# Databases Advanced Exam – Most Wanted

Racing has become an essential part of some people’s life. It’s the energy that flows through their veins. However some races get out of control creating massive traffic disasters, which forced the Laws to create The ARPU – Anti-Race Police Unit… A specialized unit, which does not spare even a penny, when it comes to stopping races. Their performance tuned cars are on par with most racers’ cars and they aren’t afraid of crashing them.

## Functionality Overview

The ARPU has hired you as their database developer, to implement a **database application**. The application should be able to easily **import** hard-formatted data from **XML** and **JSON** and **support functionality** for also **exporting** into the same formats. The application is called – **Most Wanted**.

## Project Skeleton Overview

You will be given a **Skeleton**, containing a **certain architecture** with **several classes**, some of which – completely empty. The **Skeleton** will include the **files** with which you will **seed** the **database**. You will also be given a bunch of **Unit Tests** which will assure that you **don’t write something broken**, because we are talking about a Federal Project.

## Model Definition (60 pts)

There are 6 main models that the **Most Wanted** database application should contain in its functionality.

Design them in the **most appropriate** way, considering the following **data constraints**:

### Town

* id – **integer** number, **primary identification field**.
* name – a **string** (**required**).

### District

* id – **integer** number, **primary identification field**.
* name – a **string** (**required**).
* town – a Town entity.

### Racer

* id – **integer** number, **primary identification field**.
* name – a **string** (**required**).
* age – an **integer** number.
* bounty – a **decimal** data type.
* homeTown – a Town entity.
* cars – a collection of Car entity.

### Car

* id – **integer** number, **primary identification field**.
* brand – a **string** (**required**).
* model – a **string** (**required**).
* price – a **decimal** data type.
* yearOfProduction – an **integer** number (**required**).
* maxSpeed – a **floating-point** data type.
* zeroToSixty – a **floating-point** data type.
* racer – a Racer entity.

### Race

* id – **integer** number, **primary identification field**.
* laps – **integer** number (**required**, **default** – **0**)
* district – a District entity (**required**).
* entries – a collection of RaceEntry entity.

### RaceEntry

* id – **integer** number, **primary identification field**.
* hasFinished – a **boolean** value.
* finishTime – **integer** number.
* car – a Car entity.
* racer – a Racer entity.

## Data Import (25 pts)

Use the provided **JSON** and **XML** files to populate the database with data. Import all the information from those files into the database.

**You are not allowed to modify the provided JSON and XML files.**

**ANY UNALLOWED DUPLICATE** data should be **ignored** and a message “Error: Duplicate Data!” should be printed.

**ANY INCORRECT** data should be **ignored** and a message “Error: Incorrect Data!” should be printed.

**ANY SUCCESSFUL** data import should **result** in a message “Succesfully imported {entityClass} – {entityField}.”.

The entityField depends on the entityClass:

* For Towns, Districts, Racers – {name}.
* For Cars – a string **composed** in the following format – “{brand} {model} @ {yearOfProduction}”
* For Race, RaceEntry – {id};

### JSON Import (10 pts)

#### Towns (towns.json)

|  |
| --- |
| **towns.json** |
| [  { "name" : "München" },  { "name" : "Maastricht" },  . . .  ] |

|  |
| --- |
| Successfully imported Town - München.  Successfully imported Town - Maastricht.  . . . |

#### Districts (districts.json)

|  |
| --- |
| **districts.json** |
| [  { "name" : "Larry", "townName" : "Maastricht" },  { "name" : "Transport", "townName" : "München" },  . . .  ] |

|  |
| --- |
| Successfully imported District - Larry.  Successfully imported District - Transport.  . . . |

#### Racers (racers.json)

|  |
| --- |
| **racers.json** |
| [  {  "name" : "Clarence Callahan",  "age" : 34,  "bounty" : 164177.22,  "townName" : "Troyes"  },  . . .  ] |

|  |
| --- |
| Successfully imported Racer - Clarence Callahan.  . . . |

#### Cars (cars.json)

|  |
| --- |
| **cars.json** |
| [  {  "brand" : "Volvo",  "model" : "C70",  "price" : 487452.02,  "yearOfProduction" : 2010,  "maxSpeed" : 161.6,  "zeroToSixty" : 2.24,  "racerName" : "Brigit Speller"  },  . . .  ] |

|  |
| --- |
| Successfully imported Car - Volvo C70 @ 2010.  . . . |

### XML Import (15 ptx)

The ARPU are not very smart, so their data seeds are quite messed up.   
You will need to **import** the RaceEntries first, and then import all Races with their entries.

