|  |  |
| --- | --- |
| A picture of a winding road and trees  LAB6 - Report  Lab 6 - Java Persistence Query Language (JPQL) Exercise PQL.1 | Abstract  In this exercise you will write several JPQL queries against a given data set. To gain more experience on JPQL.  Davaabayar Battogtokh  CS544 – Enterprise Applications |

Contents

[Task description 2](#_Toc26291350)

[Runtime 3](#_Toc26291351)

[Comparison by graph 3](#_Toc26291352)

[Solution summary 4](#_Toc26291353)

[Initial query 5](file:///C:\Users\610567\Documents\Lab7Optimization\Lab7-Report.docx#_Toc26291354)

[Generated queries for each method 5](#_Toc26291355)

[@LazyCollection 5](file:///C:\Users\610567\Documents\Lab7Optimization\Lab7-Report.docx#_Toc26291356)

[Batch fetching / @BatchSize(size=<n>) 5](file:///C:\Users\610567\Documents\Lab7Optimization\Lab7-Report.docx#_Toc26291357)

[FetchMode.SUBSELECT 5](file:///C:\Users\610567\Documents\Lab7Optimization\Lab7-Report.docx#_Toc26291358)

[Join fetch query 5](file:///C:\Users\610567\Documents\Lab7Optimization\Lab7-Report.docx#_Toc26291359)

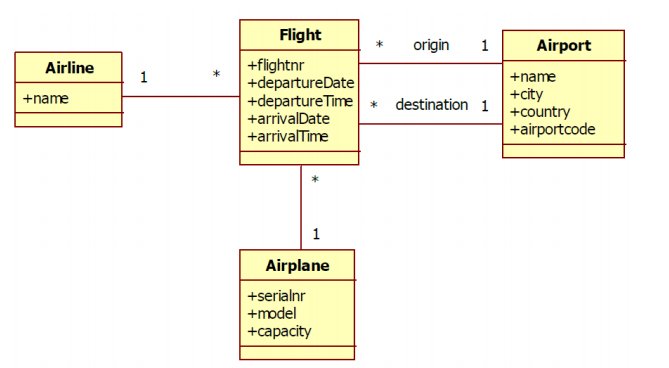
[Entity Graph 5](file:///C:\Users\610567\Documents\Lab7Optimization\Lab7-Report.docx#_Toc26291360)

# Lab 6 - Java Persistence Query Language (JPQL) Exercise JPQL.1

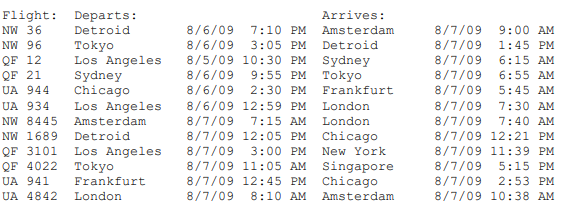
## The Setup:

In this exercise you will write several JPQL queries against a given data set. Start this exercise by downloading the provided Lab6-JPQL project from Sakai, and adding the Hibernate, MySQL and log4j dependencies to it.

**The Application:** The provided application uses the following flight scheduling problem domain.



Currently the application simply retrieves and prints all the flight info stored in the database. Running the application should produce output that contains something like:



# The Exercise

Look through the application code, and once you feel comfortable with what it does update the JPQL queries in Application.java to retrieve the following data:

1. All flights leaving the USA with a capacity > 500
2. All airlines that use A380 (model) airplanes
3. All fights using 747 planes that don’t belong to ‘Star Alliance’
4. All flights leaving before 12pm on 08/07/2009

**Hint**: you cannot use an implicit join to traverse a One-To-Many relationship, and when you use an explicit join you should also use select in your query.

**Hint**: just like most SQL dialects JPQL expects dates to be formatted according to the ISO 8601 standard which is 'yyyy-mm-dd', not the US mm/dd/yyyy format. Times should be formatted in the 24 hour 'hh:mm:ss' format.

