Design Report: Datastore Implementation

Jimmy Trac | 101624964 | Distinction Datastore Implementation Design Report

---

The formatting of this document may not suit PDF format.

Design Report

Abstract

This report outlines the design, development, and implementation of a *Kantai Collection* Database in MySQL. Exploring both the use-cases and overall design of the datastore system, the raw data was gathered from the *Kantai Collection Wiki* in table form and was processed to match the schema of the database. The report also outlines various queries for common use-cases such as creating an equipment loadout for a ship or reassigning a ship to another fleet. The database was implemented using MySQL Workbench 8.0 CE on a Windows 10 Machine. The implementation was successful and extended upon to include additional functionality through the `ALTER` keyword. The database holds the information on approximately 200 ships.

Overview of Database

The database (or *datastore*) will be used to collate and store information relating to the game *Kantai Collection*.

Background Information

Kantai Collection is an online browser-based game whereby players collect and command WW2-era ships, assembling fleets and battling against an unknown fleet of ships. With highly simplified statistics of warships, there are vast opportunities for classification and organisation. The term 'WW2-era ships' loosely describes the various ships within the game, with the earliest ships built and launched around the First World War (circa. 1914-1918), the Kongou-class ships launched around 1911-1915.

Kantai Collection was developed by Kadokawa Games and published to Japanese game site DMM.com. It was launched in 2013 and was originally developed in Flash, and has moved to HTML5 mid-2018 with the "Phase 2" Update. Gameplay consists of naval warfare, development, maintenance, supply, and operational planning.

Main Uses of the Datastore

The datastore will contain information relating to the different statistics and equipment of each ship, along with information relating to their nation or current fleet. Ships will require information relating to their statistics such as their firepower, armour, or current fuel and ammunition. Furthermore, each ship will have a limited number of equipment slots for certain types of equipment, such as naval guns, torpedoes, or RADAR.

Within typical gameplay, the basic information about each ship will rarely change, save for a few exceptions such as upgrades. On the other hand, information such as the current ammunition level, fuel level, or morale of the ship will be accessed and changed often.

MySQL Database

The reason for choosing MySQL over NoSQL is due to the structured nature of the data and the already-available data in table form. The regular groupings and connections within the data were relatively straightforward and logical to establish, such as ship type to the hull code, and ships to nations, or naval bases. Data did not exist for ships belonging to a specific fleet nor the equipment loadout, however, those can be modified through transactions and queries.

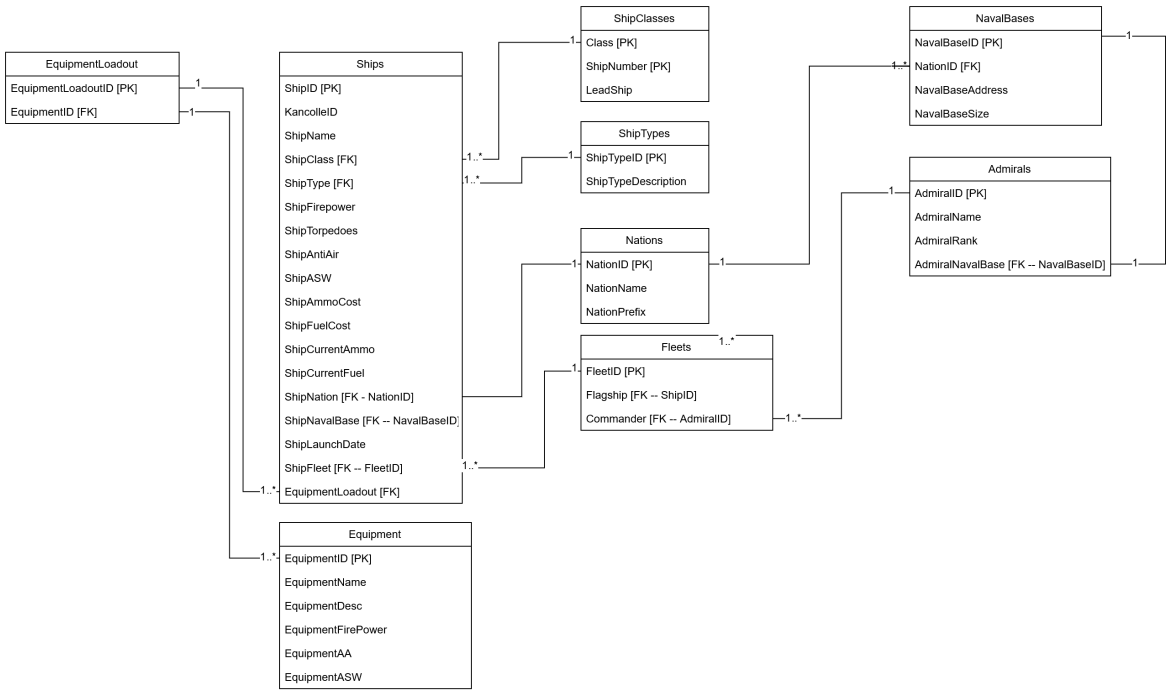
Typical Use-Cases

Due to the nature of the game, there are various use-cases for the datastore system:

