Practicum I CS5200

Marin Witherspoon and Sumit Hawal

Summer Full 2023

```
library(RMySQL)
```

```
## Warning: package 'RMySQL' was built under R version 4.2.3
```

```
## Loading required package: DBI
```

```
library(RSQLite, quietly=T)
```

```
## Warning: package 'RSQLite' was built under R version 4.2.3
```

```
##
## Attaching package: 'RSQLite'
```

```
## The following object is masked from 'package:RMySQL':
##
##     isIdCurrent
```

```
library(RMySQL, quietly=T)

dbc <- dbConnect(RSQLite::SQLite(), ":memory:")
```

Connect to Database

```
# connect to database hosted on db4free server
dbcon <-  dbConnect(RMySQL::MySQL(),
                    user = 'sql9628958',
                    password = '4aUqBzC9pE',
                    dbname = 'sql9628958',
                    host = 'sql9.freemysqlhosting.net',
                    port = 3306)
```

Create Database

Set Up

Temporally removes foreign key check so that tables can be reset.

```
SET FOREIGN_KEY_CHECKS = 0;

DROP TABLE IF EXISTS flights;

DROP TABLE IF EXISTS airports;

DROP TABLE IF EXISTS conditions;

DROP TABLE IF EXISTS strikes;

SET FOREIGN_KEY_CHECKS = 1;
```

A - Flights table

Creates tables with the necessary column restrictions

```
CREATE TABLE flights (
  fid INTEGER PRIMARY KEY,
  date DATE,
  origin INTEGER,
  airline TEXT ,
  aircraft TEXT ,
  altitude INTEGER CHECK (altitude > 0),
  heavy TINYINT);
```

B - Airports Table

```
CREATE TABLE airports (
  aid INT PRIMARY KEY ,
  airportState TEXT,
  airportCode TEXT DEFAULT NULL   /*can be left empty as mentioned in the question*/
);
```

C Connecting Flights and Airports using foreign key id.

```
ALTER TABLE flights ADD FOREIGN KEY (origin) REFERENCES airports(aid);
```

D - Creating table 'conditions'

```
CREATE TABLE conditions (
    cid INT AUTO_INCREMENT PRIMARY KEY,
    sky_condition VARCHAR(255),
    explanation TEXT
)
```

E - Creating table 'Strikes'

```
CREATE TABLE strikes (
  sid INTEGER PRIMARY KEY,
  fid INTEGER,
  numbirds INTEGER,
  impact TEXT,
  damage BOOLEAN,
  altitude INTEGER CHECK (altitude >= 0),
  conditions INTEGER REFERENCES conditions (cid),
  UNIQUE(fid)       /* taking only first instance of the strikes */
);
```

F - Adding foreign key fid to strikes

```
ALTER TABLE strikes ADD FOREIGN KEY (fid) REFERENCES flights(fid)
```

G - Evaluating the table

```
SELECT * FROM flights;
```

```
SELECT * FROM airports;
```

```
SELECT * FROM conditions;
```

```
SELECT * FROM strikes;
```

5 - Connecting the csv to the database

```
/*note the question says directly from the file. I asked professor about this*/
/*if is a problem simply change link to file name*/
```

```
bird_url <- "https://s3.us-east-2.amazonaws.com/artificium.us/datasets/BirdStrikesData-V2.csv"

bds.raw <- read.csv(file = bird_url, header = T,
                    stringsAsFactors = F)
```

6 - Populating the tables from the dataframe bds.raw

```
/*USE AS REFERENCE*/
/*http://artificium.us/lessons/06.r/l-6-301-sqlite-from-r/l-6-301.html#Querying_Data_Frames_with_sqldf*/
```

Airports

```
# airports table
states <- unique(bds.raw$origin)    # getting the unique states
n_states <- length(states)
aid <- 1:n_states

# making a data frame and then using dbwritetable to populate the table from the data.frame
airports <- data.frame(aid = aid,airportState= ifelse(is.na(states),"unknown",states),airportCode=NA)
dbWriteTable(dbcon, "airports", airports,row.names= FALSE, append= TRUE )
```

```
## [1] TRUE
```

