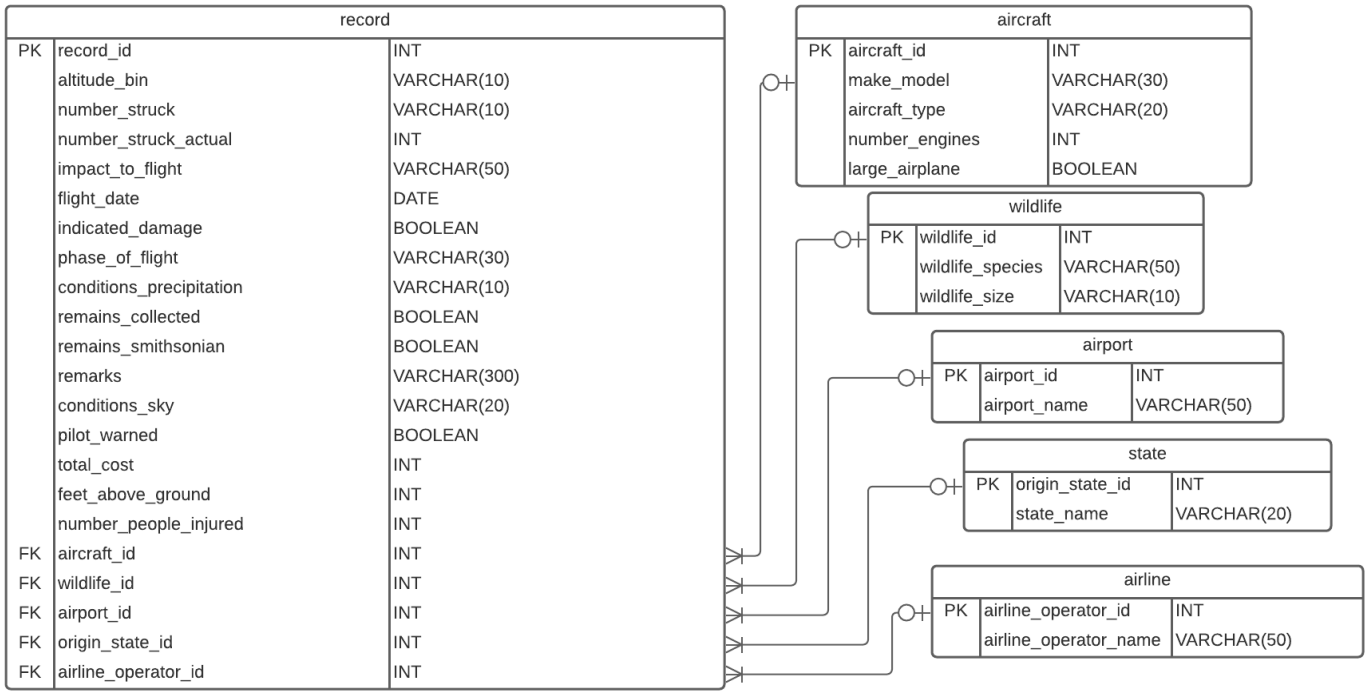
Practicum I / Design & Implement a Relational Database

Code

Author: Zihao Qiu

Email: qiu.ziha@northeastern.edu
(mailto:qiu.ziha@northeastern.edu)

1. ERD Diagram



2. Define Relational Schema

2.1 Connect to the local MySQL server

Hide

```
library(RMySQL)
db_user <- 'root'
db_password <- 'password'
db_name <- 'myDB'
db_host <- 'localhost'
db_port <- 3306
mydb <-  dbConnect(MySQL(), user = db_user, password = db_password,
                 dbname = db_name, host = db_host, port = db_port)
```

Note: all table and column names are normalized to meet the SQL convention, specifically lower cases and underscores.

2.2 The aircraft table and its definition

The reason for not using make_model as the primary key is that the same make_model may have different engines.

Hide

```
CREATE TABLE aircraft(
  aircraft_id INT,
  make_model VARCHAR(30),
  aircraft_type VARCHAR(20),
  number_engines INT,
  large_airplane BOOLEAN,
  PRIMARY KEY (aircraft_id)
);
```

2.3 The wilflife table and its definition

The wildlife_id field is an artificial primary key. There is no “AUTO INCREMENT” because it will be done in the R code chunk later.