#### Race Entries (race-entries.xml)

|  |
| --- |
| **race-entries.xml** |
| <?xml version="1.0" encoding="utf-8"?>  <race-entries>  <race-entry has-finished="true" finish-time="741.12" car-id="269">  <racer> Max Philpott</racer>  </race-entry>  <race-entry has-finished="false" finish-time="822.96" car-id="242">  <racer> Wylie Gareisr</racer>  </race-entry>  <race-entry has-finished="true" finish-time="156.57" car-id="220">  <racer> Murial Jedrzejewicz</racer>  </race-entry>  ...  </race-entries> |

|  |
| --- |
| Successfully imported RaceEntry - 1.  Successfully imported RaceEntry - 2.  Successfully imported RaceEntry - 3.  . . . |

#### Races (races.xml)

|  |
| --- |
| **races.xml** |
| <?xml version="1.0" encoding="utf-8"?>  <races>  <race>  <laps>4</laps>  <district-name>Jackson</district-name>  <entries>  <entry id="10">  <entry id="9">  </entries>  </race>  ...  </races> |

|  |
| --- |
| Successfully imported Race - 1.  . . . |

## Data Export (25 pts)

Get ready to export the data you’ve imported in the previous task. Here you will have some pretty complex database querying. Export the data in the formats specified below, In the corresponding folders and files.

### JSON Export

#### Racing Towns (racingTowns.json)

**Export all towns** which have **any** **racers** in them:

* **Export** only the **town name** (as name) and **count of racers** (as racers).
* **Order** them **descending**, by **count** of **racers** they have, and then by **town** **name alphabetically**.
* **Export** the **data** in **JSON** format.

|  |
| --- |
| **racingTowns.json** |
| [  {  "name" : "Maochen",  "racers" : 6  },  {  "name" : "Phan Thong",  "racers" : 6  },  {  "name" : "Jiangxi",  "racers" : 4  },  . . .  ] |

#### Racing Cars (racingCars.json)

**Export** **all racers** which **have any cars**:

* **Export** the racer’s **name**, **age** (but **ONLY** if it is **NOT NULL**), **list** of **cars**.
  + In case the **racer’s age** property is **NULL**, do **NOT** include it in the **JSON**.
* The **cars** should be **strings** in the following format: “{brand} {model} {yearOfProduction}”.
* **Order** them **descending**, by **count** of **cars** they have, and then by **racer name alphabetically**.
* **Export** the **data** in **JSON** format.

|  |
| --- |
| **racingCars.json** |
| [  {  "name" : "Batsheva Warry",  "cars" : [  "Mercury Grand Marquis 2011",  "Aston Martin DBS 2012",  "Mazda MX-5 1991",  "Jeep Cherokee 1996",  "Suzuki Grand Vitara 2012",  "Suzuki SJ 1993",  "BMW X5 2001"  ]  },  {  "name" : "Gard Wolseley",  "age" : 43,  "cars" : [  "Dodge Caravan 2007",  "Chevrolet Cruze 2013",  "Volvo S60 2005",  "Saturn S-Series 1998",  "GMC 2500 Club Coupe 1993",  "Chevrolet Tracker 1998",  "Chevrolet Silverado 2500 2007"  ]  },  . . .  ] |

### XML Export

#### Most Wanted (most-wanted.xml)

**Export** the **racer** with the **most entries** in **races**.

* Export the **racer’s** **name**, and his **entries**.
* For **each** of his **entries**, **export** the **car** and the **finish time**.
* The **cars** should be **strings** in the following format: “{brand} {model} @ {yearOfProduction}”.
* **Order** the **entries** **ascending**, by **finish time**.
* **Export** the **data** in **XML** format.

|  |
| --- |
| **most-wanted.xml** |
| <?xml version="1.0" encoding="utf-8"?>  <most-wanted>  <racer name="Clarence Callahan">  <entries>  <entry>  <finish-time>126.97</finish-time>  <car>Chevrolet Tahoe @ 2001</car>  </entry>  . . .  </entries>  </racer>  </most-wanted> |