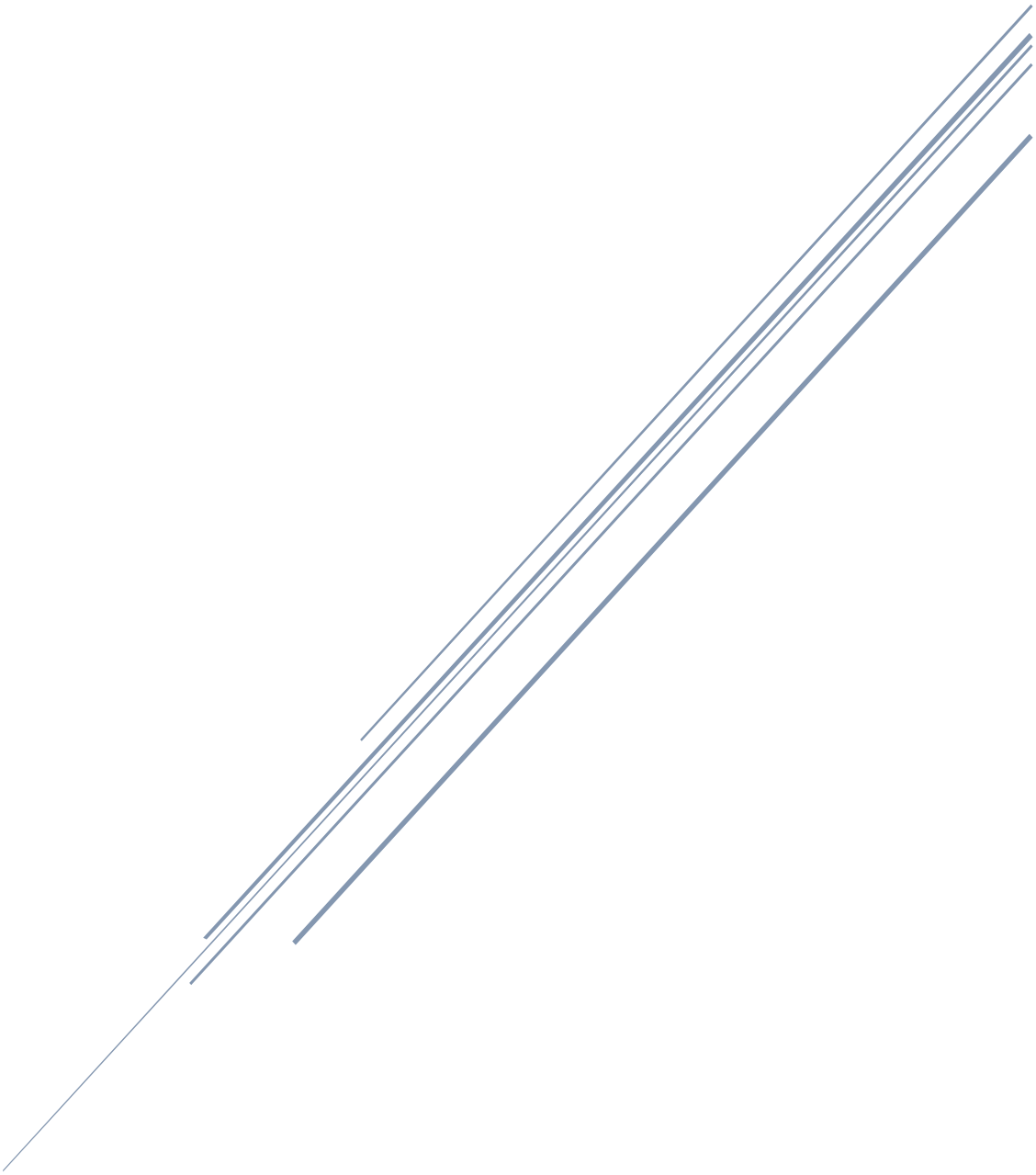
BASES DE DADES AVANÇADES

Pràctica 1



Judit Domènech Miró
Karen Samsó Muñoz

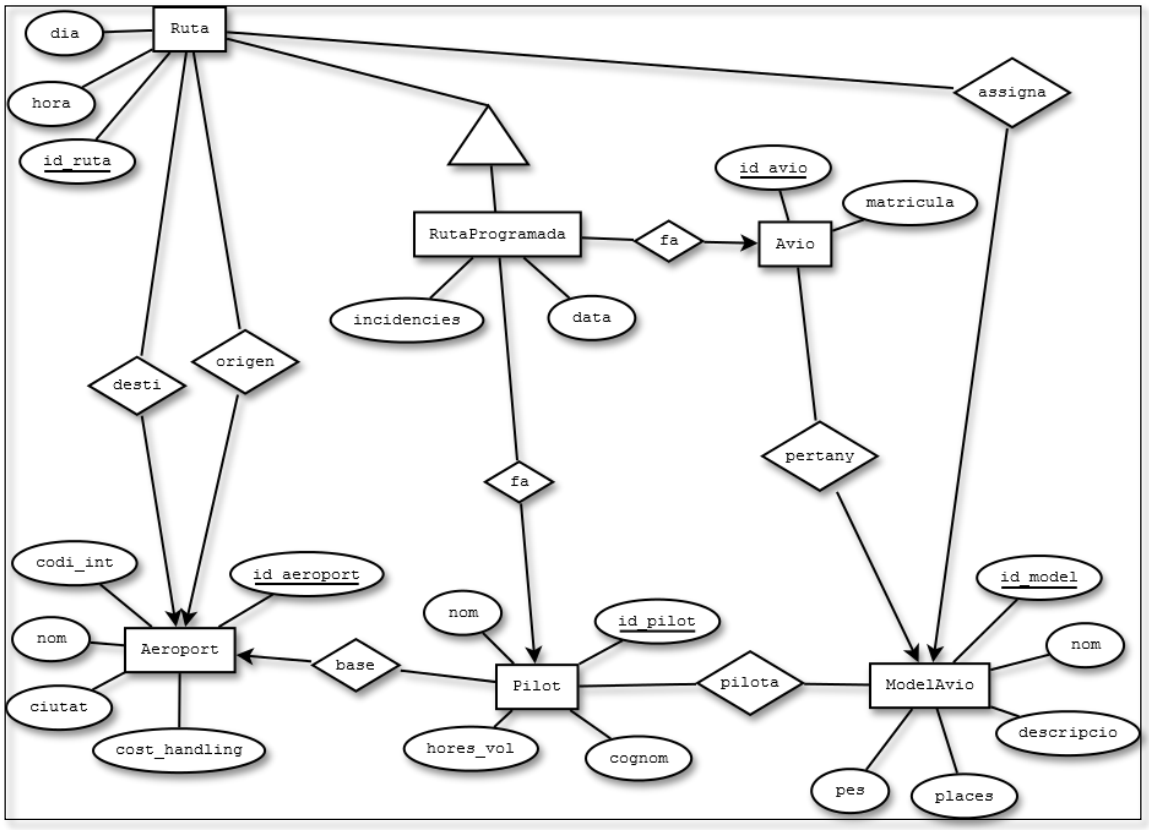
Introducció

En aquesta pràctica tenim com a objectiu realitzar una aplicació per gestionar una aerolínia de forma simplificada, d'aquesta manera aprendrem a integrar l'ús de bases de dades en aplicacions utilitzant un llenguatge d'alt nivell com és Java. A més aprendrem a utilitzar Hibernate, un framework de Java per ORM *(Object Relational mapping)*, i fer possible la persistència de dades.