Hide

```
CREATE TABLE wildlife(
  wildlife_id INT,
  wildlife_species VARCHAR(50),
  wildlife_size VARCHAR(10),
  PRIMARY KEY (wildlife_id)
);
```

2.4 The airport table and its definition

The airport table is a lookup table.

Hide

```
CREATE TABLE airport(
  airport_id INT,
  airport_name VARCHAR(50),
  PRIMARY KEY (airport_id)
);
```

2.5 The state table and its definition

The state table is a lookup table.

Hide

```
CREATE TABLE state(
  origin_state_id INT,
  state_name VARCHAR(20),
  PRIMARY KEY (origin_state_id)
);
```

2.6 The airline table and its definition

The airline table is a lookup table.

Hide

```
CREATE TABLE airline(
  airline_operator_id INT,
  airline_operator_name VARCHAR(50),
  PRIMARY KEY (airline_operator_id)
);
```

2.7 The record table and its definition

Naturally, the record id will be used as the table's primary key.
This table can be divided further, for example the phase_of_flight field can be divided into a look up table.

Hide

```
CREATE TABLE record(
  record_id INT,
  altitude_bin VARCHAR(10),
  number_struck VARCHAR(10),
  number_struck_actual INT,
  impact_to_flight VARCHAR(50),
  flight_date DATE,
  indicated_damage BOOLEAN,
  phase_of_flight VARCHAR(30),
  conditions_precipitation VARCHAR(10),
  remains_collected BOOLEAN,
  remains_smithsonian BOOLEAN,
  remarks VARCHAR(300),
  conditions_sky VARCHAR(20),
  pilot_warned BOOLEAN,
  total_cost INT,
  feet_above_ground INT,
  number_people_injured INT,
  aircraft_id INT,
  wildlife_id INT,
  airport_id INT,
  origin_state_id INT,
  airline_operator_id INT,
  PRIMARY KEY (record_id),
  FOREIGN KEY (aircraft_id) REFERENCES aircraft(aircraft_id),
  FOREIGN KEY (wildlife_id) REFERENCES wildlife(wildlife_id),
  FOREIGN KEY (airport_id) REFERENCES airport(airport_id),
  FOREIGN KEY (origin_state_id) REFERENCES state(origin_state_id),
  FOREIGN KEY (airline_operator_id) REFERENCES airline(airline_operator_id)
);
```

3. Load Data into R FROM CSV File

Note that column names were manually normalized.

Hide

```
path <- "C:/Users/zqiu9/Documents/R Projects/"
```

```
Warning: The working directory was changed to C:/Users/zqiu9/Documents inside a notebook chunk.
The working directory will be reset when the chunk is finished running. Use the knitr root.dir o
ption in the setup chunk to change the working directory for notebook chunks.
```

Hide

```
fn <- "BirdStrikesData.csv"
fileName <- paste(path, fn, sep = "/")
birdDF <- read.csv(fileName, header = TRUE, col.names = c("record_id", "aircraft_type", "airport
_name", "altitude_bin", "make_model", "number_struck", "number_struck_actual", "impact_to_fligh
t", "flight_date", "indicated_damage", "number_engines", "airline_operator_name", "state_name",
"phase_of_flight", "conditions_precipitation", "remains_collected", "remains_smithsonian", "rema
rks", "wildlife_size", "conditions_sky", "wildlife_species", "pilot_warned", "total_cost", "feet
_above_ground", "number_people_injured", "large_airplane"), stringsAsFactors = FALSE)
```

Hide

```
head(birdDF)
```

	record_id <int>	aircraft_type <chr>	airport_name <chr>	altitude_bin <chr>	make_model <chr>	num <chr
1	202152	Airplane	LAGUARDIA NY	> 1000 ft	B-737-400	Over
2	208159	Airplane	DALLAS/FORT WORTH INTL ARPT	< 1000 ft	MD-80	Over
3	207601	Airplane	LAKEFRONT AIRPORT	< 1000 ft	C-500	Over
4	215953	Airplane	SEATTLE-TACOMA INTL	< 1000 ft	B-737-400	Over
5	219878	Airplane	NORFOLK INTL	< 1000 ft	CL-RJ100/200	Over
6	218432	Airplane	GUAYAQUIL/S BOLIVAR	< 1000 ft	A-300	Over

6 rows | 1-7 of 26 columns

4. Realize Relational Schema

Load required packages for manipulating SQL in R data frame. Note that options(sqldf.driver = “SQLite”) was used for solving a compatibility issue with the sqldf and the RMySQL package.

