Assignment / Explore Query Planning and Indexing

Krishnappa, Kushal

Spring 2025

```r
# clear the environment before starting
rm(list = ls())
```

```r
#*****************************************
#* Install Required Packages
#* @param packages - list of packages
#*****************************************
installPackagesOnDemand <- function(packages) {
  installed_packages <- packages %in% rownames(installed.packages())
  if (any(installed_packages == FALSE)) {
    install.packages(packages[!installed_packages])
  }
}

#*****************************************
#* Load Required Packages
#* @param packages - list of packages
#*****************************************
loadRequiredPackages <- function(packages) {
  # load required packages
  for (package in packages) {
    suppressMessages({
      library(package, character.only = TRUE)
    })
  }
}

# required packages
packages <- c("RSQLite", "DBI", "testthat")

# install and load required packages
installPackagesOnDemand(packages)
loadRequiredPackages(packages)
```

```r
#*****************************************
#* Connect to SQLite Database
#*****************************************
connectToDatabase <- function(dbName) {
  return (dbConnect(RSQLite::SQLite(), dbname = dbName))
}
```

```
#****************************************
#* Query to DB to check connection
#****************************************
queryTheDbToCheckConnection <- function(dbCon) {
  query <- "SELECT * FROM film LIMIT 3;"
  queryResult <- dbGetQuery(dbCon, query)
  test_that("Query to the db returns table with size 3", {
    expect_equal(nrow(queryResult), 3)
  })
}

dbCon <- connectToDatabase("sakila.db")
queryTheDbToCheckConnection(dbCon)
```

```
## Test passed
```

Question 1

```
#****************************************
#* Get All User Defined Indexes
#* @param dbCon - database connection
#****************************************
getAllUserDefinedIndexes <- function(dbCon) {
  return (dbGetQuery(dbCon, "SELECT name FROM sqlite_master WHERE type='index';"))
}

#****************************************
#* Remove All User Defined Indexes
#* @param dbCon - database connection
#****************************************
removeAllUserDefinedIndexes <- function(dbCon) {
  indexes <- getAllUserDefinedIndexes(dbCon)
  for (index in indexes$name) {
    if (!grepl("sqlite_autoindex", index)) {
      dbExecute(dbCon, sprintf("DROP INDEX IF EXISTS %s;", index))
    }
  }
}


#****************************************
#* Query to Get Film Count Per Rating
#****************************************
queryToGetFilmCountPerRating <- function() {
  return ("
  SELECT rating AS Rating, COUNT(*) AS FilmCount
  FROM film
  GROUP BY rating
  ORDER BY rating;
  ")
}
```

```
#****************************************
#* Db Query to Get Film Count Per Rating
#* @param dbCon - database connection
#****************************************
getFilmCountPerRating <- function(dbCon) {
  return (dbGetQuery(dbCon, queryToGetFilmCountPerRating()))
}

#****************************************
#* Display Film Count Per Rating
#* @param dbCon - database connection
#****************************************
displayFilmCountPerRating <- function(dbCon) {
  result <- getFilmCountPerRating(dbCon)
  print(result)
}

removeAllUserDefinedIndexes(dbCon)
displayFilmCountPerRating(dbCon)
```

```
##   Rating FilmCount
## 1      G       178
## 2  NC-17       210
## 3     PG       194
## 4  PG-13       223
## 5      R       195
```

Question 2

```
#****************************************
#* Get Query Plan from SQLite DB
#* @param dbCon - database connection
#* @param query - query to get plan for
#****************************************
getQueryPlan <- function(dbCon, query){
  return(dbGetQuery(dbCon, sprintf("EXPLAIN QUERY PLAN %s;", query)))
}

#****************************************
#* Display Query Plan for Film Count Per Rating
#* @param dbCon - database connection
#****************************************
displayQueryPlanForFilmCountPerRating <- function(dbCon) {
  result <- getQueryPlan(dbCon, queryToGetFilmCountPerRating())
  print(result)
}

displayQueryPlanForFilmCountPerRating(dbCon)
```

```
##   id parent notused                      detail
```

```
## 1  7      0     216                     SCAN film
## 2  9      0       0 USE TEMP B-TREE FOR GROUP BY
```

Question 3

```
#****************************************
#* Query to Get Info on Film Zorro Ark
#****************************************
queryToGetInfoOnFilmZorroArk <- function() {
  return ("
  SELECT  title,length,rental_rate,release_year
  FROM film
  WHERE title = 'ZORRO ARK';
  ")
}

#****************************************
#* DB Query to Get Info on Film Zorro Ark
#* @param dbCon - database connection
#****************************************
getInfoOnFilmZorroArk <- function(dbCon) {
  return (dbGetQuery(dbCon, queryToGetInfoOnFilmZorroArk()))
}

queryStartTime <- Sys.time()
result <- getInfoOnFilmZorroArk(dbCon)
queryEndTime <- Sys.time()
# get the time taken to fetch the data in milliseconds
queryTimeToFetchInfoOnFilmZorroArk <- ((queryEndTime - queryStartTime)*1000)
print(result)
```

```
##       title length rental_rate release_year
## 1 ZORRO ARK     50        4.99         2006
```

