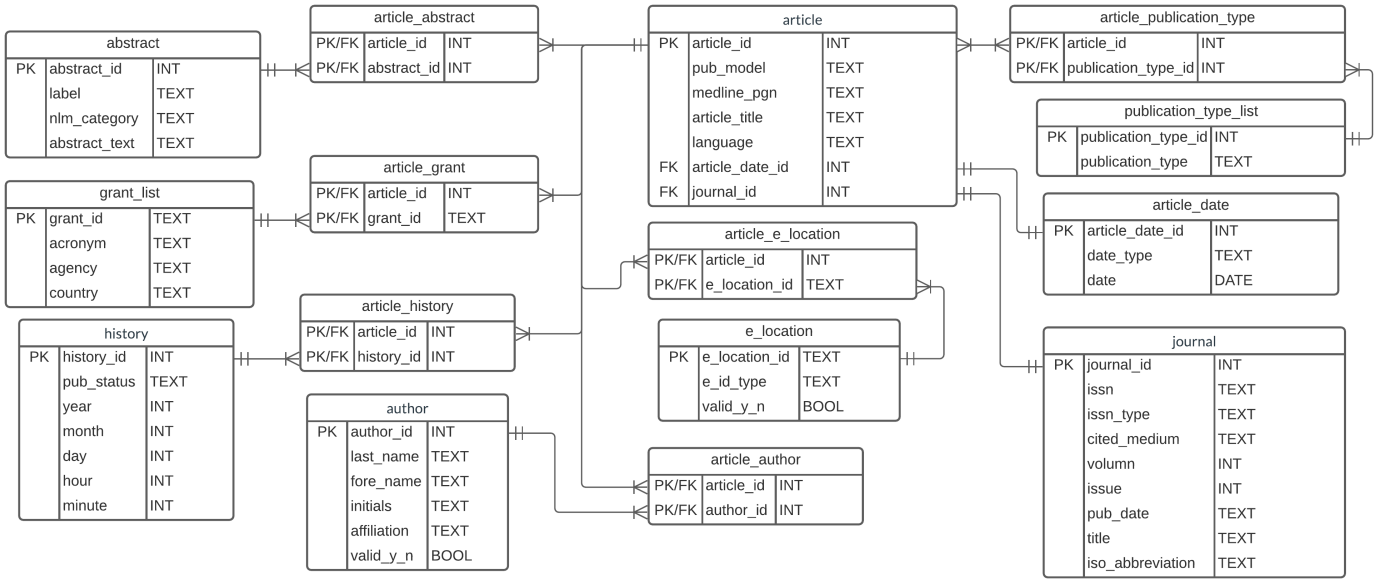
R Notebook

Author: Zihao Qiu

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

Part 1 (40 pts) Load XML

1. (5 pts) Create a normalized relational schema that contains minimally the following entities: Article, Journal, Author, History. Use the XML document to determine the appropriate attributes (fields/columns) for the entities (tables). While there may be other types of publications in the XML, you only need to deal with articles in journals. Create appropriate primary and foreign keys. Where necessary, add surrogate keys. Include an image of an ERD showing your model in your R Notebook.



2. (5 pts) Realize the relational schema in SQLite (place the CREATE TABLE statements into SQL chunks in your R Notebook).

```
library(RSQLite)
fpath = "C:/Users/zqiu9/Documents/R Projects/"
dbfile = "Practicum.II.sqlite"
dbcon <- dbConnect(RSQLite::SQLite(), paste0(fpath, dbfile))
```

create tables in SQLite database

```
CREATE TABLE abstract(
  abstract_id INT,
  label TEXT,
  nlm_category TEXT,
  abstract_text TEXT,
  PRIMARY KEY (abstract_id)
);
```

```
CREATE TABLE grand_list(
  grand_id INT,
  acronym TEXT,
  agency TEXT,
  country TEXT,
  PRIMARY KEY (grand_id)
);
```

```
CREATE TABLE history(
  history_id INT,
  pub_status TEXT,
  year INT,
  month INT,
  day INT,
  hour INT,
  minute INT,
  PRIMARY KEY (history_id)
);
```

```
CREATE TABLE author(
  author_id INT,
  last_name TEXT,
  fore_name TEXT,
  initials TEXT,
  affiliation TEXT,
  valid_y_n BOOLEAN,
  PRIMARY KEY (author_id)
);
```

```
CREATE TABLE e_location(
  e_location_id TEXT,
  e_id_type TEXT,
  valid_y_n BOOLEAN,
  PRIMARY KEY (e_location_id)
)
```

```
CREATE TABLE publication_type_list(
  publication_type_id INT,
  publication_type TEXT,
  PRIMARY KEY (publication_type_id)
);
```

```
CREATE TABLE article_date(
  article_date_id INT,
  date DATE,
  date_type TEXT,
  PRIMARY KEY (article_date_id)
);
```

```
CREATE TABLE journal(
  journal_id INT,
  issn TEXT,
  issn_type TEXT,
  cited_medium TEXT,
  volumn INT,
  issue INT,
  pub_date TEXT,
  title TEXT,
  iso_abbreviation TEXT,
  PRIMARY KEY (journal_id)
);
```

```
CREATE TABLE article(
  article_id INT,
  pub_model TEXT,
  medline_png TEXT,
  article_title TEXT,
  language TEXT,
  article_date_id INT,
  journal_id INT,
  PRIMARY KEY (article_id),
  FOREIGN KEY (journal_id) REFERENCES journal(journal_id),
  FOREIGN KEY (article_date_id) REFERENCES article_date(article_date_id)
);
```

```
CREATE TABLE article_abstract(
  article_id INT,
  abstract_id INT,
  PRiMARY KEY (article_id, abstract_id),
  FOREIGN KEY (article_id) REFERENCES article(article_id),
  FOREIGN KEY (abstract_id) REFERENCES abstract(abstract_id)
)
```

```
CREATE TABLE article_grant(
  article_id INT,
  grant_id INT,
  PRiMARY KEY (article_id, grant_id),
  FOREIGN KEY (article_id) REFERENCES article(article_id),
  FOREIGN KEY (grant_id) REFERENCES grant_list(grant_id)
)
```

```
CREATE TABLE article_history(
  article_id INT,
  history_id INT,
  PRiMARY KEY (article_id, history_id),
  FOREIGN KEY (article_id) REFERENCES article(article_id),
  FOREIGN KEY (history_id) REFERENCES history(history_id)
)
```

```
CREATE TABLE article_e_location(
  article_id INT,
  e_location_id INT,
  PRiMARY KEY (article_id, e_location_id),
  FOREIGN KEY (article_id) REFERENCES article(article_id),
  FOREIGN KEY (e_location_id) REFERENCES e_location(e_location_id)
)
```

```
CREATE TABLE article_author(
  article_id INT,
  author_id INT,
  PRiMARY KEY (article_id, author_id),
  FOREIGN KEY (article_id) REFERENCES article(article_id),
  FOREIGN KEY (author_id) REFERENCES author(author_id)
)
```

```
CREATE TABLE article_publication_type(
  article_id INT,
  publication_type_id INT,
  PRiMARY KEY (article_id, publication_type_id),
  FOREIGN KEY (article_id) REFERENCES article(article_id),
  FOREIGN KEY (publication_type_id) REFERENCES publication_type_list(publication_type_id)
)
```