Hide

```
library(sqldf)
```

```
Loading required package: gsubfn
Loading required package: proto

Attaching package: ‘gsubfn’

The following object is masked _by_ ‘.GlobalEnv’:

    fn

sqldf will default to using MySQL
```

Hide

```
library(RSQLite)
options(sqldf.driver = "SQLite")
```

Temperately disable foreign key check.

Hide

```
SET FOREIGN_KEY_CHECKS=0;
```

0 rows

This block was used for allowing the server interact with local files.

Hide

```
SET @@GLOBAL.local_infile = 1;
```

0 rows

4.1 Realize airline table

Note that airline[,1] <- seq(1, n_airline) was used for assigning artificial primary key in this table.

Hide

```
airline <- sqldf("SELECT DISTINCT 1 AS airline_operator_id, airline_operator_name FROM birdDF")
n_airline <- nrow(airline)
airline[,1] <- seq(1, n_airline)
head(airline)
```

	airline_operator_id <int>	airline_operator_name <chr>
1	1	US AIRWAYS*
2	2	AMERICAN AIRLINES
3	3	BUSINESS
4	4	ALASKA AIRLINES
5	5	COMAIR AIRLINES
6	6	UNITED AIRLINES

6 rows

Write R data frame into MySQL server.

Hide

```
dbWriteTable(mydb, "airline", airline, append = T, row.names = FALSE, field.types = c(airline_op
erator_id = "INT", airline_operator_name = "VARCHAR(50)"))
```

```
[1] TRUE
```

Check if it works.

Hide

```
SELECT * FROM airline
LIMIT 10
```

airline_operator_id <int>	airline_operator_name <chr>
1	US AIRWAYS*
2	AMERICAN AIRLINES
3	BUSINESS
4	ALASKA AIRLINES
5	COMAIR AIRLINES
6	UNITED AIRLINES
7	AIRTRAN AIRWAYS
8	AIRTOURS INTL
9	AMERICA WEST AIRLINES
10	EXECUTIVE JET AVIATION

1-10 of 10 rows

4.2 Realize state table

Hide

```
state <- sqldf("SELECT DISTINCT 1 AS origin_state_id, state_name FROM birdDF")
n_state <- nrow(state)
state[,1] <- seq(1, n_state)
head(state)
```

	origin_state_id <int>	state_name <chr>
1	1	New York
2	2	Texas
3	3	Louisiana
4	4	Washington
5	5	Virginia

	origin_state_id	state_name
	<int>	<chr>
6	6	N/A

6 rows

Hide

```
dbWriteTable(mydb, "state", state, append = T, row.names = FALSE)
```

```
[1] TRUE
```

4.3 Realize airport table

Hide

```
airport <- sqldf("SELECT DISTINCT 1 AS airport_id, airport_name FROM birdDF")
n_airport <- nrow(airport)
airport[,1] <- seq(1, n_airport)
head(airport)
```

	airport_id	airport_name
	<int>	<chr>
1	1	LAGUARDIA NY
2	2	DALLAS/FORT WORTH INTL ARPT
3	3	LAKEFRONT AIRPORT
4	4	SEATTLE-TACOMA INTL
5	5	NORFOLK INTL
6	6	GUAYAQUIL/S BOLIVAR

6 rows

Hide

```
dbWriteTable(mydb, "airport", airport, append = T, row.names = FALSE)
```

```
[1] TRUE
```

4.4 Realize wildlife table

Hide

```
wildlife <- sqldf("SELECT DISTINCT 1 AS wildlife_id, wildlife_species, wildlife_size FROM birdD
F")
n_wildlife <- nrow(wildlife)
wildlife[,1] <- seq(1, n_wildlife)
head(wildlife)
```

	wildlife_id <int>	wildlife_species <chr>	wildlife_size <chr>
1	1	Unknown bird - medium	Medium
2	2	Rock pigeon	Small
3	3	European starling	Small
4	4	Unknown bird - small	Small
5	5	Canada goose	Large
6	6	Snow goose	Large