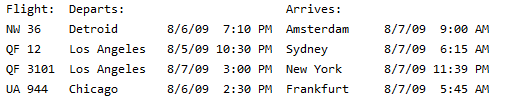
## A. All flights leaving the USA with a capacity > 500

**Query**

from Flight f where f.origin.country='USA' and f.airplane.capacity>500

*// a)* ***TODO: Flights leaving USA capacity > 500***System.***out***.println(**"Question A:"**);  
List<Flight> flights = em.createQuery(**"from Flight f where f.origin.country='USA' and f.airplane.capacity>500"**, Flight.**class**).getResultList();  
System.***out***.printf(**"%-9s%-31s%-31s\n"**, **"Flight:"**, **"Departs:"**,  
 **"Arrives:"**);  
**for** (Flight flight : flights) {  
 System.***out***.printf(**"%-7s %-12s %7s %8s %-12s %7s %8s\n"**,  
 flight.getFlightnr(), flight.getOrigin().getCity(),  
 flight.getDepartureDate(), flight.getDepartureTime(),  
 flight.getDestination().getCity(),  
 flight.getArrivalDate(), flight.getArrivalTime());  
}  
System.***out***.println();

**Output**

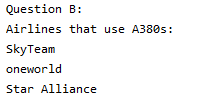


## B. All airlines that use A380 (model) airplanes

Select distinct al from Airline al join al.flights f join f.airplane ap where ap.model='A380'

*// b)* ***TODO: All airlines that use A380 airplanes***System.***out***.println(**"Question B:"**);  
List<Airline> airlines = em.createQuery(**"Select distinct al from Airline al join al.flights f join f.airplane ap where ap.model='A380'"**, Airline.**class**).getResultList();  
System.***out***.println(**"Airlines that use A380s:"**);  
**for** (Airline airline : airlines) {  
 System.***out***.println(airline.getName());  
}  
System.***out***.println();

**Output**



## C. All fights using 747 planes that don’t belong to ‘Star Alliance’

from Flight f where f.airplane.model='747' and f.airline.name not like 'Star Alliance'

*// c)* ***TODO: Flights using 747 planes that don't belong to Star Alliance***System.***out***.println(**"Question C:"**);  
flights = em.createQuery(**"from Flight f where f.airplane.model='747' and f.airline.name not like 'Star Alliance'"**, Flight.**class**).getResultList();  
System.***out***.printf(**"%-9s%-31s%-31s\n"**, **"Flight:"**, **"Departs:"**,  
 **"Arrives:"**);  
**for** (Flight flight : flights) {  
 System.***out***.printf(**"%-7s %-12s %7s %8s %-12s %7s %8s\n"**,  
 flight.getFlightnr(), flight.getOrigin().getCity(),  
 flight.getDepartureDate(), flight.getDepartureTime(),  
 flight.getDestination().getCity(),  
 flight.getArrivalDate(), flight.getArrivalTime());  
}  
System.***out***.println();

**Output:**



## D. All flights leaving before 12pm on 08/07/2009

*// d) TODO: All flights leaving before 12pm on 08/07/2009*System.*out*.println("Question D:");  
TypedQuery<Flight> query = em.createQuery("from Flight f where f.departureDate < '2009-08-07 12:00:00 PM'", Flight.class);  
flights = query.getResultList();  
System.*out*.printf("%-9s%-31s%-31s\n", "Flight:", "Departs:",  
 "Arrives:");  
for (Flight flight : flights) {  
 System.*out*.printf("%-7s %-12s %7s %8s %-12s %7s %8s\n",  
 flight.getFlightnr(), flight.getOrigin().getCity(),  
 flight.getDepartureDate(), flight.getDepartureTime(),  
 flight.getDestination().getCity(),  
 flight.getArrivalDate(), flight.getArrivalTime());  
}  
System.*out*.println();

from Flight f where f.departureDate < '2009-08-07 12:00:00 PM'



## Provided Code

**Current code:**

**package** edu.mum.cs544;  
  
**import** java.util.List;  
  
**import** edu.mum.cs544.model.Airline;  
**import** edu.mum.cs544.model.Flight;  
**import** java.text.DateFormat;  
**import** java.util.Locale;  
  
**import** javax.persistence.EntityManager;  
**import** javax.persistence.EntityManagerFactory;  
**import** javax.persistence.Persistence;  
**import** javax.persistence.TypedQuery;  
  
**public class** App {  
  
 **private static** EntityManagerFactory *emf*;  
  
 **public static void** main(String[] args) **throws** Exception {  
 *emf* = Persistence.*createEntityManagerFactory*(**"cs544"**);  
   