3. (30 pts) Extract and transform the data from the XML and then load into the appropriate tables in the database. You cannot (directly and solely) use xmlToDataFrame but instead must parse the XML node by node using a combination of node-by-node tree traversal and XPath. It is not feasible to use XPath to extract all journals, then all authors, etc. as some are missing and won't match up. You will need to iterate through the top-level nodes. While outside the scope of the course, this task could also be done through XSLT.

load the xml file into R

```
library(XML)
path <- "C:/Users/zqiu9/Documents/R Projects/"
xmlFile <- "pubmed_sample.xml"
fp <- paste0(path,xmlFile)
xmlObj <- xmlParse(fp)
```

extract and transform journal table

```
issn <- xpathSApply(xmlObj, "//ISSN ", xmlValue)
journal <- data.frame(issn)
issn_type <- xpathSApply(xmlObj, "//ISSN/@IssnType")
journal$issn_type <- data.frame(issn_type)$issn_type
cited_medium <- xpathSApply(xmlObj, "//JournalIssue/@CitedMedium")
journal$cited_medium <- data.frame(cited_medium)$cited_medium
volumn <- xpathSApply(xmlObj, "//Volume", xmlValue)
journal$volumn <- data.frame(volumn)$volumn
issue <- xpathSApply(xmlObj, "//Issue", xmlValue)
journal$issue <- data.frame(issue)$issue
pub_date <- xpathSApply(xmlObj, "//PubDate", xmlValue)
journal$pub_date <- data.frame(pub_date)$pub_date
title <- xpathSApply(xmlObj, "//Title", xmlValue)
journal$title <- data.frame(title)$title
iso_abbreviation <- xpathSApply(xmlObj, "//ISOAbbreviation", xmlValue)
journal$iso_abbreviation <- data.frame(iso_abbreviation)$iso_abbreviation
# assgin id
n = nrow(journal)
journal <- data.frame(1, journal)
journal[,1] <- seq(1, n)
# rename
names(journal)[1] <- "journal_id"
```

extract and transform article_date table

```
article_date <- xmlToDataFrame(nodes = getNodeSet(xmlObj, "//ArticleDate"))
date_type <- xpathSApply(xmlObj, "//ArticleDate/@DateType")
article_date$date_type <- data.frame(date_type)$date_type
# assign id
n = nrow(article_date)
article_date <- data.frame(1, article_date)
article_date[,1] <- seq(1, n)
# format date attribute
article_date$date <- paste0(article_date$Year, '-', article_date$Month, '-', article_date$Day)
# truncate attributes that no longer needed
article_date <- article_date[ -c(2:4) ]
# rename
names(article_date)[1] <- "article_date_id"
names(article_date)[2] <- "date_type"
```

extract foreign keys for article date

```
adfk = c()
pub_id = 1
root <- xmlRoot(xmlObj)
num <- xmlSize(root)
for (i in 1:num){
  pa <- root[[i]]
  ad <- xmlToDataFrame(nodes = getNodeSet(pa, ".//ArticleDate"))
  # if article date not empty
  if (dim(ad)[1] == 1){
    adfk <- c(adfk, pub_id)
    pub_id <- pub_id + 1
  }
  # else empty
  else{
    adfk <- c(adfk, NA)
  }
}
```

extract and transform article table

```
pub_model = xpathSApply(xmlObj, "//Article/@PubModel")
article <- data.frame(pub_model)
medline_png <- xpathSApply(xmlObj, "//MedlinePgn", xmlValue)
article$medline_png <- data.frame(medline_png)$medline_png
article_title <- xpathSApply(xmlObj, "//ArticleTitle", xmlValue)
article$article_title <- data.frame(article_title)$article_title
language <- xpathSApply(xmlObj, "//Language", xmlValue)
article$language <- data.frame(language)$language
# assign id
n = nrow(article)
article <- data.frame(1, article)
article[,1] <- seq(1, n)
names(article)[1] <- "article_id"
article$journal_id <- journal$journal_id
# add article_date foreign key
article <- data.frame(article, adfk)
# rename
names(article)[7] <- "article_date_id"
```

extract and transform publication_type_list table

```
publication_type_list <- xmlToDataFrame(nodes = getNodeSet(xmlObj, "//PublicationType"))
publication_type_list <- unique(publication_type_list)
n = nrow(publication_type_list)
publication_type_list <- data.frame(1, publication_type_list)
publication_type_list[,1] <- seq(1, n)
names(publication_type_list)[1] <- "publication_type_id"
names(publication_type_list)[2] <- "publication_type"
```

