17. Some more about Databases

Principles of Data Science with R

Dr. Uma Ravat

PSTAT 10

We did

- SQl queries involving multiple tables : joins

Next we will see...

- Other parts of SQL

SQL consists of 3 parts:

1. **Data Query Language** — DQL
   - search(query) one or more relations(tables)

2. **Data Manipulation Language** — DML
   - insert/update/delete tuples(rows) in relations(tables)

3. **Data Definition Language** — DDL
   - create/alter/delete relations(tables) and their attributes(fields, columns)

3. Data Definition Language (DDL): create/alter/delete relations(tables) and their attributes(fields, columns)

create a database - Tiny clothes

A small online clothing store called 'Tiny Clothes' needs you to create a database and run queries on it. Some relations are

CUSTOMER

CUST_NO	NAME	ADDRESS
C1	ALEX	State
C2	BOB	Hollister
C3	CAROL	Ocean
C6	JUAN	Phelps

SALES_ORDER

ORDER_NO	DATE	CUST_NO
01	11/11/17	C1
02	7/9/17	C3
09	8/16/17	C6
010	10/12/17	C6

PRODUCT

PROD_NO	NAME	COLOR
p1	PANTS	BLUE
p2	PANTS	KHAKI
p3	SOCKS	GREEN
p4	SOCKS	WHITE
p5	SHIRT	WHITE

SALES_ORDER_LINE

ORDER_NO	PROD_NO	QUANTITY
01	p1	10
02	p1	10
02	p4	20
09	p1	05
010	p1	05

create a database - Tiny clothes

1. Load required libraries for SQLite database connection
2. To **create a new RSQLite database**, you supply the filename to `dbConnect()`

```
library(DBI)
library(RSQLite)
my_tinyclothes_db <- dbConnect(RSQLite::SQLite(), "./data/my_tinyclothesdb.sqlite")
```

You can give your database any name that meets the R naming conventions.

```
dbListTables(my_tinyclothes_db)
```

```
## character(0)
```

3. Ensure the provided data is in the data directory. (csv files `EMPLOYEE`, `DEPARTMENT` and `CUSTOMER`).
4. Create data frames from each of the tables from the csv files

```
# read data as data frames
employee <- read.csv("./data/tinyclothes/EMPLOYEE.txt",
                     header = T, stringsAsFactors = F)
department <- read.csv("./data/tinyclothes/DEPARTMENT.txt",
                       header = T, stringsAsFactors = F)
customer <- read.csv("./data/tinyclothes/CUSTOMER.txt",
                     header = T, stringsAsFactors = F)
```

5. Write these data frames to your database.

```
dbWriteTable(my_tinyclothes_db,
             "EMPLOYEE", employee, overwrite = T)
dbWriteTable(my_tinyclothes_db,
             "DEPARTMENT", department, overwrite = T)
dbWriteTable(my_tinyclothes_db,
             "CUSTOMER", customer, overwrite = T)
```

6. check that the tables are included in database. How many records are in each table?

check which relations are included

```
dbListTables(my_tinyclothes_db)
```

```
## [1] "CUSTOMER"   "DEPARTMENT" "EMPLOYEE"
```

```
dbListFields( my_tinyclothes_db, "customer")
```

```
## [1] "CUST_NO" "NAME"    "ADDRESS"
```

```
dbGetQuery(my_tinyclothes_db,
           "select *  from customer")
```

```
##   CUST_NO  NAME   ADDRESS
## 1      C1  ALEX     State
## 2      C2   BOB Hollister
## 3      C3 CAROL     Ocean
## 4      C6  JUAN    Phelps
```

```
dbGetQuery(my_tinyclothes_db,
           "select count(*) as department from department")
```

```
##   department
## 1          3
```

```
dbGetQuery(my_tinyclothes_db,
           "select count(*) as employee from employee")
```

```
##   employee
## 1        4
```

2. Data Manipulation Language (DML): insert/update/delete tuples(rows) in relations(tables)

insert

```
dbExecute(my_tinyclothes_db,
    "insert into
    customer(CUST_NO, NAME , ADDRESS)
    values('C7', 'Luis', 'Storke')")
```

