16. Some more about Databases

Principles of Data Science with R

Dr. Uma Ravat

PSTAT 10

Announcement

1. Quiz will open tomorrow(Friday) at 9am and will close at 9pm
   - **No collaboration**
   - **No extensions**
   - Read questions carefully
   - Good Luck!!
2. Next week: Lecture, worksheet due Wednesday - check server on Tuesday for video/activity description
3. Make time for a short break - you'll need it for the finals week!
4. Don't forget Quiz redo on 11/29 during classtime - to boost your grade, make up for a missing quiz and review for the final exam.

4. **Collaborative Review** Sign up by clicking here
    - Choose a topic and prepare a review sheet for that topic by Monday 11/28
    - Starting Tuesday 11/29, you can comment on and add to review sheets you **didn't put together** while reviewing for the final
    - During finals week, we'll release review sheet put together by our ULA team for you to check if you missed anything or just to get a fresh perspective while reviewing for the final.

We did

- SQl queries involving multiple tables : joins

SQL

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)

SQL handles the roles of the DDL, DML, and DQL languages, all-in-one

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.

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
```

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
```

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
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

```
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)')
```

```
## 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)
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```r
dbListTables(my_tinyclothes_db)
```

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

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

```r
dbSendQuery(my_tinyclothes_db,'PRAGMA foreign_keys = ON')
```

```
## <SQLiteResult>
##   SQL  PRAGMA foreign_keys = ON
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

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

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

```
##   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
```

delete table

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

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

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

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

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

```
dbListTables(my_tinyclothes_db)
```

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

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

A little detour from databases

Course Evaluations

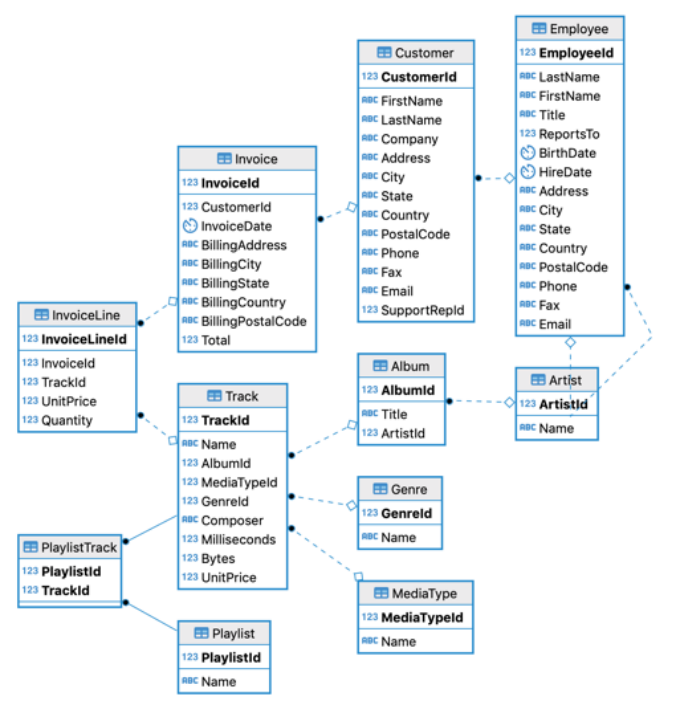
- **WHY DO THEM?** Your input will help me develop this course.
- **HOW to do them?**
  - Describe what you learned and what helped you learn.
  - Use examples and mention specific aspects of the course and instruction.
  - Be constructive: tell me what worked and avoid inappropriate personal comments.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

Course resources: Which did you use most and how?

- **Detailed lecture Slides/course notes**
  - All code - did you refer to it? when?
  - Lecture Your Turns - did this help you practice, understand material during lecture?
  - Topics presented as learnr Tutorials - were these helpful during lecture, afterwards? how?
  - Do you think less detailed slides would help you focus better?
- **Practice** with course concepts in Rmarkdown - were these easy for you to fill? were they helpful as you studied for hw, quiz, exam?
  - Rmarkdown worksheets corresponding to each lecture
  - Rmarkdown template for each homework
- Multiple choice questions on HW to prepare for the quiz
- Quiz each week on HW you just turned in
- Extra practice for exam
- Collaborative review

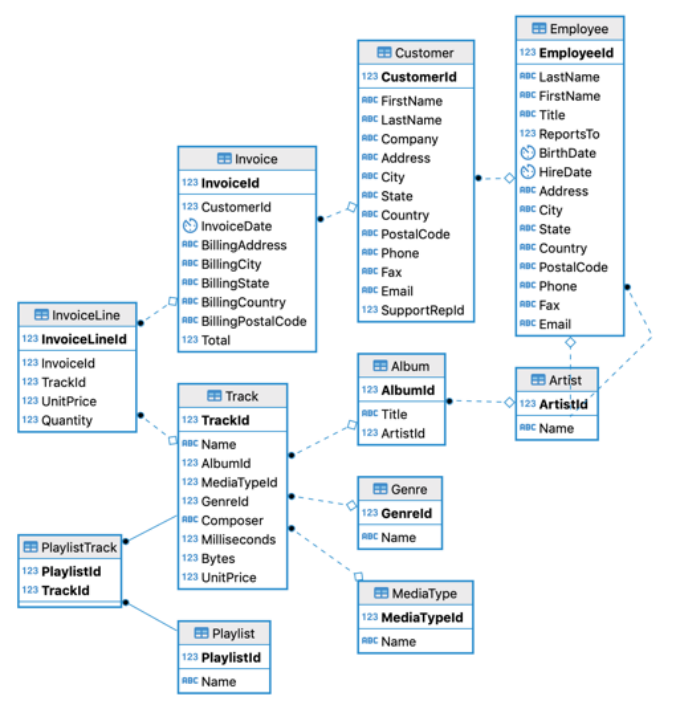
- **Organization**
  - Over 20 office hours and discussion forum - were these helpful to be spread out at different times?
  - Extra office hours at exam time.
  - Homework clinics
  - Canvas - was it organized well so you found things easily?
  - Some flexibility regarding assignments but limits to make sure you don't fall behind in course work
    - two no questions asked extensions, extensions this week, flexibility with timing for weekly quiz - were these helpful?

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.

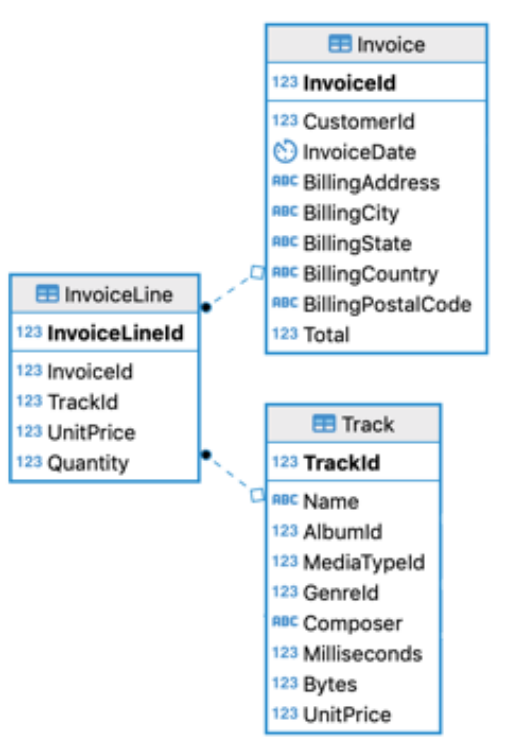
Database Design



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.

Association Table



- Multiple tracks can be purchased in one transaction(An invoice can contain multiple tracks)
- multiple people buying the same track(A track can be in multiple invoices)
- To represent this *many-to-many* relationship, we need the *association table* InvoiceLine.

Association Table