Flights Table

```
# taking the unique identifies directly form the bds.raw rid column
fid <- bds.raw$rid
date <- as.Date(bds.raw$flight_date, format = '%m/%d/%Y')     # taking only the date

# foreign key which connects the airports and the flights table
origin <- airports$aid[match(bds.raw$origin, airports$airportState)]

# default value as unknown if there is no airline and aircraft
airline <- ifelse(is.na(bds.raw$airline), "Unknown", bds.raw$airline)
aircraft <- ifelse(is.na(bds.raw$aircraft), "Unknown", bds.raw$aircraft)
# altitude is a string with ',', therefore converted it to integer and remove the ','
altitude <- as.integer(gsub(",", '', bds.raw$altitude_ft))

# {TRUE:1, FALSE:0}
heavy <- ifelse(bds.raw$heavy_flag == 'Yes', 1, 0)  # mysql does not take boolean values

# Create the flights dataFrame
flights <- data.frame(fid, date, origin, airline, aircraft, altitude, heavy)

# Insert the values into the flights table
dbWriteTable(dbcon, 'flights', flights, row.names = FALSE, append = TRUE)
```

```
## [1] TRUE
```

Conditions table

```
# taking the unique conditions
cid <- 1: length(unique(bds.raw$sky_conditions))
sky_condition <- unique(bds.raw$sky_conditions)
explanation <- NA

# making a dataframe
conditions <- data.frame(cid = cid,
                         sky_condition = sky_condition,
                        explanation = explanation)

# populating the table with data Frame
dbWriteTable(dbcon, 'conditions', conditions, row.names=FALSE, append=TRUE)
```

```
## [1] TRUE
```

Strikes table

```
#row count variable
n_rows <- nrow(bds.raw)
# sid is synthetic key
sid <- 1:n_rows
# fid is the foreign key that connects strikes and flights
fid <- flights$fid
# directly taking from the data.frame
numbirds <- as.integer(bds.raw$wildlife_struck)
impact <- bds.raw$impact

# mysql doesnt take boolean values
damage <- ifelse(bds.raw$damage == 'Caused damage', 1, 0)

# string with ',', converted into integer without ','
altitude <- as.integer(gsub(",","", bds.raw$altitude_ft))
# this is the foreign key that connects the tables strikes and conditions
cd <- match(as.character(bds.raw$sky_condition), as.character(conditions$sky_condition))

# creating a dataframe
strikes <- data.frame(sid= sid,
                      fid = fid,
                      numbirds = numbirds,
                      impact = impact,
                      damage = damage,
                      altitude = altitude,
                      conditions= cd)

# populating the table with our data.frame
dbWriteTable(dbcon, 'strikes', strikes, row.names = FALSE, append=TRUE)
```

```
## [1] TRUE
```

7 Taking a look at our created tables.

Flights table

```
select * from flights limit 5;
```

Table 1: 5 records

fid	date	origin	airline	aircraft	altitude	heavy
1195	2002-11-13	3	MILITARY	Airplane	2000	0
3019	2002-10-10	11	MILITARY	Airplane	400	0
3500	2001-05-15	3	MILITARY	Airplane	1000	0
3504	2001-05-23	3	MILITARY	Airplane	1800	0
3597	2001-04-18	2	MILITARY	Airplane	200	0

airports table

```sql
select * from airports limit 5;
```

Table 2: 5 records

aid	airportState	airportCode
1	New York	NA
2	Texas	NA
3	Louisiana	NA
4	Washington	NA
5	Virginia	NA

Conditions table

```sql
select * from conditions limit 3;
```

Table 3: 3 records

cid	sky_condition	explanation
1	No Cloud	NA
2	Some Cloud	NA
3	Overcast	NA

Strikes table

```sql
select * from strikes limit 5;
```

Table 4: 5 records

sid	fid	numbirds	impact	damage	altitude	conditions
1	202152	859	Engine Shut Down	1	1500	1
2	208159	424	None	1	0	2
3	207601	261	None	0	50	1
4	215953	806	Precautionary Landing	0	50	2
5	219878	942	None	0	50	1

8 Greatest number of bird stirke incidents.

```sql
select COUNT(sid) AS strikCount, airportState
FROM strikes
JOIN flights ON strikes.fid = flights.fid
JOIN airports ON  flights.origin = airports.aid
GROUP BY airportState
ORDER BY strikCount DESC
limit 10;
```