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

```
dbGetQuery(my_tinyclothes_db,
           "select *  from customer")
```

```
##   CUST_NO  NAME   ADDRESS
## 1      C1  ALEX     State
## 2      C2   BOB Hollister
## 3      C3 CAROL     Ocean
## 4      C6  JUAN    Phelps
## 5      C7  Luis    Storke
```

delete

```
dbGetQuery(my_tinyclothes_db,
           "select *  from customer where Name like 'l%'")
```

```
##   CUST_NO NAME ADDRESS
## 1      C7 Luis  Storke
```

```
dbExecute(my_tinyclothes_db,"DELETE FROM CUSTOMER
WHERE Name like 'l%'")
```

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

```
dbGetQuery(my_tinyclothes_db,
           "select *  from customer")
```

```
##   CUST_NO  NAME   ADDRESS
## 1      C1  ALEX     State
## 2      C2   BOB Hollister
## 3      C3 CAROL     Ocean
## 4      C6  JUAN    Phelps
```

3. Data Definition Language (DDL): create/alter/delete relations(tables) and their attributes(fields, columns)

Create new table

Suppose 'Tiny Clothes' wants to expand the business and start selling soft toys.

Soft Toys will be supplied to Tiny Clothes by several suppliers.

New relations will need to be added to the existing database.

We will create:

1. a relation showing the name color and price of the toys.
    - **schema:** SOFT_TOYS(Toy_ID, name, color, price) with Toy_ID primary key
    - equivalently **schema:** SOFT_TOYS(Toy_ID , name, color, price)
    - equivalently **schema:** SOFT_TOYS(**Toy_ID** , name, color, price)
2. A relation giving the details of the suppliers.
    - schema: TOY_SUPPLIER (Supplier_ID, Supplier_name, Toy_ID) with Toy_ID as a foreign key

Create new table

```
dbListTables(my_tinyclothes_db)
```

```
## [1] "CUSTOMER"   "DEPARTMENT" "EMPLOYEE"
```

```
dbSendQuery(my_tinyclothes_db,
            'CREATE TABLE SOFT_TOYS
            (TOY_ID TEXT NOT NULL PRIMARY KEY,
            NAME TEXT,
            COLOR TEXT,
            PRICE TEXT)')
```

```
## <SQLiteResult>
##   SQL  CREATE TABLE SOFT_TOYS
##             (TOY_ID TEXT NOT NULL PRIMARY KEY,
##             NAME TEXT,
##             COLOR TEXT,
##             PRICE TEXT)
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
dbSendQuery(my_tinyclothes_db,
            'CREATE TABLE TOY_SUPPLIER
            (SUPPLIER_ID TEXT NOT NULL PRIMARY KEY,
            SUPPLIER_NAME TEXT,
            TOY_ID TEXT,
            FOREIGN KEY(TOY_ID) REFERENCES SOFT_TOYS(TOY_ID)
            )')
```

```
## Warning: Closing open result set, pending rows
```

```
## <SQLiteResult>
##   SQL  CREATE TABLE TOY_SUPPLIER
##             (SUPPLIER_ID TEXT NOT NULL PRIMARY KEY,
##             SUPPLIER_NAME TEXT,
##             TOY_ID TEXT,
##             FOREIGN KEY(TOY_ID) REFERENCES SOFT_TOYS(TOY_ID)
##             )
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
dbListTables(my_tinyclothes_db)
```

```
## Warning: Closing open result set, pending rows
```

```
## [1] "CUSTOMER" "DEPARTMENT" "EMPLOYEE" "SOFT_TOYS" "TOY_SUPPLIER"
```

```
dbGetQuery(my_tinyclothes_db, "PRAGMA table_info('SOFT_TOYS')")
```

```
##   cid   name type notnull dflt_value pk
## 1   0 TOY_ID TEXT       1         NA  1
## 2   1   NAME TEXT       0         NA  0
## 3   2  COLOR TEXT       0         NA  0
## 4   3  PRICE TEXT       0         NA  0
```

```
dbGetQuery(my_tinyclothes_db, "PRAGMA table_info('TOY_SUPPLIER')")
```

```
##   cid          name type notnull dflt_value pk
## 1   0   SUPPLIER_ID TEXT       1         NA  1
## 2   1 SUPPLIER_NAME TEXT       0         NA  0
## 3   2        TOY_ID TEXT       0         NA  0
```

```
dbGetQuery(my_tinyclothes_db, "PRAGMA foreign_key_list(SOFT_TOYS)")
```

```
## [1] id        seq       table     from      to        on_update on_delete
## [8] match
## <0 rows> (or 0-length row.names)
```

```
dbGetQuery(my_tinyclothes_db, "PRAGMA foreign_key_list(TOY_SUPPLIER)")
```

```
##   id seq     table   from     to on_update on_delete match
## 1  0   0 SOFT_TOYS TOY_ID TOY_ID NO ACTION NO ACTION  NONE
```