```
dbGetQuery(chinook_db,
"select InvoiceLineId, InvoiceId, TrackId
from InvoiceLine
where trackid = 8 or invoiceid = 2")
```

```
##   InvoiceLineId InvoiceId TrackId
## 1             4         2       8
## 2          1155       214       8
## 3             3         2       6
## 4             5         2      10
## 5             6         2      12
```

InvoiceLine has foreign keys to Invoice and Track tables.

To associate Track and Invoice, join on the association table.

Association Table

```
dbGetQuery(chinook_db,
"select t.Name as TrackName, InvoiceLineId,
             i.InvoiceId, t.TrackId
from Invoice i
     inner join InvoiceLine il
     on i.InvoiceId = il.InvoiceId
    inner join Track t
    on il.TrackId = t.TrackId
where i.invoiceid = 2")
```

```
##               TrackName InvoiceLineId InvoiceId TrackId
## 1 Put The Finger On You             3         2       6
## 2      Inject The Venom             4         2       8
## 3            Evil Walks             5         2      10
## 4    Breaking The Rules             6         2      12
```

```
dbGetQuery(chinook_db,
"select t.Name as TrackName, InvoiceLineId,
            i.InvoiceId, t.TrackId
from Invoice i
      inner join InvoiceLine il
      on i.InvoiceId = il.InvoiceId
      inner join Track t
      on il.TrackId = t.TrackId
where t.trackid = 8")
```

```
##          TrackName InvoiceLineId InvoiceId TrackId
## 1 Inject The Venom             4         2       8
## 2 Inject The Venom          1155       214       8
```

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