extract and transform history table

```
history <- xmlToDataFrame(nodes = getNodeSet(xmlObj, "//PubMedPubDate"))
PubStatus <- xpathSApply(xmlObj, "//PubMedPubDate/@PubStatus")
history$PubStatus <- data.frame(PubStatus)$PubStatus
# assign id
history <- unique(history)
n = nrow(history)
history <- data.frame(1, history)
history[,1] <- seq(1, n)
# rename
names(history)[names(history) == "X1"] <- "history_id"
names(history)[names(history) == "Year"] <- "year"
names(history)[names(history) == "Month"] <- "month"
names(history)[names(history) == "Day"] <- "day"
names(history)[names(history) == "Hour"] <- "hour"
names(history)[names(history) == "Minute"] <- "minute"
names(history)[names(history) == "PubStatus"] <- "pub_status"
```

extract and transform abstract table

```
Abstract <- xpathSApply(xmlObj, "//AbstractText")
AbstractText <- xpathSApply(xmlObj, "//AbstractText", xmlValue)
abstract <- data.frame(AbstractText)

abstract$label = ""
abstract$nlm_category = ""
abstract$abstract_id = ""

for (n in 1:length(Abstract)){
  # if label attribute exists
  if (length(xmlAttrs(Abstract[[n]])[1]) > 0){
  abstract$label[n] <- xmlAttrs(Abstract[[n]])[1]
  }
  # if nlm_category attribute exists
  if (length(xmlAttrs(Abstract[[n]])[2]) > 0){
  abstract$nlm_category[n] <- xmlAttrs(Abstract[[n]])[2]
  }
  abstract$abstract_id[n] <- n
}
# rename
names(abstract)[1] <- "abstract_text"
```

extract and transform grand_list table

```
grant_list <- xmlToDataFrame(nodes = getNodeSet(xmlObj, "//Grant"))

names(grant_list)[names(grant_list) == "GrantID"] <- "grant_id"
names(grant_list)[names(grant_list) == "Acronym"] <- "acronym"
names(grant_list)[names(grant_list) == "Agency"] <- "agency"
names(grant_list)[names(grant_list) == "Country"] <- "country"
```

extract and transform author table

```
author <- xmlToDataFrame(nodes = getNodeSet(xmlObj, "//Author"))
xpathEx <- "//Author/@ValidYN"
ValidYN <- xpathSApply(xmlObj, xpathEx)
valid.df <- data.frame(ValidYN)
author$ValidYN <- valid.df$ValidYN
author <- unique(author)
n = nrow(author)
author <- data.frame(1, author)
author[,1] <- seq(1, n)
# rename
names(author)[names(author) == "X1"] <- "author_id"
names(author)[names(author) == "LastName"] <- "last_name"
names(author)[names(author) == "ForeName"] <- "fore_name"
names(author)[names(author) == "Initials"] <- "initials"
names(author)[names(author) == "Affiliation"] <- "affiliation"
names(author)[names(author) == "ValidYN"] <- "valid_y_n"
# replace NA with empty string
author[is.na(author)] = ''
```

extract and transform e_location table

```
e_location_id <- xpathSApply(xmlObj, "//ELocationID", xmlValue)
e_location <- data.frame(e_location_id)
e_id_type <- xpathSApply(xmlObj, "//ELocationID/@EIdType")
e_location$e_id_type <- data.frame(e_id_type)$e_id_type
valid_y_n <- xpathSApply(xmlObj, "//ELocationID/@ValidYN")
e_location$valid_y_n <- data.frame(valid_y_n)$valid_y_n
```

extract and transform article_grant, article_abstract, article_history, and article_e_location table

```
root <- xmlRoot(xmlObj)
num <- xmlSize(root)

article_grant <- c()
article_abstract <- c()
article_history <- c()
article_e_location <- c()


for (i in 1:num){
  pa <- root[[i]]
  grant_count <- xpathSApply(pa, "count(.//Grant)", xmlValue)
  abstract_count <- xpathSApply(pa, "count(.//AbstractText)", xmlValue)
  history_count <- xpathSApply(pa, "count(.//PubMedPubDate)", xmlValue)
  e_location_count <- xpathSApply(pa, "count(.//ELocationID)", xmlValue)
  # count to match composite key pairs
  if (grant_count > 0){
    for (j in 1: grant_count){
      article_grant <- c(article_grant, i)
    }
  }
  if (abstract_count > 0){
    for (j in 1: abstract_count){
      article_abstract <- c(article_abstract, i)
    }
  }
  if (history_count > 0){
    for (j in 1: history_count){
      article_history <- c(article_history, i)
    }
  }
  if (e_location_count > 0){
    for (j in 1: e_location_count){
      article_e_location <- c(article_e_location, i)
    }
  }
}
# assign ids an rename
article_grant = data.frame(article_grant)
n = nrow(article_grant)
article_grant <- data.frame(1, article_grant)
article_grant[,1] <- seq(1, n)
names(article_grant)[1] <- "grant_id"
names(article_grant)[2] <- "article_id"

article_abstract = data.frame(article_abstract)
n = nrow(article_abstract)
article_abstract <- data.frame(1, article_abstract)
article_abstract[,1] <- seq(1, n)
names(article_abstract)[1] <- "abstract_id"
names(article_abstract)[2] <- "article_id"

article_history = data.frame(article_history)
```

```
n = nrow(article_history)
article_history <- data.frame(1, article_history)
article_history[,1] <- seq(1, n)
names(article_history)[1] <- "history_id"
names(article_history)[2] <- "article_id"

article_e_location = data.frame(article_e_location)
n = nrow(article_e_location)
article_e_location <- data.frame(1, article_e_location)
article_e_location[,1] <- seq(1, n)
names(article_e_location)[1] <- "e_location_id"
names(article_e_location)[2] <- "article_id"
```