- Moving a ship into and out of a fleet
- Accessing information about a ship
- Upgrading (known as *modernising*) a ship
- Mounting Equipment on a ship
- Repairing a ship

Illustration of Top-Level Design

The following figure is an Entity-Relationship Diagram showing the relationships between each table, normalised to 3NF.



Attribute Data Types

The following table outlines the data types for each attribute based off the table and the reason for the data type.

ShipTypes Table

Attribute	Data Type	Justification
`shipTypeID`	`varchar(3)`	Ship hull classification codes, notably NATO classification codes are in the form `XXX`, such as `BB`, `BBV`, `CV`, `DD` to name a few. This refers to one ship type only and is not only unique, but human-readable and descriptive. For example, the 61st Battleship may be referred to as `BB-61`. Furthermore, the `varchar` justification is that primary keys do not change, and therefore will take up less space compared to `char`
`ShipTypeDescription`	`varchar(100)`	Connecting `BB` to Battleship and `DD` to Destroyer, again, these names do not change at all and new codes are created for derivations of ships, such as `BBV` for Aviation Battleship. Hence, to reduce memory usage, `varchar` reduces the size of the attribute.

Primary Key: `shipTypeID`,
`ShipTypeID` refers directly to the ship type, and is fully unique between hull classifications.

Nations Table

Attribute	Data Type	Justification

Attribute	Data Type	Justification
`nationID`	`int`	Auto-incrementing unique ID Number
`nationName`	`varchar(100)`	Name of the Nation, e.g. `Australia` or `United Kingdom`. These names do not change, and therefore `varchar` reduces storage requirements.
`nationPrefix`	`varchar(100)`	Prefix of a nation, e.g. `HMAS` for *Her Majesty's Australian Ship* or `KMS` as an approximate classification for *Kriegsmarine*.

Primary Key: `nationID`,
A unique primary key.

NavalBases Table

Attribute	Data Type	Justification
`navalBaseID`	`int`	Auto-incrementing unique ID Number
`navalBaseNationID`	`int`	**Foreign Key**, connecting to Nations Table.
`navalBaseName`	`char(200)`	Name of a naval base. Doesn't change and therefore uses `varchar`
`navalBaseAddress`	`char(200)`	Address of a naval base, doesn't change, and therefore uses `varchar`
`navalBaseSize`	`int`	Capacity of a Naval Base, used in calculations and requires manipulation, therefore is an `int`.

Primary Key: `navalBaseID`,
A unique primary key.

Admirals Table

Attribute	Data Type	Justification
`admiralID`	`int`	Auto-incrementing unique ID Number
`admiralFirstName`	`varchar(100)`	First name, rarely changes, `varchar`
`admiralLastName`	`varchar(100)`	Last name, rarely changes, `varchar`
`admiralRank`	`char(100)`	Rank of an admiral, may not necessarily be of official rank, generally refers to codes (e.g. `VCADML`) over `Vice Admiral`, however, there is no set system. It would be less restrictive to use an open-ended system instead of strict enforcement due to each country's different systems.
`admiralNavalBase`	`int`	The foreign key associated with the naval base.

Primary Key: `admiralID`,
A unique primary key.

Equipment Table

Attribute	Data Type	Justification
`equipmentID`	`int auto_increment`	Auto-incrementing unique ID Number
`equipmentKancolleID`	`int`	
`equpimentName`	`varchar(100)`	
`equipmentDesc`	`varchar(100)`	
`equipmentFirePower`	`int`	
`equipmentAA`	`int`	
`equpimentASW`	`int`	

Primary Key: `ShipTypeID`,
`ShipTypeID` refers directly to the ship type, and is fully unique between hull classifications.