6 rows

Hide

```
dbWriteTable(mydb, "wildlife", wildlife, append = T, row.names = FALSE)
```

```
[1] TRUE
```

4.5 Realize aircraft table

Hide

```
aircraft <- sqldf("SELECT DISTINCT 1 AS aircraft_id, make_model, aircraft_type, number_engines,
 large_airplane FROM birdDF ORDER BY make_model")
n_aircraft <- nrow(aircraft)
aircraft[,1] <- seq(1, n_aircraft)
aircraft$large_airplane <- ifelse(aircraft$large_airplane=="Yes", 1, 0)
head(aircraft)
```

	aircraft_id <int>	make_model <chr>	aircraft_type <chr>	number_engines <chr>	large_airplane <dbl>
1	1	A-10A	Airplane		0
2	2	A-23 MUSKATEER	Airplane		0
3	3	A-300	Airplane	2	0
4	4	A-300	Airplane	4	0
5	5	A-310	Airplane	2	0
6	6	A-318	Airplane	2	0

6 rows

Hide

```
dbWriteTable(mydb, "aircraft", aircraft, append = T, row.names = FALSE)
```

```
[1] TRUE
```

4.6 Realize record table

This R chunk contains the following steps:
1. Select relevant data from the R data frame
2. Convert the data type of remains_collected, remains_smithsonian, and pilot_warned to a format that can be understood by SQL.
3. Convert total_cost and feet_above_ground from char to int format.
4. Convert flight_date from char to date format.
5. Assign the foreign key values.

Hide

```
record <- sqldf("SELECT record_id, altitude_bin, number_struck, number_struck_actual, impact_to_
flight, flight_date, indicated_damage, phase_of_flight, conditions_precipitation, remains_collec
ted, remains_smithsonian, remarks, conditions_sky, pilot_warned, total_cost, feet_above_ground,
 number_people_injured FROM birdDF")

record$remains_collected <- ifelse(record$remains_collected==TRUE, 1, 0)
record$remains_smithsonian <- ifelse(record$remains_smithsonian==TRUE, 1, 0)
record$pilot_warned <- ifelse(record$pilot_warned=="Y", 1, 0)
record$total_cost <- as.numeric(gsub(",", "", record$total_cost))
record$feet_above_ground <- as.numeric(gsub(",", "", record$feet_above_ground))
record$flight_date <- as.Date(record$flight_date, format="%m/%d/%Y")

for (r in 1:nrow(record)){
  a <- airline$airline_operator_id[which(airline$airline_operator_name == birdDF$airline_operato
r_name[r])]
  record$airline_operator_id[r] <- a
  s <- state$origin_state_id[which(state$state_name == birdDF$state_name[r])]
  record$origin_state_id[r] <- s
  ap <- airport$airport_id[which(airport$airport_name == birdDF$airport_name[r])]
  record$airport_id[r] <- ap
  w <- wildlife$wildlife_id[which(wildlife$wildlife_species == birdDF$wildlife_species[r] & wild
life$wildlife_size == birdDF$wildlife_size[r])]
  record$wildlife_id[r] <- w
  ac <- aircraft$aircraft_id[which(aircraft$make_model == birdDF$make_model[r] & aircraft$aircra
ft_type == birdDF$aircraft_type[r] & aircraft$number_engines == birdDF$number_engines[r] & aircr
aft$large_airplane == birdDF$large_airplane[r])]
  record$aircraft_id[r] <- ac
}
head(record)
```

	record_id <int>	altitude_bin <chr>	number_struck <chr>	number_struck_actual <int>	impact_to_flight <chr>
1	202152	> 1000 ft	Over 100	859	Engine Shut Down
2	208159	< 1000 ft	Over 100	424	None
3	207601	< 1000 ft	Over 100	261	None
4	215953	< 1000 ft	Over 100	806	Precautionary Landing
5	219878	< 1000 ft	Over 100	942	None
6	218432	< 1000 ft	Over 100	537	None

6 rows | 1-7 of 21 columns

Hide

```
dbWriteTable(mydb, "record", record, append = T, row.names = FALSE)
```

```
[1] TRUE
```

5. Create a SQL query against your database to find the number of bird strike incidents for each airline upon take-off or climb

To get the number of incidents per airline, the best approach is to count how many times the airline names appear.