extract and transform article_author table

```
article_id <- c()
last_name <- c()
fore_name <- c()
affiliation <- c()

for (i in 1:num){
  pa <- root[[i]]
  ln <- xpathSApply(pa, ".//Author/LastName", xmlValue)
  fn <- xpathSApply(pa, ".//Author/ForeName", xmlValue)
  a <- xmlToDataFrame(nodes = getNodeSet(pa, ".//Author"))
  # count to link the relation between author and article
  count <- xpathSApply(pa, "count(.//Author)")
  for (j in 1:count){
    article_id <- c(article_id, i)
  }
  last_name <- c(last_name, ln)
  fore_name <- c(fore_name, fn)
  affiliation <- c(affiliation, a$Affiliation)
}
article_author <- data.frame(article_id)
article_author <- data.frame(article_author, last_name)
article_author <- data.frame(article_author, fore_name)
article_author <- data.frame(article_author, affiliation)
# replace NA with empty string
article_author[is.na(article_author)] = ''
# assign author_id foreign key
for (i in 1:nrow(article_author)){
  au <- author$author_id[which(author$fore_name == article_author$fore_name[i] & author$last_nam
e == article_author$last_name[i] & author$affiliation == article_author$affiliation[i])]
  article_author$author_id[i] <- au
}
# truncate attributes that no longer needed
article_author <- article_author[ -c(2:4) ]
```

extract and transform article_publication_type table

```
article_id <- c()
publication_type <- c()
for (i in 1:num){
  pa <- root[[i]]
  a <- xmlToDataFrame(nodes = getNodeSet(pa, ".//PublicationType"))
  count = nrow(a)
  # count to link the relation
  for (j in 1:count){
    article_id <- c(article_id, i)
  }
  publication_type <- c(publication_type, a$text)
}
article_publication_type = data.frame(publication_type, article_id)
# assign foreign key
for (i in 1:nrow(article_publication_type)){
  pt <- publication_type_list$publication_type_id[which(publication_type_list$publication_type =
= article_publication_type$publication_type[i])]

  article_publication_type$publication_type_id[i] <- pt
}
# truncate attributes that no longer needed
article_publication_type <- article_publication_type[ -c(1) ]
```

populate tables

```
dbWriteTable(dbcon, "abstract", abstract, append = T, row.names = FALSE)
dbWriteTable(dbcon, "grant_list", grant_list, append = T, row.names = FALSE)
dbWriteTable(dbcon, "history", history, append = T, row.names = FALSE)
dbWriteTable(dbcon, "e_location", e_location, append = T, row.names = FALSE)
dbWriteTable(dbcon, "author", author, append = T, row.names = FALSE)
dbWriteTable(dbcon, "publication_type_list", publication_type_list, append = T, row.names = FALS
E)
dbWriteTable(dbcon, "article_date", article_date, append = T, row.names = FALSE)
dbWriteTable(dbcon, "journal", journal, append = T, row.names = FALSE)
dbWriteTable(dbcon, "article", article, append = T, row.names = FALSE)
dbWriteTable(dbcon, "article_abstract", article_abstract, append = T, row.names = FALSE)
dbWriteTable(dbcon, "article_grant", article_grant, append = T, row.names = FALSE)
dbWriteTable(dbcon, "article_history", article_history, append = T, row.names = FALSE)
dbWriteTable(dbcon, "article_e_location", article_e_location, append = T, row.names = FALSE)
dbWriteTable(dbcon, "article_author", article_author, append = T, row.names = FALSE)
dbWriteTable(dbcon, "article_publication_type", article_publication_type, append = T, row.names
 = FALSE)
```

test

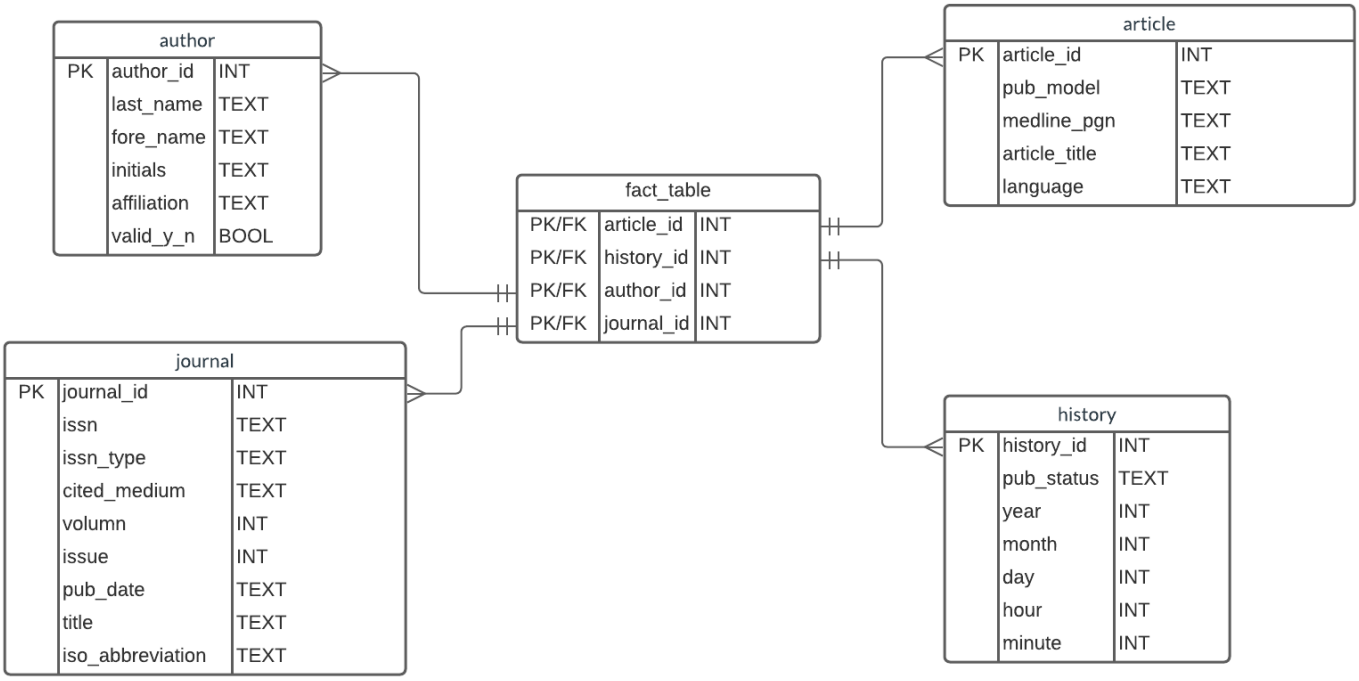
```
select article.article_id, last_name from article join article_author on article.article_id = ar
ticle_author.article_id join author on author.author_id = article_author.author_id
```