Entity integrity

```
dbExecute(my_tinyclothes_db,
"INSERT INTO SOFT_TOYS
VALUES
(1, 'Luis', 'green',  19.99)")
```

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

```
dbExecute(my_tinyclothes_db,
"INSERT INTO SOFT_TOYS
VALUES
(1, 'Lucy', 'red',  9.99)")
```

```
## Error: UNIQUE constraint failed: SOFT_TOYS.TOY_ID
```

Enforcing foreign keys

```
# Required for foreign-key support otherwise foreign keys are not enforced
dbExecute(my_tinyclothes_db, "pragma foreign_keys = on")
```

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

```
dbExecute(my_tinyclothes_db,
"INSERT INTO TOY_SUPPLIER
VALUES
(1, 'Angelina', 1)")
```

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

```
dbExecute(my_tinyclothes_db,
"INSERT INTO TOY_SUPPLIER
VALUES
(2, 'Angela', 1)")
```

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

```
dbGetQuery(my_tinyclothes_db, "Select * from TOY_SUPPLIER")
```

```
##   SUPPLIER_ID SUPPLIER_NAME TOY_ID
## 1           1      Angelina      1
## 2           2        Angela      1
```

Referential integrity

```
dbGetQuery(my_tinyclothes_db, "Select * from SOFT_TOYS")
```

```
##   TOY_ID NAME COLOR PRICE
## 1      1 Luis green 19.99
```

```
dbExecute(my_tinyclothes_db,
"INSERT INTO TOY_SUPPLIER
VALUES
(3, 'Angel', 2)")
```

```
## Error: FOREIGN KEY constraint failed
```

```
dbExecute(my_tinyclothes_db,
"INSERT INTO TOY_SUPPLIER
VALUES
(3, 'Angel', NULL)")
```

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

delete table

```
dbSendQuery(my_tinyclothes_db,
            'DROP TABLE TOY_SUPPLIER')
```

```
## <SQLiteResult>
##   SQL  DROP TABLE TOY_SUPPLIER
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
dbSendQuery(my_tinyclothes_db,
            'DROP TABLE SOFT_TOYS')
```

```
## Warning: Closing open result set, pending rows
```

```
## <SQLiteResult>
##   SQL  DROP TABLE SOFT_TOYS
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

Try switching order of droping tables - error because of foreign keys.