Desenvolupament

Diagrama E-R i Pas a taules

El primer pas per dissenyar la gestió de l'areolínia ha estat crear el diagrama Entitat-Relació i, seguidament, fer el pas a taules. El resultat i les explicacions es poden veure seguidament:



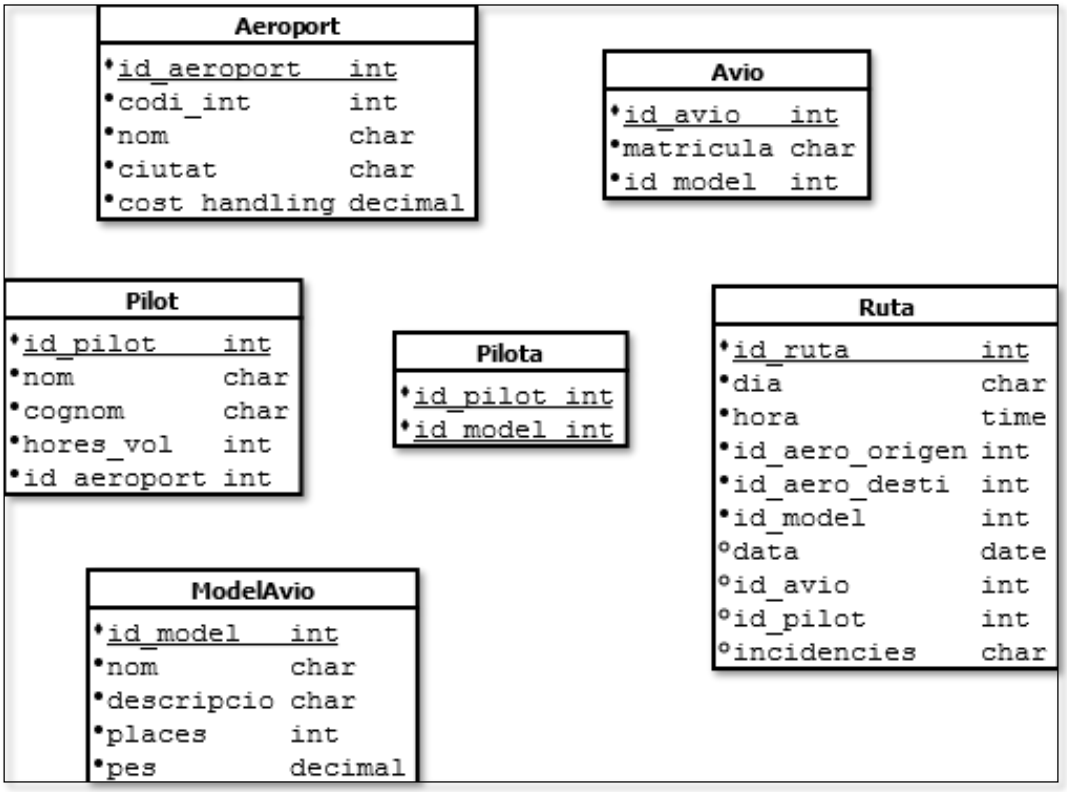
Il·lustració 1: Diagrama E-R

S'han creat les entitats especificades a l'enunciat, amb els atributs corresponents, i un "id" numèric per a cada una, de cara a que Hibernate necessita un identificador per a cada entitat definida.