Displaying records 1 - 10

article_id	last_name
1	Kuo
1	Edwards
1	Mazumdar
1	Memtsoudis
2	Stundner
2	Kirksey
2	Chiu
2	Mazumdar
2	Poultsides
2	Gerner

Part 2 (40 pts) Create Star/Snowflake Schema

1. (20 pts) Create and populate a star schema with dimension and transaction fact tables. Each row in the fact table will represent one article fact. Include the image of an updated ERD that contains the fact table and any additional required dimension tables. Populate the star schema in R. When building the schema, look a head to Part 3 as the schema is dependent on the eventual OLAP queries.




```
dbfile = "Practicum.II.2.sqlite"
dbcon2 <- dbConnect(RSQLite::SQLite(), paste0(fpath, dbfile))
```

create tables

```
CREATE TABLE author(
  author_id INT,
  last_name TEXT,
  fore_name TEXT,
  initials TEXT,
  affiliation TEXT,
  valid_y_n BOOLEAN,
  PRIMARY KEY (author_id)
);
```

```
CREATE TABLE journal(
  journal_id INT,
  issn TEXT,
  issn_type TEXT,
  cited_medium TEXT,
  volumn INT,
  issue INT,
  pub_date TEXT,
  title TEXT,
  iso_abbreviation TEXT,
  PRIMARY KEY (journal_id)
);
```

```
CREATE TABLE article(
  article_id INT,
  pub_model TEXT,
  medline_png TEXT,
  article_title TEXT,
  language TEXT,
  PRIMARY KEY (article_id)
);
```

```
CREATE TABLE history(
  history_id INT,
  pub_status TEXT,
  year INT,
  month INT,
  day INT,
  hour INT,
  minute INT,
  PRIMARY KEY (history_id)
);
```

```
CREATE TABLE fact_table(
  article_id INT,
  history_id INT,
  author_id INT,
  journal_id INT,
  PRIMARY KEY (article_id, history_id, author_id, journal_id),
  FOREIGN KEY (article_id) REFERENCES article(article_id),
  FOREIGN KEY (history_id) REFERENCES history(history_id),
  FOREIGN KEY (author_id) REFERENCES author(author_id),
  FOREIGN KEY (journal_id) REFERENCES journal(journal_id)
)
```

update article table

```
library(sqldf)
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
```

```
##
## Attaching package: 'gsubfn'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     fn
```

```
sqlCmd = 'SELECT article_id, pub_model, medline_png, article_title, language FROM article'
new_article = dbGetQuery(dbcon, sqlCmd)
```