Fleets Table

Attribute	Data Type	Justification
`fleetID`	`int`	Auto-incrementing unique ID Number
`admiralID`	`int`	

Primary Key: `fleetID`

Later updated to add the attribute `fleetName`, with the name being a `Char(100)`

Ships Table

Attribute	Data Type	Justification
`shipID`	`int`	Auto-incrementing unique ID Number
`shipKancolleID`	`char(4)`	The Kantai Collection ID (e.g. 113, 113a, 113b)
`shipName`	`varchar(50)`	Name of a ship. Doesn't change. If so, new ship.
`shipClass`	`varchar(50)`	Class of a ship, Doesn't change. If so, new ship.
`shipType`	`varchar(4)`	Foreign Key `REFERENCES ShipTypeTable(ShipTypeID)`
`shipFirePower`	`int`	Ship Stat
`shipTorpedoes`	`int`	^
`shipAA`	`int`	^
`shipASW`	`int`	^
`shipAmmoCost`	`int`	^
`shipFuelCost`	`int`	^
`shipNation`	`int`	Foreign Key `REFERENCES Nations(nationID)`
`shipNavalBase`	`int`	Foreign Key `REFERENCES NavalBases(navalbaseID)`
`shipFleet`	`int`	Foreign Key `REFERENCES Fleets(fleetID)`

Primary Key: `shipID`,
A unique ID compared to the `KancolleID` as it can be auto-incremented

EquipmentLoadout Table

Attribute	Data Type	Justifcation
`equpimentLoadoutID`	`int`	Auto-incrementing unique ID Number
`equippedShip`	`int`	Foreign Key `REFERENCES Ships(ShipID)`
`equipmentID`	`int`	Foreign Key `REFERENCES Equipmment(EquipmentID)`

Primary Key: `EquipmentLoadoutID`,
A weak entity /associative entity to describe the many-to-many relationship between Ships and their Equipment

---

Database Development

Scripts Necessary for Design

The datastore will be designed in MySQL and tested locally on a virtual instance of Ubuntu 19.04 'Disco Dingo' utilising MySQL Workbench and/or MySQL Workbench 6.0 CE in Windows 10.

The scripts necessary will involve the construction and proper references of each table and their relationships with each other. It will also include the data type required by MySQL specified for each attribute within each table.