L'única relació n-m que s'ha considerat és la formada per Pilot-ModelAvio (pilota), ja que es té en compte que un pilot pot manejar més d'un model d'avió, i un model pot ser manejat per diversos pilots. La resta de relacions són n-1.

El més destacat d'aquest diagrama és l'herència de l'entitat RutaProgramada (de l'entitat pare Ruta). S'ha considerat l'herència perquè hi ha certs atributs de la ruta que no es tenen en compte fins que aquesta està programada, per tant, com a concepte (en el model E-R) es tracten com dues entitats diferents.

De totes maneres, en la implementació de l'aplicació final, com es pot veure també en el pas a taules, s'ha decidit tenir únicament una entitat Ruta.



Il·lustració 2: Pas a taules

En el pas a taules, l'única relació del diagrama E-R representada és "Pilota" per tractar-se d'una relació n-m, la resta de relacions estan representades afegint l'atribut corresponent a l'identificador a l'entitat amb valor n de l'entitat amb valor 1.

S'ha decidit tenir només una taula Ruta, sense necessitat de fer una taula RutaProgramada, de manera que els atributs corresponents a la ruta programada estan sempre presents però poden tenir un valor nul. Aquests valors només s'ompliran quan anteriorment s'hagin omplert els altres (que són obligatoris).

Implementació de l'aplicació

Un cop fet el model i el pas a taules, hem passat a implementar l'aplicació a partir del codi d'exemple que se'ns va proporcionar per a la Pràctica 0. S'ha creat una classe Java per a cada entitat i, a diferència de l'exemple, el mapejat s'ha fet mitjançant les *annotations* de Hibernate.

L'aplicació funciona de manera que, un cop l'usuari ha fet el *log in*, apareix un menú amb les opcions **CRUD**, Afegir (Create), Consultar (Retrieve), Modificar (Update) i Borrar (Delete). Un cop seleccionada alguna d'aquestes opcions, **en tots els casos** apareix una **llista de les entitats** sobre les quals les podem realitzar.

- **Create:** Es demanen les dades necessàries per a crear un nou objecte de l'entitat seleccionada, es crea i s'afegeix a la base de dades. Per acabar, es fa un *commit* de la transacció. [Es recomana no equivocar-se de tipus de dada a l'hora d'entrar-les, ja que no hi ha gaire control d'errors per qüestió de falta de temps].

- **Retrieve:** Un cop seleccionada l'entitat a consultar, es fa una Query a la base de dades i s'obtenen tots els objectes, després es mostra per pantalla tota la informació disponible sobre aquests objectes.
- **Update:** Es demana a l'usuari que seleccioni l'entitat que vol editar i, seguidament, tots els objectes disponibles de tal entitat a la base de dades per tal que esculli quin vol actualitzar. Es demanen de nou totes les dades a l'usuari i es modifica l'objecte mitjançant els mètodes *set* corresponents. Finalment, es fa *commit* de la transacció a la base de dades.
- **Delete:** Igual que en el cas de l'*update*, es demana a l'usuari que seleccioni l'objecte a esborrar, s'elimina de la base de dades fent un *delete* i un *commit*.

A part de les opcions anteriors, el menú consta d'una opció especial (**Programar Ruta**) per programar una ruta, és a dir, omplir els camps buits d'una ruta que ha estat creada anteriorment però encara no havia estat programada (o editar una ruta ja programada). En aquesta opció, es mostren totes les rutes creades i es demana quina es vol programar (o editar) i, seguidament, es demanen els camps a omplir per considerar que una ruta està programada (tal com es mostra en el model E-R). Un cop fet això, es modifiquen aquests camps buits amb els mètodes *set* i es fa un *commit* de la transacció.

En tots els casos, es comprova que hi ha instàncies de totes les entitats necessàries (tant per mostrar-les, com per esborrar-les, com per assignar-les en cas de relacions entre entitats), en cas contrari, la transacció acaba sense fer cap canvi.

Fonts consultades

- Codi d'exemple del campus virtual.
- https://docs.jboss.org/hibernate/stable/annotations/reference/en/html_single/
- https://examples.javacodegeeks.com/enterprise-java/hibernate/hibernate-annotations-example/
- www.stackoverflow.com
- http://www.simplecodestuffs.com/auto-generate-primary-key-in-hibernate/
- http://www.java2s.com/Code/Java/Development-Class/Javaprogramtodemonstratemenuselection.htm
- http://docs.oracle.com/javase/8/docs/