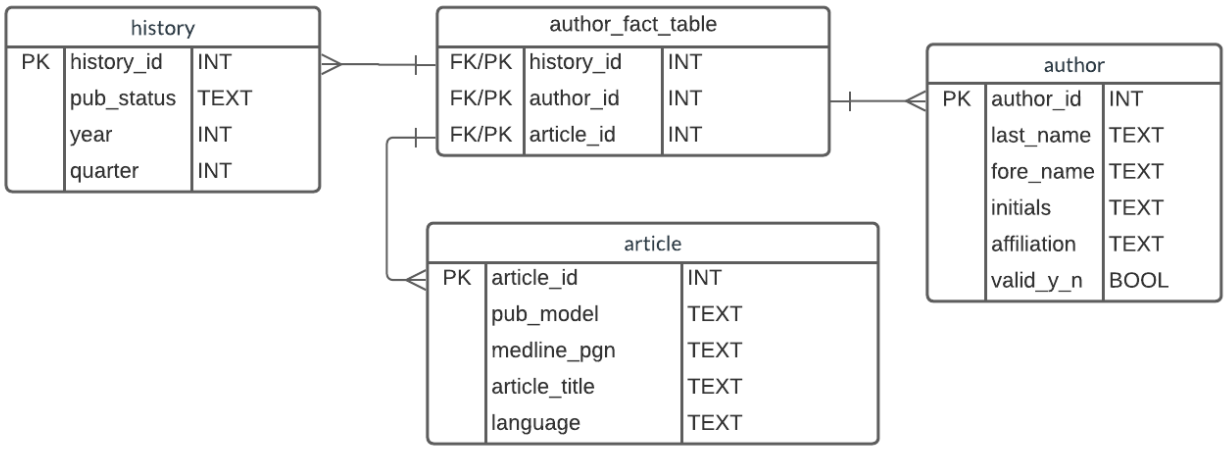
per-join for fact table

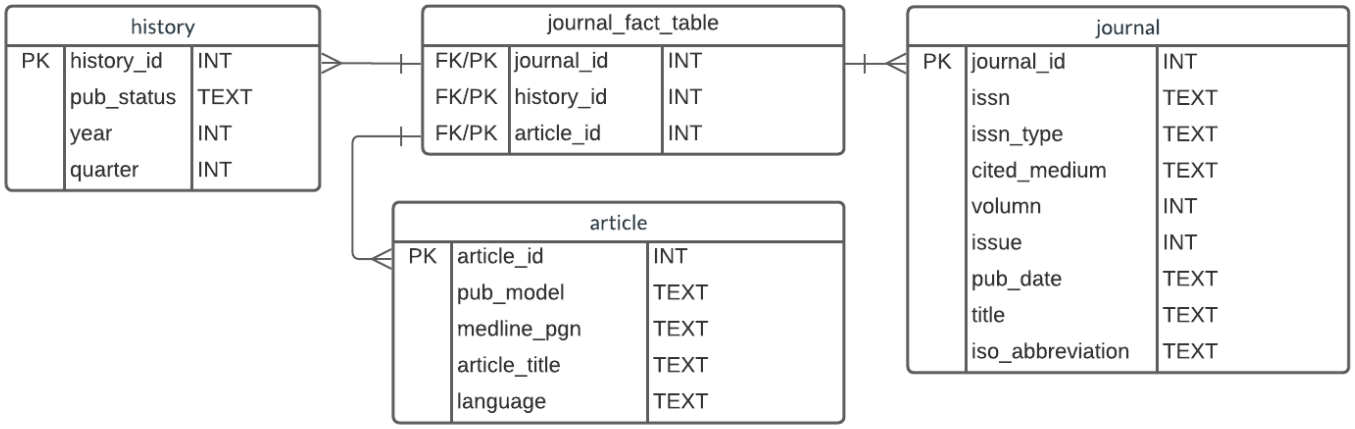
```
sqlCmd = 'SELECT article.article_id, history_id, author_id, article.journal_id FROM article JOIN
article_history ON article.article_id = article_history.article_id JOIN article_author ON articl
e.article_id = article_author.article_id JOIN journal ON article.journal_id = journal.journal_i
d'
fact_table = dbGetQuery(dbcon, sqlCmd)
```

populate tables

```
dbWriteTable(dbcon2, "article", new_article, append = T, row.names = FALSE)
dbWriteTable(dbcon2, "fact_table", fact_table, append = T, row.names = FALSE)
dbWriteTable(dbcon2, "journal", journal, append = T, row.names = FALSE)
dbWriteTable(dbcon2, "history", history, append = T, row.names = FALSE)
dbWriteTable(dbcon2, "author", author, append = T, row.names = FALSE)
```

2. (20 pts) In the same schema as the previous step, create and populate a summary fact table or revise the previously created fact table that represents number of articles per time period (quarter, year) by author and by journal. Include the image of an updated ERD that contains the fact table. Populate the fact table in R. When building the schema, look a head to Part 3 as the schema is dependent on the eventual OLAP queries.





calculate new field quarter for history table

```
history$month <- as.numeric(history$month)
n = nrow(history)
quarter = c()
for (i in 1:n){
  # print(history$month[i])
  if (history$month[i] >= 1 & history$month[i] <= 3){
    quarter <- c(quarter, 1)
  }
  else if (history$month[i] >=4 & history$month[i] <= 6){
    quarter <- c(quarter, 2)
  }
  else if (history$month[i] >=7 & history$month[i] <= 9){
    quarter <- c(quarter, 3)
  }
  else if (history$month[i] >=10){
    quarter <- c(quarter, 4)
  }
}
new_history <- history
new_history$quarter <- quarter
new_history <- new_history[ -c(3:6) ]
```

2.1 fact table by author

```
dbfile = "Practicum.II.3.sqlite"
dbcon3 <- dbConnect(RSQLite::SQLite(), paste0(fpath, dbfile))
```

calculate author_fact_table

```
sqlCmd = 'SELECT article.article_id, history_id, author_id FROM article JOIN article_history on
 article.article_id = article_history.article_id JOIN article_author ON  article.article_id = ar
ticle_author.article_id'
author_fact_table = dbGetQuery(dbcon, sqlCmd)
```

create tables

```
CREATE TABLE author(
  author_id INT,
  last_name TEXT,
  fore_name TEXT,
  initials TEXT,
  affiliation TEXT,
  valid_y_n BOOLEAN,
  PRIMARY KEY (author_id)
);
```

```
CREATE TABLE article(
  article_id INT,
  pub_model TEXT,
  medline_png TEXT,
  article_title TEXT,
  language TEXT,
  PRIMARY KEY (article_id)
);
```

```
CREATE TABLE history(
  history_id INT,
  pub_status TEXT,
  year INT,
  quarter INT,
  PRIMARY KEY (history_id)
);
```

```
CREATE TABLE fact_table(
  article_id INT,
  history_id INT,
  author_id INT,
  PRIMARY KEY (article_id, history_id, author_id),
  FOREIGN KEY (article_id) REFERENCES article(article_id),
  FOREIGN KEY (history_id) REFERENCES history(history_id),
  FOREIGN KEY (author_id) REFERENCES author(author_id)
)
```

```
dbWriteTable(dbcon3, "history", new_history, append = T, row.names = FALSE)
dbWriteTable(dbcon3, "author", author, append = T, row.names = FALSE)
dbWriteTable(dbcon3, "article", new_article, append = T, row.names = FALSE)
dbWriteTable(dbcon3, "fact_table", author_fact_table, append = T, row.names = FALSE)
```