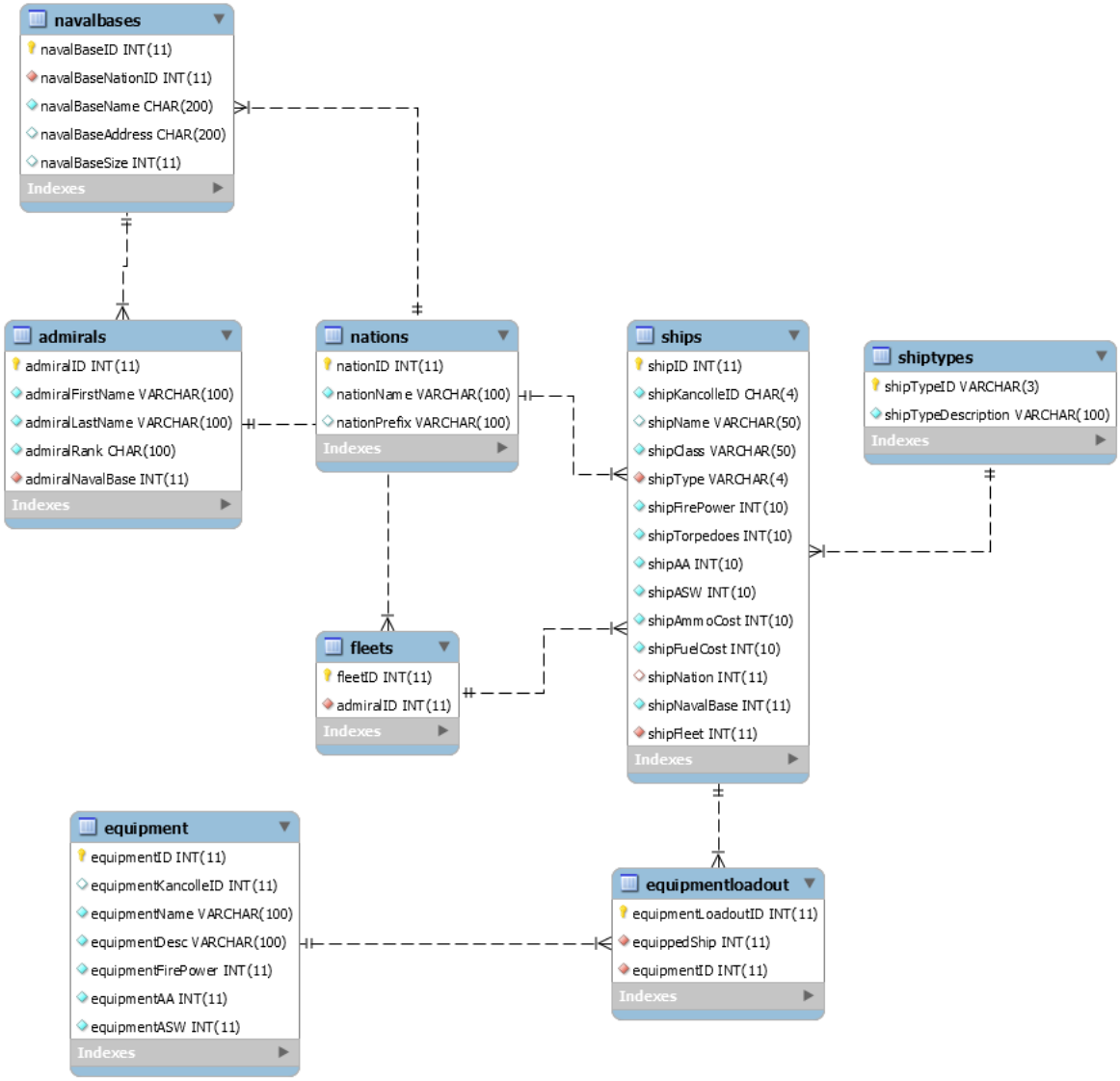
Constructing the Database

The following script constructs a database that was refined to better accept relational data and data that can be found easily.

```sql
create schema kancolledb;
use kancolledb;

set AUTOCOMMIT = 0;

create table ShipTypes (
    shipTypeID varchar(3) not null,
    shipTypeDescription varchar(100) not null,

    primary key (shipTypeID)
);


create table Nations (
    nationID int not null AUTO_INCREMENT,
    nationName varchar(100) not null,
    nationPrefix varchar(100),

    primary key (nationID)
);

create table NavalBases (
    navalBaseID int not null AUTO_INCREMENT,
    navalBaseNationID int not null,
    navalBaseName char(200) not null,
    navalBaseAddress char(200),
    navalBaseSize int,

    primary key (navalBaseID),
    foreign key (navalBaseNationID) references Nations(nationID)

);

create table Admirals (
    admiralID int not null AUTO_INCREMENT,
    admiralFirstName varchar(100) not null,
    admiralLastName varchar(100) not null,
    admiralRank char(100) not null,
    admiralNavalBase int not null,

    primary key (admiralID),
    foreign key (admiralNavalBase) references NavalBases(navalBaseID)

);


create table Equipment (
    equipmentID int not null AUTO_INCREMENT,
    equipmentKancolleID int,
    equipmentName varchar(100) not null,
    equipmentDesc varchar(100) not null,
    equipmentFirePower int not null,
    equipmentAA int not null,
```

```sql
    equipmentASW int not null,

    primary key (equipmentiD)

);

create table Fleets (
    fleetID int not null AUTO_INCREMENT,
    admiralID int not null,

    primary key (fleetID),
    foreign key (admiralID) references Admirals(admiralID)
);

create table Ships (
    shipID int not null AUTO_INCREMENT,
    shipKancolleID char(4) not null,
    shipName varchar(50),
    shipClass varchar(50) not null,
    shipType varchar(4) not null,
    shipFirePower int unsigned not null,
    shipTorpedoes int unsigned not null,
    shipAA int unsigned not null,
    shipASW int unsigned not null,
    shipAmmoCost int unsigned not null,
    shipFuelCost int unsigned not null,
    shipNation int,
    shipNavalBase int not null,
    shipFleet int not null,

    primary key (shipID),
    foreign key (shipType) references ShipTypes(shipTypeID),
    foreign key (shipNation) references Nations(nationID),
    foreign key (shipFleet) references Fleets(fleetID)
);

create table EquipmentLoadout (
    equipmentLoadoutID int not null AUTO_INCREMENT,
    equippedShip int not null,
    equipmentID int not null,

    primary key (equipmentLoadoutID),
    foreign key (equippedShip) references Ships(ShipID),
    foreign key (equipmentID) references Equipment(equipmentID)
);
```

The final design of the database, according to MySQL Workbench's reverse-engineering tool can be found in the following figure.



