBLES IS NULL)
    OR (TIPO='COMERCIAL' AND CAPACIDAD_DISPONIBLE IS NULL AND
SILLAS_ECONOMICAS_DISPONIBLES IS NOT NULL AND SILLAS_EJECUTIVAS_DISPONIBLES I
NOT NULL))
);

CREATE TABLE RESERVAS
(
  NUMERO_RESERVA NUMBER NOT NULL,
  FECHA_RESERVA DATE NOT NULL,
  TIPO_ID_CLIENTE VARCHAR2(32) NOT NULL,
```

```
  ID_CLIENTE NUMBER NOT NULL,
  IATA_VUELO VARCHAR2(5) NOT NULL,
  FECHA DATE NOT NULL,
  TIPO VARCHAR2(9) NOT NULL CHECK (TIPO IN('CARGA','COMERCIAL')),
  TIPO_SILLA VARCHAR(9),
  VOLUMEN NUMBER,
  PESO NUMBER,
  CONTENIDO NUMBER,
  CONSTRAINT RESERVAS_PK PRIMARY KEY (NUMERO_RESERVA),
  CONSTRAINT RESERVAS_CLIENTE_FK FOREIGN KEY (TIPO_ID_CLIENTE, ID_CLIENTE)
REFERENCES CLIENTES(TIPO_ID, ID),
  CONSTRAINT RESERVAS_VIAJE_FK FOREIGN KEY (IATA_VUELO, FECHA) REFERENCES
VIAJES(IATA, FECHA),
  CONSTRAINT RESERVAS_CK1 CHECK ((TIPO='CARGA' AND VOLUMEN IS NOT NULL AND
PESO IS NOT NULL AND CONTENIDO IS NOT NULL)
    OR (TIPO='COMERCIAL' AND VOLUMEN IS NULL AND PESO IS NULL AND CONTENIDO I
NULL)),
  CONSTRAINT RESERVAS_CK2 CHECK((TIPO='COMERCIAL' AND TIPO_SILLA is not null
and TIPO_SILLA IN ('ECONOMICA','EJECUTIVA'))
    OR (TIPO='CARGA' and TIPO_SILLA is null))
);


/*Triggers*/


/*Generar numero de reserva */
CREATE SEQUENCE ISIS2304B261620.num_reserva
START WITH 1
INCREMENT BY 1;

create or replace trigger generar_num_reserva
before insert on RESERVAS
for each row
begin
  :new.numero_reserva := num_reserva.nextVal;
end;
/
```

Veáse que es necesario crear las tablas y secuencia en este orden, pues las dependencias que tienen unas tablas sobre otras podría generar conflictos en la creación.

c. Lógica de los nuevos requerimientos:

- **RF12:** Se pensó en la implementación de un grafo, que contenga como nodos los aeropuertos que hacen parte de los vuelos relevantes, con los arcos siendo la conexión que existe al haber un vuelo que tenga como origen y destino los aeropuertos dados, y el peso de su arco el parámetro decidido por entrada.

- **RF13:** Primero se confirma si la identidad del pasajero es válida, luego luego se procede en este orden a:
    - Revisar si se hizo en las últimas 24 horas
    - Actualizar el inventario del viaje
    - Eliminar la reserva.
- **RF14:** Se hace una iteración de RF13 con los viajes que se desean cancelar, consiguiendo el número de todas las reservas involucradas en el viaje completo, y aplicándose a cada una.
- **RF15:** Para iniciar, se recuperan los datos de todas las reservas asociadas al viaje a cancelar. Luego se procede a iterar sobre estas reservas sobre el cual:
    - Se registra una nueva reserva al viajero asociado a la reserva del viaje a cancelar con los mismos parámetros (RF12)
    - Se elimina la reserva relacionada al vuelo a cancelar (RF13)

  Finalmente, se elimina el viaje de la base de datos.
- **RF16:** Se itera sobre los datos de los clientes, creando la reserva al viajero con los parámetros dados para todos (RF12), y en caso de ser carga, se hace uso de los mismos parámetros, adicionando
- **RFC5:** Se revisa si el usuario que hace el pedido tiene permisos para acceder (Si es un cliente, su tipo de id y su id), o para un administrador su id de administrador. Luego se recuperan los datos pertinentes en las reservas y se devuelven al usuario.
- **RFC6:** Se revisa si el usuario tiene permisos para acceder a la información de la aeronave (es creador de la aeronave o gerente) y según el parámetro de consulta que se pide, calcula la información y la retorna.

Propiedades ACID

a. Atomicidad: Se procura que solo se empiecen los cálculos cuando toda la información necesaria sea recuperada, para cada método se ejecutan las acciones bajo un *try-catch*, a un *SqlException*, la cual en caso de atraparla, se hace un *rollback* de toda la transacción, con fin de que no quede incompleta en la base de datos.
b. Consistencia: Ya son existentes ciertas restricciones en las relaciones de las bases de datos para evitar datos incoherentes (como un viaje de pasajeros asociado con una aeronave de carga), o un trigger que genera el número de reserva de un vuelo.
c. Isolación: El sistema de base de datos procura que acción seguir si alguien recupera la información mientras otro trata de actualizarla. (Manejo de abrazos mortales y otros)
d. Durabilidad: Manejado en los servidores de las bases de datos. Se hace uso de *logs* y el esquema de respaldo de datos descrito en el punto ***3.e.***

3. Construcción de la aplicación.

e. Esquema de respaldo de datos:

Como garantía principal, y para solucionar problemas relacionados con la falta de energía, y otros fenómenos naturales asociados a esto, sería crucial tener una fuente secundaria de energía que pueda cumplir con las necesidades de *backup* de la base de datos en caso que ocurra una de estas eventualidades.

Para el respaldo de datos, se puede hacer uso de un sistema con acceso concurrente a la base de datos, para que tenga actualización inmediata, y pueda implementarse un **log** en un dispositivo de almacenamiento secundario (*no volátil*). Este log tendrá la información de todas las operaciones que están ocurriendo en la base de datos, con datos relevantes como:

- Id transacción, tipo de operación, id registro.
- Apuntador al anterior registro asociado a la transacción.
- Registro de actualización:
    - Estado anterior
    - Estado posterior
- Registros de commit, rollback
- Datos de cursores.

Otra herramienta útil para el respaldo de datos, sería el uso de **checkpoints**, con el fin de poder volver a posiciones en las que la base de datos era coherente, con fin de poder recobrar la funcionalidad de la base de datos después de alguna eventualidad.

El análisis del último checkpoint, junto al log de las operaciones de las transacciones y una infraestructura capaz de ofrecer energía después de una eventualidad, debería ser suficiente para retornar a la base de datos a un estado actualizado y coherente.