 EntityManager em = *emf*.createEntityManager();  
 em.getTransaction().begin();  
  
 *// a)* ***TODO: Flights leaving USA capacity > 500*** System.***out***.println(**"Question A:"**);  
 List<Flight> flights = em.createQuery(**"from Flight "**, Flight.**class**).getResultList();  
 System.***out***.printf(**"%-9s%-31s%-31s\n"**, **"Flight:"**, **"Departs:"**,  
 **"Arrives:"**);  
 **for** (Flight flight : flights) {  
 System.***out***.printf(**"%-7s %-12s %7s %8s %-12s %7s %8s\n"**,  
 flight.getFlightnr(), flight.getOrigin().getCity(),  
 flight.getDepartureDate(), flight.getDepartureTime(),  
 flight.getDestination().getCity(),  
 flight.getArrivalDate(), flight.getArrivalTime());  
 }  
 System.***out***.println();  
  
 em.getTransaction().commit();  
 em.close();  
  
 em = *emf*.createEntityManager();  
 em.getTransaction().begin();  
  
 *// b)* ***TODO: All airlines that use A380 airplanes*** System.***out***.println(**"Question B:"**);  
 List<Airline> airlines = em.createQuery(**"from Airline "**, Airline.**class**).getResultList();  
 System.***out***.println(**"Airlines that use A380s:"**);  
 **for** (Airline airline : airlines) {  
 System.***out***.println(airline.getName());  
 }  
 System.***out***.println();  
  
 em.getTransaction().commit();  
 em.close();  
  
 em = *emf*.createEntityManager();  
 em.getTransaction().begin();  
  
 *// c)* ***TODO: Flights using 747 planes that don't belong to Star Alliance*** System.***out***.println(**"Question C:"**);  
 flights = em.createQuery(**"from Flight "**, Flight.**class**).getResultList();  
 System.***out***.printf(**"%-9s%-31s%-31s\n"**, **"Flight:"**, **"Departs:"**,  
 **"Arrives:"**);  
 **for** (Flight flight : flights) {  
 System.***out***.printf(**"%-7s %-12s %7s %8s %-12s %7s %8s\n"**,  
 flight.getFlightnr(), flight.getOrigin().getCity(),  
 flight.getDepartureDate(), flight.getDepartureTime(),  
 flight.getDestination().getCity(),  
 flight.getArrivalDate(), flight.getArrivalTime());  
 }  
 System.***out***.println();  
  
 em.getTransaction().commit();  
 em.close();  
  
 em = *emf*.createEntityManager();  
 em.getTransaction().begin();  
  
 DateFormat df = DateFormat.*getDateInstance*(DateFormat.***SHORT***,  
 Locale.***US***);  
 DateFormat tf = DateFormat.*getTimeInstance*(DateFormat.***SHORT***,  
 Locale.***US***);  
  
 *// d)* ***TODO: All flights leaving before 12pm on 08/07/2009*** System.***out***.println(**"Question D:"**);  
 TypedQuery<Flight> query = em.createQuery(**"from Flight "**, Flight.**class**);  
 flights = query.getResultList();  
 System.***out***.printf(**"%-9s%-31s%-31s\n"**, **"Flight:"**, **"Departs:"**,  
 **"Arrives:"**);  
 **for** (Flight flight : flights) {  
 System.***out***.printf(**"%-7s %-12s %7s %8s %-12s %7s %8s\n"**,  
 flight.getFlightnr(), flight.getOrigin().getCity(),  
 flight.getDepartureDate(), flight.getDepartureTime(),  
 flight.getDestination().getCity(),  
 flight.getArrivalDate(), flight.getArrivalTime());  
 }  
 System.***out***.println();  
 em.getTransaction().commit();  
 em.close();  
 }  
}

## Solution code for App.class

**package** edu.mum.cs544;  
  
**import** java.util.List;  
  
**import** edu.mum.cs544.model.Airline;  
**import** edu.mum.cs544.model.Flight;  
**import** java.text.DateFormat;  
**import** java.util.Locale;  
  
**import** javax.persistence.EntityManager;  
**import** javax.persistence.EntityManagerFactory;  
**import** javax.persistence.Persistence;  
**import** javax.persistence.TypedQuery;  
  