Collecting the necessary data

No database is without data, hence, the transformation from table data into database will be discussed in the following section to outline the tools, methodology, and reasoning behind the design of the database. All of the data gathered for the database comes from the Kantai Collection Wiki (http://en.kancollewiki.net/wiki/Ship_list) with tables such as the following:

Ship list

ID	Name	Class	Type														
001	Nagato	Nagato	Battleship	82	0	31	0	12	20	80	75	24	12	Slow	Long	100	130
001a	Nagato Kai	Nagato	Battleship	90	0	33	0	15	32	90	85	24	12	Slow	Long	100	160
002	Mutsu	Nagato	Battleship	82	0	31	0	12	3	80	75	24	12	Slow	Long	100	130
002a	Mutsu Kai	Nagato	Battleship	90	0	33	0	15	6	90	85	24	12	Slow	Long	100	160
003	Ise	Ise	Battleship	74	0	28	0	10	15	74	70	22	12	Slow	Long	85	120
004	Hyuuga	Ise	Battleship	74	0	28	0	10	15	74	70	22	12	Slow	Long	85	120
005	Yukikaze	Kagerou	Destroyer	10	24	12	24	6	50	16	7	50	0	Fast	Short	15	20
005a	Yukikaze Kai	Kagerou	Destroyer	12	28	16	27	8	60	32	14	67	0	Fast	Short	15	20

The tables were imported into Microsoft Excel for further processing. Imported into a sheet into excel, the raw data import is used as a reference sheet in order to derive the required tables.

ID	Name	Class	Type														
1	Nagato	Nagato	Battleship	82	0	31	0	12	20	80	75	24	12	Slow	Long	100	130
001a	Nagato Kai	Nagato	Battleship	90	0	33	0	15	32	90	85	24	12	Slow	Long	100	160
2	Mutsu	Nagato	Battleship	82	0	31	0	12	3	80	75	24	12	Slow	Long	100	130
002a	Mutsu Kai	Nagato	Battleship	90	0	33	0	15	6	90	85	24	12	Slow	Long	100	160
3	Ise	Ise	Battleship	74	0	28	0	10	15	74	70	22	12	Slow	Long	85	120
4	Hyuuga	Ise	Battleship	74	0	28	0	10	15	74	70	22	12	Slow	Long	85	120

shipID	shipKancolleID	shipName	shipClass	shipType	shipFirePower	shipTorpedoes	shipAA	shipASW	shipFuelCost	shipAmmoCost	ShipNation	ShipNavalBase	ShipFleet
1	1	Nagato	Nagato	BB	82	0	31	0	100	130	1	1	1
2	001a	Nagato Kai	Nagato	BB	90	0	33	0	100	160	1	1	1
3	2	Mutsu	Nagato	BB	82	0	31	0	100	130	1	1	1
4	002a	Mutsu Kai	Nagato	BB	90	0	33	0	100	160	1	1	1
5	3	Ise	Ise	BB	74	0	28	0	85	120	1	1	1
6	4	Hyuuga	Ise	BB	74	0	28	0	85	120	1	1	1

The second figure shows the table generated by the excel spreadsheet, showing all the attributes of the *Ships* table. Using tools such as `VLOOKUP`, the excel spreadsheet converts `Battleship` to `BB` and removes unnecessary attributes. Furthermore, default values are added for `ShipNavalBase` and `ShipFleet` whilst `ShipNation` was added in manually.

Scripts for Typical Use-Cases

The scripts for the database will be based on the use-cases described previously:

- Moving a ship into and out of a fleet
- Accessing information about a ship
- Upgrading (known as *modernising*) a ship
- Mounting Equipment on a ship
- etc.

Accessing information about a ship

Accessing basic information about a ship

```
select shipKancolleID as 'Kancolle ID', CONCAT(nationPrefix, " ", shipName) as 'Ship Designation',
shipClass as 'Class', shipTypeDescription as 'Type'
from Ships s
    join ShipTypes t
        on s.shipType=t.shipTypeID
    join Nations n
        on s.shipNation=n.nationID
where shipKancolleID = '113'
order by shipName asc;
```

Output:

```
select shipKancolleID as 'Kancolle ID', CONC
from Ships s
    join ShipTypes t
        on s.shipType=t.shipTypeID
    join Nations n
        on s.shipNation=n.nationID
where shipKancolleID = '113'
```