Hide

```
SELECT airline.airline_operator_name, COUNT(airline.airline_operator_name) count_during_climb_ta
ke_off
FROM record
JOIN airline
ON airline.airline_operator_id = record.airline_operator_id
WHERE record.phase_of_flight = 'Climb' OR record.phase_of_flight = 'Take-off run'
GROUP BY airline.airline_operator_name
ORDER BY airline.airline_operator_name
```

airline_operator_name <chr>	count_during_climb_take_off <dbl>
ABX AIR	51
ACM AVIATION	1
ADI SHUTTLE GROUP	5
AER LINGUS	2
AEROMEXICO	1

airline_operator_name <chr>	count_during_climb_take_off <dbl>
AIR AMERICA/TOTAL AIR	1
AIR BC	2
AIR CANADA	34
AIR CANADA JAZZ	20
AIR CARGO CARRIERS	3

1-10 of 210 rows Previous **1** 2 3 4 5 6 … 21 Next

6. Create a SQL query against your database to find the airports that had the most bird strike incidents (during any flight phase)

The following SQL chunk utilized subqueries. Get the number of incidents per airport first, then select the max out of it.

Hide

```
SELECT name airport_name, MAX(counts) count
FROM
  (SELECT a.airport_name name, COUNT(a.airport_name) counts
  FROM record r
  JOIN airport a
  ON a.airport_id = r.airport_id
  GROUP BY name) temp_table
```

airport_name <chr>	count <dbl>
LAGUARDIA NY	803

1 row

7. Create a SQL query against your database to find the number of bird strike incidents by year

Use YEAR() function to get the year of a DATE data type.

Hide

```
SELECT YEAR(flight_date) year, Count(record_id) count
FROM record
WHERE flight_date IS NOT NULL
GROUP BY year
ORDER BY year
```

year <int>	count <dbl>
2000	1367
2001	1230
2002	1681
2003	1568
2004	1692
2005	1853
2006	2159
2007	2301
2008	2258
2009	3247

1-10 of 12 rows Previous **1** 2 Next

8. Using the above data, build a column chart that visualizes the number of bird strikes incidents per year from 2008 to 2011 during take-off/climbing and during descent/approach/landing

Install and load the package for plotting.

Hide

```
library("ggplot2")
```

There are two queries and a UNION operator was used to combine them together. The result was stored in R data frame for plotting.

Hide

```
sqlCmd =
"
SELECT year, Count(year) count, 'take-off/climbing' phase_of_flight
FROM
  (SELECT YEAR(flight_date) year, phase_of_flight
  FROM record
  WHERE phase_of_flight = 'Climb' OR phase_of_flight = 'Take-off run') temp
WHERE year BETwEEN 2008 AND 2011
GROUP BY year

UNION

SELECT year, Count(year) count, 'descent/approach/landing' phase_of_flight
FROM
  (SELECT YEAR(flight_date) year, phase_of_flight
  FROM record
  WHERE phase_of_flight = 'Approach' or phase_of_flight = 'Landing Roll' OR phase_of_flight = 'D
escent') temp
WHERE year BETwEEN 2008 AND 2011
GROUP BY year
"
rdf = dbGetQuery(mydb, sqlCmd)
rdf
```