Table 5: Displaying records 1 - 10

strikCount	airportState
2520	California
2453	Texas
2055	Florida
1319	New York
1008	Illinois
986	Pennsylvania
960	Missouri
812	Kentucky
778	Ohio
729	Hawaii

9 Above average number of birdstrike.

```
SELECT COUNT(sid) AS strikCount, airline
FROM strikes
JOIN flights ON strikes.fid = flights.fid
GROUP BY airline
HAVING COUNT(strikes.sid) > (
  SELECT AVG(strikeCount) AS avStrikeCount
  FROM (
    SELECT COUNT(strikes.sid) AS strikeCount
    FROM strikes
    JOIN flights ON strikes.fid = flights.fid
    GROUP BY flights.airline
  ) AS subquery
)
```

Table 6: Displaying records 1 - 10

strikCount	airline
129	
189	ABX AIR
95	AIR CANADA
229	AIR WISCONSIN AIRLINES
414	AIRTRAN AIRWAYS
304	ALASKA AIRLINES
107	ALLEGIANT AIR
135	ALOHA AIRLINES
157	AMERICA WEST AIRLINES
2058	AMERICAN AIRLINES

10 - Total number of bird strike by month.

```
p10 <- dbGetQuery(dbcon, "
SELECT SUM(numbirds) as birds,DATE_FORMAT(flights.date, '%m') AS month
```

```
FROM strikes
JOIN flights ON strikes.fid = flights.fid
GROUP BY month")
```

```
## Warning in .local(conn, statement, ...): Decimal MySQL column 0 imported as
## numeric
```

```
print(p10)
```

```
##    birds month
## 1    141  <NA>
## 2   3106    01
## 3   2602    02
## 4   3539    03
## 5   3802    04
## 6   4077    05
## 7   5209    06
## 8   9344    07
## 9  11013    08
## 10  9201    09
## 11  7277    10
## 12  6306    11
## 13  3173    12
```

11 - Bird strike by month - visualization.

ggplot(p10, aes(x = month, y = birds)) + ggtitle('Bird Strikes by month - Visualized') + geom_col()

12 Stored Procedure

```
/* drop the procedure if exists */
DROP PROCEDURE if exists newstrike;
```

```
# defining the procedure
sql_statement <- "
CREATE PROCEDURE newstrike(
  IN n_date DATE,
  IN n_origin VARCHAR(255),
  IN n_aircraft VARCHAR(255),
  IN n_altitude INT,
  IN n_numbirds INT,
  IN n_conditions VARCHAR(255)
)
BEGIN
  DECLARE v_fid INT;
  DECLARE v_aid INT;

  SELECT aid INTO v_aid FROM airports WHERE aid = n_origin;
  IF v_aid IS NULL THEN
```

```
    INSERT INTO airports (aid) VALUES (n_origin);
    SET v_aid = LAST_INSERT_ID();
  END IF;

  SELECT fid INTO v_fid FROM flights WHERE fid = n_aircraft;
  IF v_fid IS NULL THEN
    INSERT INTO flights (fid) VALUES (n_aircraft);
    SET v_fid = LAST_INSERT_ID();
  END IF;

  INSERT INTO strikes (fid, numbirds, impact, damage, altitude, conditions)
  VALUES (v_fid, n_numbirds, 'N/A', 0, n_altitude, n_conditions);

END;
"

# Execute the SQL statement
dbExecute(dbcon, statement = sql_statement)
```

```
## [1] 0
```

tests that procedure is working

```
CALL newstrike('2023-06-24', 3, 'Airplane', 1000, 12, 'No Cloud');
```

As we can see that the new strike is added to the strike table

```
select * from strikes limit 5;
```

Table 7: 5 records

sid	fid	numbirds	impact	damage	altitude	conditions
0	0	12	N/A	0	1000	0
1	202152	859	Engine Shut Down	1	1500	1
2	208159	424	None	1	0	2
3	207601	261	None	0	50	1
4	215953	806	Precautionary Landing	0	50	2

```
dbDisconnect(dbcon)
```

```
## [1] TRUE
```