2.2 fact table by journal

```
dbfile = "Practicum.II.4.sqlite"
dbcon4 <- dbConnect(RSQLite::SQLite(), paste0(fpath, dbfile))
```

```
CREATE TABLE journal(
  journal_id INT,
  issn TEXT,
  issn_type TEXT,
  cited_medium TEXT,
  volumn INT,
  issue INT,
  pub_date TEXT,
  title TEXT,
  iso_abbreviation TEXT,
  PRIMARY KEY (journal_id)
);
```

```
CREATE TABLE article(
  article_id INT,
  pub_model TEXT,
  medline_png TEXT,
  article_title TEXT,
  language TEXT,
  PRIMARY KEY (article_id)
);
```

```
CREATE TABLE history(
  history_id INT,
  pub_status TEXT,
  year INT,
  quarter INT,
  PRIMARY KEY (history_id)
);
```

```
CREATE TABLE fact_table(
  article_id INT,
  history_id INT,
  journal_id INT,
  PRIMARY KEY (article_id, history_id, journal_id),
  FOREIGN KEY (article_id) REFERENCES article(article_id),
  FOREIGN KEY (history_id) REFERENCES history(history_id),
  FOREIGN KEY (journal_id) REFERENCES journal(journal_id)
)
```

calculate journal_fact_table

```
sqlCmd = 'SELECT article.article_id, history_id, journal.journal_id FROM article JOIN article_hi
story on article.article_id = article_history.article_id JOIN journal ON  article.journal_id = j
ournal.journal_id'
journal_fact_table = dbGetQuery(dbcon, sqlCmd)
```

```
dbWriteTable(dbcon4, "history", new_history, append = T, row.names = FALSE)
dbWriteTable(dbcon4, "journal", journal, append = T, row.names = FALSE)
dbWriteTable(dbcon4, "article", new_article, append = T, row.names = FALSE)
dbWriteTable(dbcon4, "fact_table", journal_fact_table, append = T, row.names = FALSE)
```

test

```
select * from fact_table
```

Displaying records 1 - 10

article_id	history_id	author_id
1	1	1
1	1	2
1	1	3
1	1	4
1	2	1
1	2	2
1	2	3
1	2	4
1	3	1
1	3	2

Part 3 (20 pts) Explore and Mine Data

1. (20 pts) Write queries using your data warehouse to explore whether the publications show a seasonal pattern. For example, create a line graph that shows the number of publications for all journals each quarter or the average number of days between submission and publication. If necessary, adjust your fact table(s) as needed to support your new queries.

If you need to update the fact table, document your changes and your reasons why the changes are needed.

test

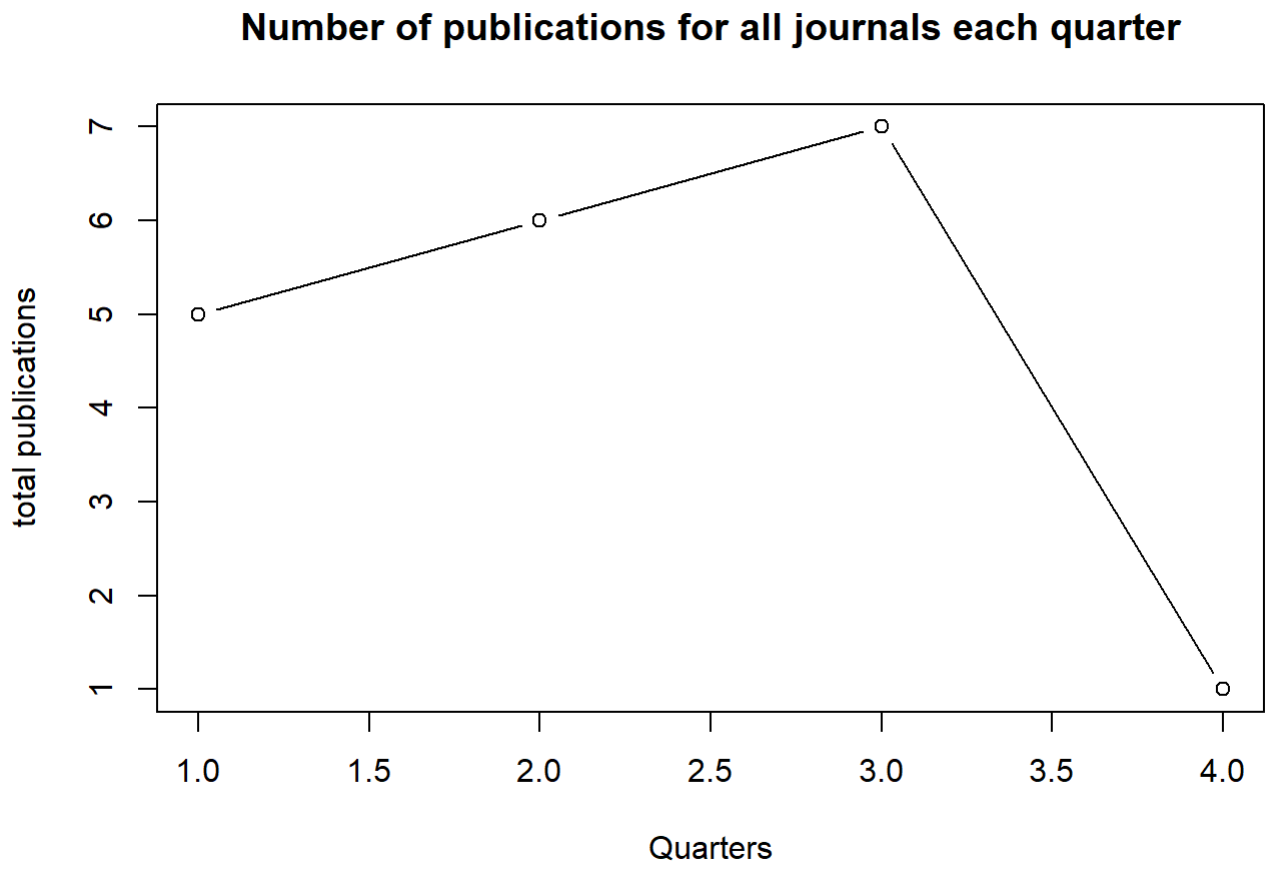
```
SELECT  count(journal_id) AS num_of_pub, quarter FROM fact_table JOIN history on fact_table.hist
ory_id = history.history_id WHERE history.pub_status = 'medline' GROUP BY quarter
```