**public class** App {  
  
 **private static** EntityManagerFactory *emf*;  
  
 **public static void** main(String[] args) **throws** Exception {  
 *emf* = Persistence.*createEntityManagerFactory*(**"cs544"**);  
   
 EntityManager em = *emf*.createEntityManager();  
 em.getTransaction().begin();  
  
 *// a)* ***TODO: Flights leaving USA capacity > 500*** System.***out***.println(**"Question A:"**);  
 List<Flight> flights = em.createQuery(**"from Flight f where f.origin.country='USA' and f.airplane.capacity>500"**, Flight.**class**).getResultList();  
 System.***out***.printf(**"%-9s%-31s%-31s\n"**, **"Flight:"**, **"Departs:"**,  
 **"Arrives:"**);  
 **for** (Flight flight : flights) {  
 System.***out***.printf(**"%-7s %-12s %7s %8s %-12s %7s %8s\n"**,  
 flight.getFlightnr(), flight.getOrigin().getCity(),  
 flight.getDepartureDate(), flight.getDepartureTime(),  
 flight.getDestination().getCity(),  
 flight.getArrivalDate(), flight.getArrivalTime());  
 }  
 System.***out***.println();  
  
 em.getTransaction().commit();  
 em.close();  
  
 em = *emf*.createEntityManager();  
 em.getTransaction().begin();  
  
 *// b)* ***TODO: All airlines that use A380 airplanes*** System.***out***.println(**"Question B:"**);  
 List<Airline> airlines = em.createQuery(**"Select distinct al from Airline al join al.flights f join f.airplane ap where ap.model='A380'"**, Airline.**class**).getResultList();  
 System.***out***.println(**"Airlines that use A380s:"**);  
 **for** (Airline airline : airlines) {  
 System.***out***.println(airline.getName());  
 }  
 System.***out***.println();  
  
 em.getTransaction().commit();  
 em.close();  
  
 em = *emf*.createEntityManager();  
 em.getTransaction().begin();  
  
 *// c)* ***TODO: Flights using 747 planes that don't belong to Star Alliance*** System.***out***.println(**"Question C:"**);  
 flights = em.createQuery(**"from Flight f where f.airplane.model='747' and f.airline.name not like 'Star Alliance'"**, Flight.**class**).getResultList();  
 System.***out***.printf(**"%-9s%-31s%-31s\n"**, **"Flight:"**, **"Departs:"**,  
 **"Arrives:"**);  
 **for** (Flight flight : flights) {  
 System.***out***.printf(**"%-7s %-12s %7s %8s %-12s %7s %8s\n"**,  
 flight.getFlightnr(), flight.getOrigin().getCity(),  
 flight.getDepartureDate(), flight.getDepartureTime(),  
 flight.getDestination().getCity(),  
 flight.getArrivalDate(), flight.getArrivalTime());  
 }  
 System.***out***.println();  
  
 em.getTransaction().commit();  
 em.close();  
  
 em = *emf*.createEntityManager();  
 em.getTransaction().begin();  
  
 DateFormat df = DateFormat.*getDateInstance*(DateFormat.***SHORT***,  
 Locale.***US***);  
 DateFormat tf = DateFormat.*getTimeInstance*(DateFormat.***SHORT***,  
 Locale.***US***);  
  
 *// d)* ***TODO: All flights leaving before 12pm on 08/07/2009*** System.***out***.println(**"Question D:"**);  
 TypedQuery<Flight> query = em.createQuery(**"from Flight f where f.departureDate < '2009-08-07 12:00:00 PM'"**, Flight.**class**);  
 flights = query.getResultList();  
 System.***out***.printf(**"%-9s%-31s%-31s\n"**, **"Flight:"**, **"Departs:"**,  
 **"Arrives:"**);  
 **for** (Flight flight : flights) {  
 System.***out***.printf(**"%-7s %-12s %7s %8s %-12s %7s %8s\n"**,  
 flight.getFlightnr(), flight.getOrigin().getCity(),  
 flight.getDepartureDate(), flight.getDepartureTime(),  
 flight.getDestination().getCity(),  
 flight.getArrivalDate(), flight.getArrivalTime());  
 }  
 System.***out***.println();  
 em.getTransaction().commit();  
 em.close();  
 }  
}