Result Grid | Filter Rows: | Export: | W

Kancolle ID	Ship Designation	Class	Type
113	IJN Zuihou Kai	Shouhou	Light Carrier

Removing the restriction gives:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Kancolle ID	Ship Designation	Class	Type
017a	IJN Ayanami Kai	Ayanami	Destroyer
171	KMS Bismarck	Bismarck	Fast Battleship
121	IJN Chitose Carrier Kai Ni	Chitose	Light Carrier
122	IJN Chiyoda Carrier Kai Ni	Chitose	Light Carrier
291	FS Commandant Teste	Comma...	Seaplane Tender
291a	FS Commandant Teste Kai	Comma...	Seaplane Tender
334	IJN Etorofu	Etorofu	Destroyer Escort
334a	IJN Etorofu Kai	Etorofu	Destroyer Escort
11	IJN Fubuki	Fubuki	Destroyer
011a	IJN Fubuki Kai	Fubuki	Destroyer
34	IJN Fumizuki	Mutsuki	Destroyer
034a	IJN Fumizuki Kai	Mutsuki	Destroyer
26	IJN Fusou	Fusou	Battleship
026a	IJN Fusou Kai	Fusou	Aviation Battleship
311	USSRS Gangut	Gangut	Fast Battleship
232	KMS Graf Zeppelin	Graf Ze...	Standard Aircraft Carrier
232a	KMS Graf Zeppelin Kai	Graf Ze...	Standard Aircraft Carrier
170	IJN Hamakaze	Kagerou	Destroyer
170a	IJN Hamakaze Kai	Kagerou	Destroyer
23	IJN Haruna	Kongou	Fast Battleship
023a	IJN Haruna Kai	Kongou	Fast Battleship

More detailed information about a ship

Obtaining more detailed information about a ship or ships can be given by the following query:

```
use kancolledb;
select
    shipID as 'Ship ID',
    fleetName as 'Current Fleet',
    shipKancolleID as 'Kancolle ID',
    CONCAT(nationPrefix, " ", shipName) as 'Ship Designation',
    shipClass as 'Class',
    shipTypeDescription as 'Type',
    shipFirePower as 'Firepower',
    shipTorpedoes as 'Torpedoes',
    shipAA as 'Anti-Air',
```

```
    shipASW as 'Anti-Submarine',
    shipAmmoCost as 'Ammunition Consumption',
    shipFuelCost as 'Fuel Consumption',
    navalBaseName as 'Current Base'

from Ships s
    join ShipTypes t
        on s.shipType=t.shipTypeID
    join Nations n
        on s.shipNation=n.nationID
    join Fleets f
        on f.fleetID = s.shipFleet
    join NavalBases nb
        on s.shipNavalBase = nb.navalBaseID
-- where shipKancolleID = '113'
order by shipName asc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Ship ID	Current Fleet	Kancolle ID	Ship Designation	Class	Type	Firepower	Torpedoes	Anti-Air	Anti-Submarine	Ammunition Consumption	Fuel Consumption	Current Base
9	Combined Fleet	6	IJN Akagi	Akagi	Standard Aircraft Carrier	0	0	32	0	55	60	Hashirajima Anchorage
10	Combined Fleet	006a	IJN Akagi Kai	Akagi	Standard Aircraft Carrier	0	0	35	0	75	75	Hashirajima Anchorage
134	Combined Fleet	132	IJN Akigumo	Kagerou	Destroyer	10	24	9	24	20	15	Hashirajima Anchorage
135	Combined Fleet	132a	IJN Akigumo Kai	Kagerou	Destroyer	8	28	22	27	20	15	Hashirajima Anchorage
149	Combined Fleet	161	IJN Akitsu Maru	Hei	Amphibious Assault Ship	6	0	13	0	10	40	Hashirajima Anchorage
157	Combined Fleet	166	IJN Akitsu Maru Kai	Hei	Amphibious Assault Ship	8	0	15	0	25	45	Hashirajima Anchorage
103	Combined Fleet	88	IJN Arashio	Asashio	Destroyer	10	24	9	21	20	15	Hashirajima Anchorage
104	Combined Fleet	088a	IJN Arashio Kai	Asashio	Destroyer	12	28	16	24	20	15	Hashirajima Anchorage
97	Combined Fleet	85	IJN Asashio	Asashio	Destroyer	10	24	12	21	20	15	Hashirajima Anchorage
98	Combined Fleet	085a	IJN Asashio Kai	Asashio	Destroyer	12	28	16	24	20	15	Hashirajima Anchorage
31	Combined Fleet	17	IJN Ayanami	Ayanami	Destroyer	10	27	12	20	20	15	Hashirajima Anchorage

Altering Ships

Ships can be upgraded for increased performance. The following scripts update ship statistics and other values.