4 records

num_of_pub	quarter
5	1
6	2
7	3
1	4

Number of publications for all journals each quarter

```
sqlCmd = 'SELECT  count(journal_id) AS num_of_pub, quarter FROM fact_table JOIN history on fact_
table.history_id = history.history_id WHERE history.pub_status = "medline" GROUP BY quarter'
num_of_pub = dbGetQuery(dbcon4, sqlCmd)
# plot the graph
plot(num_of_pub$quarter, num_of_pub$num_of_pub, type = "b", main = "Number of publications for a
ll journals each quarter", xlab = "Quarters", ylab = "total publications")
```



test

```
select count(article_id), quarter from fact_table JOIN author ON fact_table.author_id = author.author_id JOIN history on fact_table.history_id = history.history_id WHERE last_name = 'Chiu' and history.pub_status = 'received' GROUP BY quarter
```

3 records

count(article_id)	quarter
1	1
1	2
2	3

test

```
select count(article_id), quarter from fact_table JOIN author ON fact_table.author_id = author.author_id JOIN history on fact_table.history_id = history.history_id WHERE last_name = 'Chiu' and history.pub_status = 'accepted' GROUP BY quarter
```

3 records

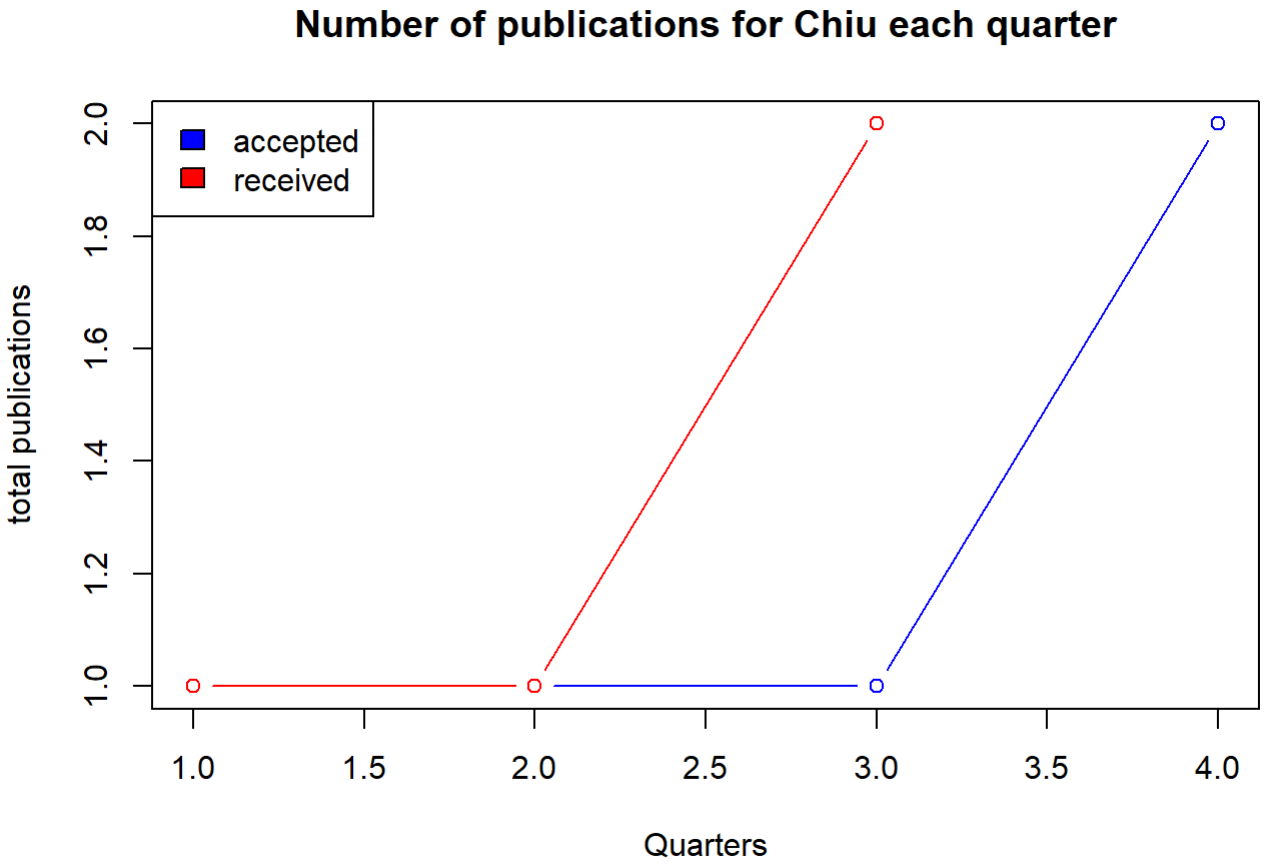
count(article_id)	quarter
1	2
2	3
1	4

Number of received and accepted publications for Chiu each quarter

```
sqlCmd = 'select count(article_id) as count, quarter from fact_table JOIN author ON fact_table.a
uthor_id = author.author_id JOIN history on fact_table.history_id = history.history_id WHERE las
t_name = "Chiu" and history.pub_status = "received" GROUP BY quarter'
chiu_received = dbGetQuery(dbcon3, sqlCmd)

sqlCmd = 'select count(article_id) as count, quarter from fact_table JOIN author ON fact_table.a
uthor_id = author.author_id JOIN history on fact_table.history_id = history.history_id WHERE las
t_name = "Chiu" and history.pub_status = "accepted" GROUP BY quarter'
chiu_accepted = dbGetQuery(dbcon3, sqlCmd)
```

```
plot(chiu_accepted$quarter, chiu_received$count, type = "b", main = "Number of publications for
 Chiu each quarter", xlab = "Quarters", ylab = "total publications", col="blue", xlim=c(1,4))
lines(chiu_received$quarter, chiu_received$count, type = "b", xlab = "Quarters", ylab = "total p
ublications", col="red")
legend("topleft",
c("accepted","received"),
fill=c("blue","red")
)
```


Number of publications for Chiu each quarter
accepted
received
total publications
1.0
1.2
1.4
1.6
1.8
2.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Quarters