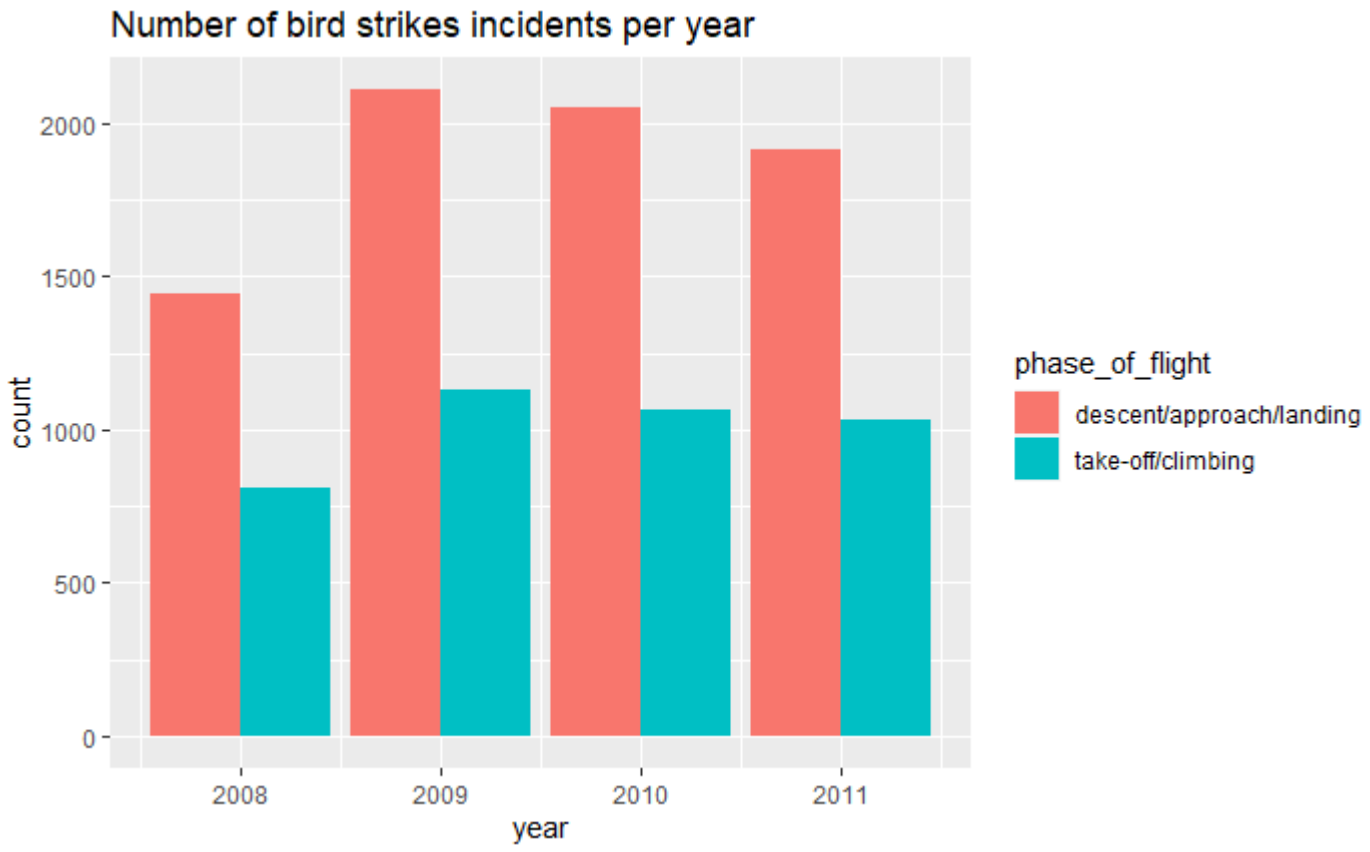
year <dbl>	count <dbl>	phase_of_flight <chr>
2009	1127	take-off/climbing
2008	810	take-off/climbing
2010	1062	take-off/climbing
2011	1030	take-off/climbing
2008	1442	descent/approach/landing
2009	2109	descent/approach/landing
2010	2053	descent/approach/landing
2011	1913	descent/approach/landing

8 rows

R code for plotting.

Hide

```
ggplot(rdf, aes(x = year, y = count, fill = phase_of_flight)) +
geom_bar(stat = "identity", position = "dodge") + ggtitle("Number of bird strikes incidents per
 year")
```



9. Create a stored procedure in MySQL

This stored procedure is used to delete a record by ID.

Hide

```
CREATE PROCEDURE delete_by_id (IN id INT)
  BEGIN
    DELETE FROM record where record_id = id;
  END
```

Record 202701 exists before running the stored procedure.

Hide

```
SELECT record_id from record where record_id = 202701;
```

record_id <int>
202701

1 row

Run the stored procedure.

Hide

```
CALL delete_by_id(202701);
```

0 rows

There is no records of this id after running the stored procedure.

Hide

```
SELECT record_id from record where record_id = 202701;
```

0 rows