```
use kancolledb;

update Ships
    set
        shipFirePower = shipFirePower + 1
    where
        shipID = 9;

```

The altered value can be found with the red arrow below:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Ship ID	Current Fleet	Kancolle ID	Ship Designation	Class	Type	Firepower	Torpedoe
9	Combined Fleet	6	IJN Akagi	Akagi	Standard Aircraft Carrier	1	0
10	Combined Fleet	006a	IJN Akagi Kai	Akagi	Standard Aircraft Carrier	0	0
134	Combined Fleet	132	IJN Akigumo	Kagerou	Destroyer	10	24

Grouping Queries

The following queries provide information on certain groups of ships.

Ships Per Nation

```
use kancolledb;

select nationName as 'Nation', count(shipNation) as 'Number of Ships'
from ships s
join nations n
on s.shipNation = n.nationID
group by shipNation;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Nation	Number of Ships
Imperial Japan	176
USSR	1
United Kingdom	4
United States of America	3
Kriegsmarine	5
Kingdom of Italy	7
France	4

Ships by Class

```
use kancolledb;

select shipClass as 'Ship Class', count(shipClass) as 'Number of Ships'
from ships
group by shipClass
order by count(shipClass) desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Ship Class	Number of Ships
Mutsuki	20
Kagerou	16
Fubuki	12
Kuma	10
Nagara	8
Kongou	8
Asashio	8

Ships by Type

```
use kancolledb;

select shipType as 'Ship Hull Code', shipTypeDescription as 'Ship Type', count(shipType) as 'Number of Ships'
from ships s
join shipTypes t on s.shipType = t.shipTypeID
group by shipType;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Ship Hull Code	Ship Type	Number of Ships
AO	Fleet Oiler	1
AV	Seaplane Tender	2
BB	Battleship	9
BBV	Aviation Battleship	2
CA	Heavy Cruiser	8
CAV	Aviation Cruiser	3
CL	Light Cruiser	28
CLp	Training Cruiser	2
CLT	Torpedo Cruiser	2
CV	Standard Aircraft Carrier	13
CVB	Armored Carrier	2
CVL	Light Carrier	15
DD	Destroyer	77
DE	Destroyer Escort	4
FBB	Fast Battleship	14
LHA	Amphibious Assault Ship	2
SSV	Aircraft Carrying Submarine	16

Fleet Scripts

To make fleets more interesting, I've altered the Fleets table to include the attribute *FleetName*

```sql
use kancolledb;

ALTER TABLE Fleets
  ADD fleetName char(100) not null;
```

To update the names of the fleets:

```sql
use kancolledb;

update Fleets
    set
        fleetName = 'Combined Fleet'
    where
        fleetID = 1;
```

Querying for the Fleets table gives us:

Result Grid | Filter Rows:

fleetID	admiralID	fleetName
1	1	Combined Fleet
2	2	
NULL	NULL	NULL

As the second fleet has no name, the attribute is empty.

Querying Ships in a given Fleet

```sql
use kancolledb;

select
    fleetName, shipKancolleID as 'Kancolle ID', CONCAT(nationPrefix, " ", shipName) as 'Ship Designation', shipClass as 'Class', shipTypeDescription as 'Type'
    from
        Ships s
    join
        ShipTypes t
        on s.shipType=t.shipTypeID
    join
        Nations n
        on s.shipNation=n.nationID
    join
        Fleets f
        on f.fleetID = s.shipFleet
    where f.fleetID = '1';
```

```
select
    fleetName, shipKancolleID as 'Kancolle ID', CONCAT(nationPrefix, " ", shipName) as 'Ship Designation'
    from
        Ships s
    join
        ShipTypes t
        on s.shipType=t.shipTypeID
    join
        Nations n
        on s.shipNation=n.nationID
    join
        Fleets f
        on f.fleetID = s.shipFleet
    where f.fleetID = '1';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fleetName	Kancolle ID	Ship Designation	Class	Type
Combined Fleet	1	IJN Nagato	Nagato	Battleship
Combined Fleet	001a	IJN Nagato Kai	Nagato	Battleship
Combined Fleet	2	IJN Mutsu	Nagato	Battleship
Combined Fleet	002a	IJN Mutsu Kai	Nagato	Battleship
Combined Fleet	3	IJN Ise	Ise	Battleship
Combined Fleet	4	IJN Hyuuga	Ise	Battleship
Combined Fleet	5	IJN Yukikaze	Kagerou	Destroyer
Combined Fleet	005a	IJN Yukikaze Kai	Kagerou	Destroyer

To move a ship between fleets:

```
use kancolledb;

update Ships
    set
        shipFleet = 2
    where
        shipID = 25;
```

Searching for Fleet 2 gives us:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fleetName	Kancolle ID	Ship Designation	Class	Type
Secondary Fleet	14	IJN Miyuki	Fubuki	Destroyer

Equipment Queries

Adding Equipment to a ship

The `kancolledb` database uses a weak entity to connect ships to their mounted equipment using the `EquipmentLoadout` table. The following query can be used to add equipment (the *Type 3 SONAR*) to a ship (*Murakumo*).