Question 4

```
#****************************************
#* Display Query Plan for Info on Film Zorro Ark
#* @param dbCon - database connection
#****************************************
displayQueryPlanForInfoOnFilmZorroArk <- function(dbCon) {
  result <- getQueryPlan(dbCon, queryToGetInfoOnFilmZorroArk())
  print(result)
}

displayQueryPlanForInfoOnFilmZorroArk(dbCon)
```

```
##   id parent notused    detail
## 1  2      0     216 SCAN film
```

Question 5

```
#****************************************
#* Query to Create Title Index
#****************************************
queryToCreateTitleIndex <- function() {
  return ("
  CREATE INDEX IF NOT EXISTS TitleIndex
  ON film(title);
  ")
}

#****************************************
#* Create Title Index
#* @param dbCon - database connection
#****************************************
createTitleIndex <- function(dbCon) {
  invisible(dbExecute(dbCon, queryToCreateTitleIndex()))
}

createTitleIndex(dbCon)
```

Question 6

```
queryStartTime <- Sys.time()
result <- getInfoOnFilmZorroArk(dbCon)
queryEndTime <- Sys.time()
# get the time taken to fetch the data in milliseconds
queryTimeToFetchInfoOnFilmZorroArkAfterIndex <- ((queryEndTime - queryStartTime)*1000)
print(result)
```

```
##       title length rental_rate release_year
## 1 ZORRO ARK     50        4.99         2006
```

```
# check for the method in question 4
displayQueryPlanForInfoOnFilmZorroArk(dbCon)
```

```
##   id parent notused                                        detail
## 1  3      0      63 SEARCH film USING INDEX TitleIndex (title=?)
```

Question 7

- The query plan for question 4 and question 6 are different.
- Question 6 uses the index to fetch the data. On the contrary, question 4 scans the entire film table to fetch the data.
- From the query plan of question 6, we can see that the query is using the index 'TitleIndex' to fetch the data.
- This can be verified by checking the 'USING INDEX' keyword in the query plan.

Question 8

Performance Measurement: Time to fetch "Zorro Ark" data was 0.433 ms before indexing and 0.747 ms after indexing (0.314 ms slower with the index).

Analysis: Despite indexes typically improving query performance, the indexed query was actually slower in this case. For small tables, full table scans can be faster than index lookups. This is usually because the overhead of scanning the index and then fetching the data from the table is more than scanning the entire table and fetching the data. Hence, the time taken to fetch the data before and after creating the index is almost the same or greater in case of smaller datasets.

Question 9

```
#****************************************
#* Query to Get Actors Containing WIL in Last Name
#****************************************
queryToGetActorsContainingWILInLastName <- function() {
  return ("
  SELECT a.first_name AS FirstName,
         a.last_name AS LastName,
         COUNT(f.film_id) AS NumberOfFilms
  FROM actor a
  JOIN film_actor fa ON a.actor_id = fa.actor_id
  JOIN film f ON fa.film_id = f.film_id
  WHERE a.last_name LIKE 'WIL%' COLLATE NOCASE
  GROUP BY a.actor_id, a.first_name, a.last_name
  ORDER BY a.last_name, a.first_name;
  ")
}

#****************************************
#* DB Query to Get Actors Containing WIL in Last Name
#* @param dbCon - database connection
#****************************************
getActorsContainingWILInLastName <- function(dbCon) {
  return (dbGetQuery(dbCon, queryToGetActorsContainingWILInLastName()))
}

#****************************************
#* Display Actors Containing WIL in Last Name
#* @param dbCon - database connection
#****************************************
displayActorsContainingWILInLastName <- function(dbCon) {
  result <- getActorsContainingWILInLastName(dbCon)
  print(result)
}

displayActorsContainingWILInLastName(dbCon)
```

```
##   FirstName LastName NumberOfFilms
## 1   GROUCHO WILLIAMS            25
## 2    MORGAN WILLIAMS            27
```

```
## 3      SEAN WILLIAMS           26
## 4       BEN   WILLIS           33
## 5      GENE   WILLIS           23
## 6  HUMPHREY   WILLIS           26
## 7      WILL   WILSON           31
```

Question 10

```
#**************************************
#* Display Query Plan for Actors Containing WIL in Last Name
#* @param dbCon - database connection
#**************************************
displayQueryPlanForActorsContainingWILInLastName <- function(dbCon) {
  result <- getQueryPlan(dbCon, queryToGetActorsContainingWILInLastName())
  print(result)
}

displayQueryPlanForActorsContainingWILInLastName(dbCon)
```

```
##   id parent notused                                                detail
## 1  9      0     214 SCAN fa USING COVERING INDEX sqlite_autoindex_film_actor_1
## 2 11      0      45               SEARCH f USING INTEGER PRIMARY KEY (rowid=?)
## 3 14      0      45               SEARCH a USING INTEGER PRIMARY KEY (rowid=?)
## 4 20      0       0                                USE TEMP B-TREE FOR GROUP BY
## 5 63      0       0                                USE TEMP B-TREE FOR ORDER BY
```

- The query plan shows that it is not using index TitleIndex created earlier.
- The behaviour is due to the use of LIKE operator in the query. It is an approximate search and the index is not useful in this case.
- The table is scanned row by row to fetch the data. Which confirms that the index is not used in this case.

```
dbDisconnect(dbCon) # disconnect from the database
```