```
dbListTables(my_tinyclothes_db)
```

```
## Warning: Closing open result set, pending rows
```

```
## [1] "CUSTOMER"   "DEPARTMENT" "EMPLOYEE"
```

```
dbDisconnect(my_tinyclothes_db)
```

A little detour from databases

Course Evaluations

- **WHY DO THEM?** Your input will help me develop this course.
- **HOW to do them?**
  - Describe what you learned and what helped you learn.
  - Use examples and mention specific aspects of the course and instruction that was beneficial to you
  - Be constructive: tell me what worked and avoid inappropriate personal comments.
  - See next couple slides for beneficial things you could mention as you complete the course evaluation.

Course resources: Which did you use most and how?

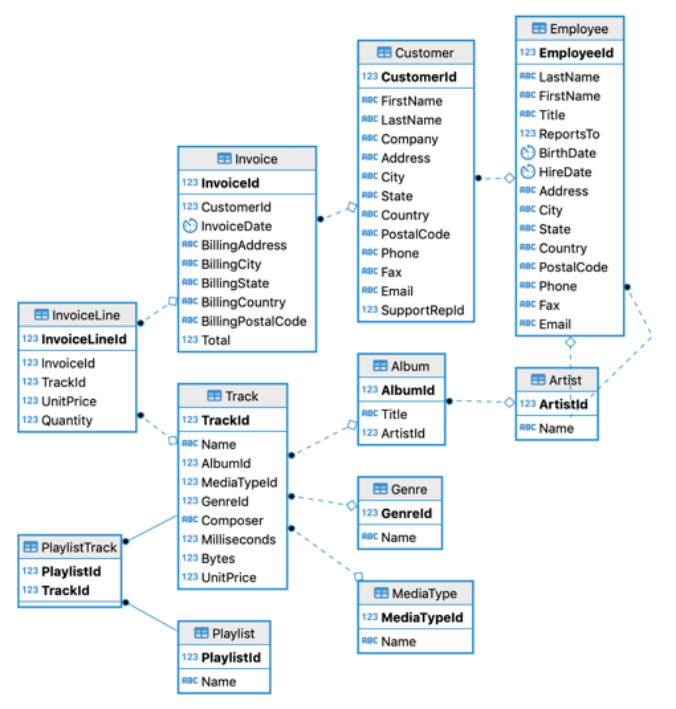
- **Detailed lecture Slides/course notes**
  - All code shown to you in slides - did you refer to it? when?
    - Did you skip class and detailed slides helped you follow along
    - Did you stop coming to class because slides have all code to complete homework
    - Would points for lecture attendance change whether or not you came?
  - Lecture Your Turns - did this help you practice, understand material during lecture/after lecture?
  - Topics presented as learnr Tutorials - did these help you actively learn by doing in lecture?
  - Do you think less detailed slides would help you focus better?
- **Questions** during and after lecture

- **Practice** with course concepts
  - worksheets corresponding to each lecture
  - homework each week
  - quiz every other week
- Was this organization - lecture with YT, worksheet, homework, quiz helpful in getting practice with course concepts?
- Extra opportunities/industry insights

- PSTAT server
  - Was it easy, useful to get started with the course on the server on Day 1.
  - Were the silent videos helpful in getting used to working on the server?
  - Was it easy to have all course material on the server? before lecture?
- Did Rmarkdown in lecture, Rmarkdown template for worksheets and homework help you gain mastery in an industry skill for data scientists?

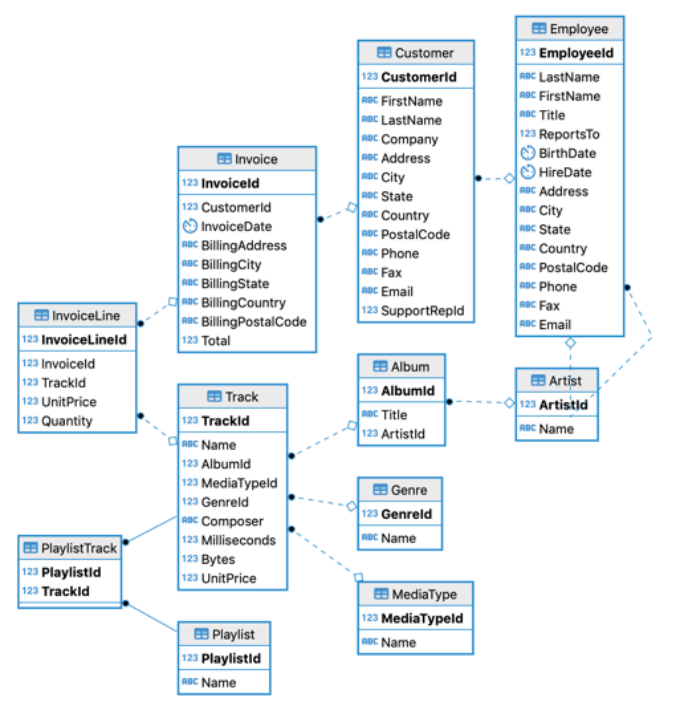
- **Organization**
  - Over 20 office hours - were these helpful to be spread out at different times, did you use them?
  - Extra review for exam
  - Extra office hours at exam time.
  - Canvas, Server organization - Did you find things easily?
  - Some flexibility regarding assignments but limits to make sure you don't fall behind in course work
    - two no questions asked extensions,
    - flexibility with timing for weekly quiz - were these helpful?
    - fair to all students

Database Design



- Tables should represent distinct real-world concepts.
- Records should be uniquely identified with a primary key.
- Relationships between tables represented by primary key/foreign key relations.

Database Design



Straightforward relationships

- Customer/Employee
  - An employee can serve multiple customers
  - A customer has at most one employee support rep
- Track/Album
  - An album can contain multiple tracks
  - A track can be on at most one album.

Summary:

1. **Data Query Language** — DQL
   - search(query) one or more relations(tables)
2. **Data Manipulation Language** — DML
   - insert/update/delete tuples(rows) in relations(tables)
3. **Data Definition Language** — DDL
   - create/alter/delete relations(tables) and their attributes(fields, columns)
4. Database design