```
insert into equipmentLoadout(equippedShip, equipmentID)
values (27, 21);
```

Showing the Equipment on a ship

To query which ships have which pieces of equipment equipped, the following query shows the ship and any accompanying equipment.

```
use kancolledb;

select shipTypeDescription as 'Ship Type', shipName as 'Ship', e.equipmentName as 'Equipped',
e.equipmentDesc as 'Type', e.equipmentFirePower as 'Firepower', e.equipmentAA as 'AA', e.equipmentASW
as 'ASW'
from ships s
join shiptypes t on s.shipType = t.shipTypeID
join equipmentLoadout l on l.equippedShip = s.shipID
join equipment e on l.equipmentID = e.equipmentID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Ship Type	Ship	Equipped	Type	Firepower	AA	ASW
Fast Battleship	Richelieu	38cm Twin Gun Mount	Heavy Naval Gun	16	1	0
Fast Battleship	Richelieu	38cm Twin Gun Mount	Heavy Naval Gun	16	1	0
Destroyer	Murakumo	10cm Twin High-angle Mount + Anti-Aircraft Fir...	Dual Purpose Gun	3	10	0
Destroyer	Murakumo	Type 3 Depth Charge Launcher	Depth Charge Launchers	0	0	8
Destroyer	Murakumo	Type 3 SONAR	Sonar	0	0	10

Counting Equipment

Counting equipment by type uses:

```
use kancolledb;

select equipmentDesc as 'Equipment Type', count(equipmentDesc) as 'Number of Equipment' from
equipment
group by equipmentDesc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Equipment Type	Number of Equipment
Light Naval Gun	5
Dual Purpose Gun	6
Medium Naval Gun	2
Heavy Naval Gun	4
Depth Charge Launchers	2
Sonar	2

Conclusion

The distinction-level task is to analyse, design, develop, implement, and document a reasonably complex datastore with accompanying real-world data. Designed after an browser-based game, *Kantai Collection*, the Kantai Collection Database (known in schema as `kancolledb`) stores information on approximately 200 ships and a number of equipment with relationships between the various tables highlighting the overall structure of the game itself. The database was written in MySQL Workbench 8 on Windows 10, and all data gathered on the game, from the fan-made wiki *Kancolle Wiki*'s tables. The end product was an effective datastore solution containing all the required data along with appropriate queries and transactions to facilitate normal use of both the game and datastore itself. Improvements to the system could include additional restrictions on the number of ships per fleet, though this may be an application-level restriction, or additional information and relationships pertaining to ship classes and their relationship with the real-world ship counterparts.

Glossary

Term	Meaning

Term	Meaning
Kantai Collection	Name of the game in question; literally translates to *Fleet Collection*.
Kancolle	Shortening of **Kan**tai **Colle**ction.
Ships	May refer to Kantai Collection's *Fleet Girls, or Kantai-musume, shortened Kan-musu*, moe-anthropomorphic representation of real-world warships, notably of the 20th Century.
Modernisation	Upgrading a ship through the assimilation of others.
Admiral	In this case, it may also refer to the player as they assume the role of an admiral.
Rear-Admiral	Admiral rank, often there are two captains on a ship, one in the forward section and one in the back. In case one meets untimely end, the second can take command.
Commander	Commanders 'command' 'smaller' ships such as Destroyers up to Light Cruisers, Admirals often command flagships, which tend to be Heavy Cruiser or above. This is Kantai Collection-specific, however, and only roughly matches up with the real world.

References

W3Schools, *SQL Data Types for MySQL, SQL Server, and MS Access*, last accessed 10th September 2019,<https://www.w3schools.com/sql/sql_datatypes.asp>

Kantai Collection Wiki, *Ship List*, 2019, last accessed 24 October 2019, <http://en.kancollewiki.net/wiki/Ship_list>

Kantai Collection Wiki, *List of Main Guns by Stats*, 2019, last accessed 24 October 2019, <http://en.kancollewiki.net/wiki/List_of_Main_Guns_by_stats>

Kantai Collection Wiki, *List of Torpedoes by Stats*, 2019, last accessed 24 October 2019, <http://en.kancollewiki.net/wiki/List_of_Torpedoes_by_stats>

Kantai Collection Wiki, *List of Anti-Submarine Equipment by Stats*, 2019, last accessed 24 October 2019, <http://en.kancollewiki.net/wiki/List_of_Anti-Submarine_Equipment_by_stats>

Kantai Collection Wiki, *Kancolle Wiki*, last accessed 24 October 2019, <http://en.kancollewiki.net/wiki/Kancolle_Wiki>

Appendix: CSV Files